IMSL
by Perforce

IMSL® Fortran Stat Library

Version 2021.0

PERFORCE

www.perforce.com

ACKNOWLEDGMENTS

IMSL
https://www.imsl.com/

Contents

Introduction

Basic Statistics

Regression

Categorical and Discrete Data Analysis

Nonparametric Statistics

Tests of Goodness of Fit and Randomness

Time Series Analysis and Forecasting

Covariance Structures and Factor Analysis

Discriminant Analysis

Density and Hazard Estimation

Line Printer Graphics

Probability Distribution Functions

Random Number Generation

Utilities

Mathematical Support

Reference Material

Introduction

The IMSL Fortran Numerical Library

The IMSL Fortran Numerical Library consists of two separate but coordinated Libraries that allow easy user access. These Libraries are organized as follows:

- MATH/LIBRARY general applied mathematics and special functions

  The User's Guide for IMSL MATH/LIBRARY has two parts:

  a. MATH LIBRARY

  b. MATH LIBRARY Special Functions

- STAT LIBRARY statistics

Most of the routines are available in both single and double precision versions. Many routines for linear solvers and eigensystems are also available for complex and complex-double precision arithmetic. The same user interface is found on the many hardware versions that span the range from personal computer to supercomputer.

This library is the result of a merging of the products: IMSL Fortran Numerical Libraries and IMSL Fortran 90 Library.

User Background

Vendor Supplied Libraries Usage

The IMSL Fortran Numerical Library contains functions which may take advantage of functions in vendor supplied libraries such as Intel's® Math Kernel Library (MKL) or Sun's High Performance Library. Functions in the vendor supplied libraries are finely tuned for performance to take full advantage of the environment for which they are supplied. For these functions, the user of the IMSL Fortran Numerical Library has the option of linking to code which is based on either the IMSL legacy functions or the functions in the vendor supplied library. The following icon in the function documentation alerts the reader when this is the case:

Details on linking to the appropriate IMSL Library and alternate vendor supplied libraries are explained in the online README file of the product distribution.

Getting Started

The IMSL STAT LIBRARY is a collection of FORTRAN subroutines and functions useful in research and statistical analysis. Each routine is designed and documented to be used in research activities as well as by technical specialists.

To use any of these routines, you must write a program in FORTRAN (or possibly some other language) to call the STAT LIBRARY routine. Each routine conforms to established conventions in programming and documentation. We give first priority in development to efficient algorithms, clear documentation, and accurate results. The uniform design of the routines makes it easy to use more than one routine in a given application. Also, you will find that the design consistency enables you to apply your experience with one STAT LIBRARY routine to all other IMSL routines that you use.

Finding the Right Routine

The STAT LIBRARY is organized into chapters; each chapter contains routines with similar computational or analytical capabilities. To locate the right routine for a given problem, you may use either the table of contents located in each chapter introduction, or one of the indexes at the end of this manual.

Often the quickest way to use the STAT LIBRARY is to find an example similar to your problem and then to mimic the example. Each routine document has at least one example demonstrating its application. The example for a routine may be created simply for illustration, it may be from a textbook (with reference to the source) or it may be from the statistical literature, in which case IMSL routine **GDATA** retrieves the data set.

Organization of the Documentation

This manual contains a concise description of each routine, with at least one demonstrated example of each routine, including sample input and results. You will find all information pertaining to the STAT LIBRARY in this manual. Moreover, all information pertaining to a particular routine is in one place within a chapter.

Each chapter begins with an introduction followed by a table of contents that lists the routines included in the chapter. Documentation of the routines consists of the following information:

- IMSL Routine's Generic Name
- Purpose: a statement of the purpose of the routine. If the routine is a function rather than a subroutine the purpose statement will reflect this fact.
- Function Return Value: a description of the return value (for functions only).
- Required Arguments: a description of the required arguments in the order of their occurrence. Input arguments usually occur first, followed by input/output arguments, with output arguments described last. Futhermore, the following terms apply to arguments:

  Input Argument must be initialized; it is not changed by the routine.

  Input/Output Argument must be initialized; the routine returns output through this argument; cannot be a constant or an expression.

  Input or Output Select appropriate option to define the argument as either input or output. See individual routines for further instructions.

  Output No initialization is necessary; cannot be a constant or an expression. The routine returns output through this argument.
- Optional Arguments: a description of the optional arguments in the order of their occurrence.
- Fortran 90 Interface: a section that describes the generic and specific interfaces to the routine.
- Fortran 77 Style Interfaces: an optional section, which describes Fortran 77 style interfaces, is supplied for backwards compatibility with previous versions of the Library.
- Description: a description of the algorithm and references to detailed information. In many cases, other IMSL routines with similar or complementary functions are noted.
- Comments: details pertaining to code usage.
- Programming notes: an optional section that contains programming details not covered elsewhere.
- Example: at least one application of this routine showing input and required dimension and type statements.

- Output: results from the example(s). **Note** that unique solutions may differ from platform to platform.
- Additional Examples: an optional section with additional applications of this routine showing input and required dimension and type statements.

Naming Conventions

The names of the routines are mnemonic and unique. Most routines are available in both a single precision and a double precision version, with names of the two versions sharing a common root. The root name is also the generic interface name. The name of the double precision specific version begins with a "**D_**." The single precision specific version begins with an "**S_**". For example, the following pairs are precision specific names of routines in the two different precisions: **S_UVSTA**/**D_UVSTA** (the root is "**UVSTA** ," for "Basic Univariate Statistics") and **S_TWFRQ**/**D_TWFRQ** (the root is "**TWFRQ**," for "Two-Way Frequency Table"). Of course the generic name can be used as an entry point for all precisions supported.

Except when expressly stated otherwise, the names of the variables in the argument lists follow the FORTRAN default type for integer and floating point. In other words, a variable whose name begins with one of the letters "**I**" through "**N**" is of type **INTEGER**, and otherwise is of type **REAL** or **DOUBLE PRECISION**, depending on the precision of the routine.

An assumed size array with more than one dimension that is used as a FORTRAN argument can have an assumed-size declarator for the last dimension only. In the MATH/LIBRARY routines, the information about the first dimension is passed by a variable with the prefix "**LD**" and with the array name as the root. For example, the argument **LDA** contains the leading dimension of array *A*. In most cases, information about the dimensions of arrays is obtained from the array through the use of Fortran 90's *size* function. Therefore, arguments carrying this type of information are usually defined as optional arguments.

Where appropriate, the same variable name is used consistently throughout a chapter in the STAT LIBRARY. For example, in the routines for random number generation, **NR** denotes the number of random numbers to be generated, and **R** or **IR** denotes the array that stores the numbers.

When writing programs accessing the STAT LIBRARY, the user should choose FORTRAN names that do not conflict with names of IMSL subroutines, functions, or named common blocks. The careful user can avoid any conflicts with IMSL names if, in choosing names, the following rules are observed:

- Do not choose a name that appears in the Alphabetical Summary of Routines, at the end of the *User's Manual*, nor one of these names preceded by a **D**, **S_**, **D_**, **C_**, or **Z_**.
- Do not choose a name consisting of more than three characters with a numeral in the second or third position.

For further details, see the section on Reserved Names in the Reference Material section of this manual.

Using Library Subprograms

The documentation for the routines uses the generic name and omits the prefix, and hence the entire suite of routines for that subject is documented under the generic name.

Examples that appear in the documentation also use the generic name. To further illustrate this principle, note the **OWFRQ** documentation (see Chapter 1, "Basic Statistics"), for tallying observation into a one-way frequency table. A description is provided for just one data type. There are two documented routines in this subject area: **S_OWFRQ** and **D_OWFRQ**.

These routines constitute single-precision and double-precision versions of the code.

The appropriate routine is identified by the Fortran 90 compiler. Use of a module is required with the routines. The naming convention for modules joins the suffix "**_int**" to the generic routine name. Thus, the line "use **OWFRQ_INT**" is inserted near the top of any routine that calls the subprogram "**OWFRQ**". More inclusive modules are also available. For example, the module named "**imsl_libraries**" contains the interface modules for all routines in the library.

Programming Conventions

In general, the IMSL STAT LIBRARY codes are written so that computations are not affected by underflow, provided the system (hardware or software) places a zero value in the register. In this case, system error messages indicating underflow should be ignored.

IMSL codes also are written to avoid overflow. A program that produces system error messages indicating overflow should be examined for programming errors such as incorrect input data, mismatch of argument types, or improper dimensioning.

In many cases, the documentation for a routine points out common pitfalls that can lead to failure of the algorithm.

Library routines detect error conditions, classify them as to severity, and treat them accordingly. This error-handling capability provides automatic protection for the user without requiring the user to make any specific provisions for the treatment of error conditions. See the section on User Errors in the Reference Material for further details.

Module Usage

Users are required to incorporate a "`use`" statement near the top of their program for the IMSL routine being called when writing new code that uses this library. However, legacy code which calls routines in the previous version of the library without the use of a "`use`" statement will continue to work as before. Also, code which employed the "`use numerical_libraries`" statement from the previous version of the library will continue to work properly with this version of the library.

Users wishing to update existing programs so as to call other routines from this library should incorporate a use statement for the specific new routine being called. (Here, the term "new routine" implies any routine in the library, only "new" to the user's program.) Use of the more encompassing "`imsl_libraries`" module in this case could result in argument mismatches for the "old" routine(s) being called. (This would be caught by the compiler.)

Users wishing to update existing programs so as to call the new generic versions of the routines must change their calls to the existing routines so as to match the new calling sequences and use either the routine specific interface modules or the all encompassing "`imsl_libraries`" module.

Programming Tips

It is strongly suggested that users force all program variables to be explicitly typed. This is done by including the line "`IMPLICIT NONE`" as close to the first line as possible. Study some of the examples accompanying an IMSL Fortran Library routine early on. These examples are available online as part of the product.

Each subject routine called or otherwise referenced requires the "`use`" statement for an interface block designed for that subject routine. The contents of this interface block are the interfaces to the separate routines available for that subject. Packaged descriptive names for option numbers that modify documented optional data or internal parameters might also be provided in the interface block. Although this seems like an additional complication, many typographical errors are avoided at an early stage in development through the use of these interface blocks. The "`use`" statement is required for each routine called in the user's program.

However, if one is only using the Fortran 77 interfaces supplied for backwards compatibility then the "`use`" statements are not required.

Optional Subprogram Arguments

IMSL Fortran Library routines have *required* arguments and may have *optional* arguments. All arguments are documented for each routine. For example, consider the routine **ORDST** that determines order statistics. The required arguments are **X**, **NOS**, **OS**, and **NMISS**. The input data for the problem are the **X** array and **NOS**, the number of order statistics; the output is returned in the **OS** array. The number of missing values is returned in **NMISS**. This routine has as optional arguments **NOBS**, **IOPT**, and **IOS**. If one wishes to calculate a different set of order statistics than the default (the first **NOS** order statistics) then the optional argument given by the "**IOPT=**" keyword should be used in the argument list. See Example 2 of **ORDST** in Chapter 1, "Basic Statistics" for an example of this functionality.

For compatibility with previous versions of the IMSL Libraries, the **NUMERICAL_LIBRARIES** interface module includes backwards compatible positional argument interfaces to all routines which existed in the Fortran 77 version of the Library. Note that it is not necessary to use "use" statements when calling these routines by themselves. Existing programs which called these routines will continue to work in the same manner as before.

Error Handling

The routines in the IMSL STAT LIBRARY attempt to detect and report errors and invalid input. Errors are classified and are assigned a code number. By default, errors of moderate or worse severity result in messages being automatically printed by the routine. Moreover, errors of worse severity cause program execution to stop. The severity level as well as the general nature of the error is designated by an “error type” with numbers from 0 to 5. An error type 0 is no error; types 1 through 5 are progressively more severe. In most cases, you need not be concerned with our method of handling errors. For those interested, a complete description of the error-handling system is given in the Reference Material, which also describes how you can change the default actions and access the error code numbers.

Printing Results

Several routines in the IMSL STAT LIBRARY have an option for printing results. These routines have an optional argument, **IPRINT**, to control the printing. In any routine that allows printing, if **IPRINT** = 0, (the default) then no printing is done (except possibly error messages). Some routines allow various amounts of printing; one value of **IPRINT** might result in printing only summary statistics, while another value might cause more detailed statistics or intermediate results to be printed. Other routines in the STAT LIBRARY do not print any of the results. In all routines, of course, the output is returned in FORTRAN variables, so if the routine does not do printing, or if you use the default **IPRINT** value, you can print the results yourself. The STAT LIBRARY contains some special routines just for printing arrays. For example, **WRRRN** and **WRRRL** are two convenient routines for printing matrices. See Chapter 19, "Utilities" for detailed descriptions of these routines.

A commonly used routine in the examples is the IMSL routine **UMACH**, which retrieves the FORTRAN device unit number for printing the results. Because this routine obtains device unit numbers, it can be used to redirect the input or output. The section on Machine-Dependent Constants in the Reference Material contains a description of the routine **UMACH**.

Shared-Memory Multiprocessors and Thread Safety

The IMSL Fortran Numerical Library allows users to leverage the high-performance technology of shared memory parallelism (SMP) when their environment supports it. Support for SMP systems within the IMSL Library is delivered through various means, depending upon the availability of technologies such as OpenMP, high performance LAPACK and BLAS, and hardware-specific IMSL algorithms. Use of the IMSL Fortran Numerical Library on SMP systems can be achieved by using the appropriate link environment variable when building your application. Details on the available link environment variables for your installation of the IMSL Fortran Numerical Library can be found in the online README file of the product distribution.

The IMSL Fortran Numerical Library is thread-safe in those environments that support OpenMP. This was achieved by using OpenMP directives that define global variables located in the code so they are private to the individual threads. Thread safety allows users to create instances of routines running on multiple threads and to include any routine in the IMSL Fortran Numerical Library in these threads.

Missing Values

Many of the routines in the IMSL STAT LIBRARY allow the data to contain missing values. These routines recognize as a missing value the special value referred to as 'not a number,' or NaN. The actual value is different on different computers, but it can be obtained by reference to the IMSL routines **AMACH** or **DMACH**, described in the Machine-Dependent Constants section of the Reference Material. In routines that allow missing values, two common arguments are **NMISS** and **NRMISS**. The definitions of these arguments vary somewhat depending on the specific routine. However, in a data structure where the rows represent observations and the columns represent variables, **NRMISS** is the number of rows containing missing values and **NMISS** is the total number of missing values.

The way that missing values are treated depends on the individual routine, and is described in the documentation for the routine.

Routines that Accumulate Results over Several Calls

Often in statistical analyses, not all of the data are available in computer memory at once. Many of the routines in the STAT LIBRARY accept a part of the data, accumulate some statistics, and continue accepting data and accumulating statistics until all of the data have been processed. The routines that allow the data to be processed a little at a time have an argument called "`IDO`."

Using IMSL Fortran Library on Shared-Memory Multiprocessors

The IMSL Fortran Library allows users to leverage the high-performance technology of shared memory parallelism (SMP) when their environment supports it. Support for SMP systems within the IMSL Library is delivered through various means, depending upon the availability of technologies such as OpenMP, high performance BLAS, and hardware-specific IMSL algorithms. Use of the IMSL Fortran Library on SMP systems can be achieved by using the appropriate link environment variable when building your application. Details on the available link environment variables for your installation of the IMSL Fortran Library can be found in the online README file of the product distribution.

This introduction has acquainted you with a few general characteristics of IMSL STAT LIBRARY. If you are using the STAT LIBRARY at a computer center, the computer center consultant will provide the details necessary to use the IMSL routines on your computer system.

Basic Statistics

Routines

Usage Notes

Frequency Tabulations

The routines for frequency tabulations accept raw data in the form of vectors or matrices and produce counts. Two of these routines assume generally that the data are continuous and tally the observations into groups based on grouping information that the user supplies. Another routine for frequency tabulations assumes basically that the data are discrete and counts the number of observations with each value. Other analyses of discrete data or count data can be performed using IMSL routines in Chapter 5, "Categorical and Discrete Data Analysis."

Univariate Summary Statistics

The routine UVSTA computes the sample mean, variance, minimum, maximum, and other basic statistics for each variable in a data set. It also computes confidence intervals for the mean and variance if the sample is assumed to be from a normal distribution.

Ranks and Order Statistics

The routines for ranks and order statistics accept data from a single sample stored in a vector. Ranks, order statistics, and sample quantiles form the basis for many nonparametric and robust statistical techniques (see Conover 1980 and Hoaglin et al. 1983). Letter values, computed by the routine LETTR, are a special set of order statistics particularly useful in exploratory data analysis (see Hoaglin 1983).

Parametric Estimates and Tests

The routines described in this section compute statistics for simple inferences about the parameters in normal, binomial, and Poisson distributions. General discussions of estimation techniques for these distributions can be found in Johnson and Kotz (1969, 1970a, 1970b). The routine UVSTA, for univariate summary statistics, also computes statistics for simple inferences about the parameters in a single normal distribution.

Grouped Data

The routine GRPES computes several basic statistics, such as arithmetic means, geometric means, harmonic means, and moments about the arithmetic mean for grouped data. The second, third, and fourth moments are computed both with and without Sheppard's corrections.

Continuous Data in a Table

The routine **CSTAT** accepts data sets with both classification variables and response variables. The classification variables define cells in a table. Within each cell, means and sums of squares are computed for the response variables. Further analysis of the response variables, in particular, assessment of the effects of the classification variables, may be performed using the routines described in Chapter 4 on analysis of variance. An alternative for two-way tables is median polish, which is more resistant to outliers, but which is more exploratory. That is, no test is performed to confirm statistically that row and/or column effects are present. The routine **MEDPL** in this section performs median polish. (See Tukey, 1977; Velleman and Hoaglin, 1981; and Emerson and Hoaglin, 1983.) For count data (frequencies), the routines described in Chapter 5: Categorical and Discrete Data Analysis," are appropriate for determining the amount of association among the rows and columns.

OWFRQ

Tallies observations into a one-way frequency table.

Required Arguments

X — Vector of length NOBS containing the data. (Input)

K — Number of intervals. (Input)

TABLE — Vector of length K containing the counts. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

IOPT — Tallying option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Intervals of equal length, determined from the data, are used. Let XMIN and XMAX be the minimum and maximum values in X, respectively. Then, TABLE(1) is the tally of observations less than or equal to XMIN + (XMAX - XMIN)/K, TABLE(2) is the tally of observations greater than XMIN + (XMAX - XMIN)/K and less than or equal to XMIN + 2 * (XMAX - XMIN)/K, and so on. TABLE(K) is the tally of observations greater than XMAX - (XMAX - XMIN)/K.
1	Intervals of equal length are used just as in the case of IOPT = 0, except the upper and lower bounds are taken as the user supplied variables XLO and XHI, instead of the actual minimum and maximum in the data. Therefore, the first and the last intervals are semi-infinite in length. K must be greater than 2.
2	K-1 cutpoints are input in DIV. The tally in TABLE(1) is the number of observations less than or equal to DIV(1). For I greater than 1 and less than K, the tally in TABLE(I) is the number of observations greater than DIV(I - 1) and less than or equal to DIV(I). The tally in TABLE(K) is the number of observations greater than DIV(K - 1). K must be greater than 1.
3	Class marks are input in DIV and a constant class half-width is input in CLHW. The total of the elements in TABLE may be less than NOBS. The tally in TABLE(I) is the number of observations between DIV(I) - CLHW and DIV(I) + CLHW.

XLO — If IOPT = 1, XLO is the lower bound at which to begin forming the class intervals. (Input)
XLO is used only if IOPT = 1.

XHI — If IOPT = 1, XHI is the upper bound to use in forming the class intervals. (Input)
XHI is used only if IOPT = 1.

CLHW — If IOPT = 3, CLHW is the half-width of the class intervals. (Input)
CLHW is not used if IOPT is not equal to 3.

DIV — Vector of varying length and contents depending on IOPT. (Input if IOPT= 2 or 3; output if IOPT = 0 or 1.)
The contents of DIV are in ascending order.

IOPT	Contents
0	DIV is of length K containing interval midpoints. (DIV is output.)
1	DIV is of length K containing interval midpoints. Since the first and last intervals are semi-infinite in length, DIV(1) contains XLO minus half the interval length, and DIV(K) contains XHI plus half the interval length. (DIV is output.)
2	DIV is a vector of length K - 1 containing cutpoints. (DIV is input.)
3	DIV is of length K containing classmarks. (DIV is input.)

FORTRAN 90 Interface

Generic: CALL OWFRQ (X, K, TABLE [, ...])
Specific: The specific interface names are S_OWFRQ and D_OWFRQ.

FORTRAN 77 Interface

Single: CALL OWFRQ (NOBS, X, K, IOPT, XLO, XHI, CLHW, DIV, TABLE)
Double: The double precision name is DOWFRQ.

Description

The routine OWFRQ groups numerical data into categories, which can be defined in any of four different ways as chosen by IOPT. If IOPT = 0, K intervals of equal length are formed between the minimum and maximum values in the data, and then the data are tallied in these intervals. The midpoints of the intervals are output in DIV.

If IOPT = 1, K - 2 intervals of equal length are formed between XLO and XHI, and then the data are tallied in these intervals. In this option, there is one group that consists of data less than XLO and one group of data greater than XHI. This option is similar to IOPT = 0, except with this option, the midpoints of the classes are under control of the user. The midpoints of the intervals are output in DIV. The first and last values of DIV, respectively, contain XLO minus half the class width and XHI plus half the class width.

For **IOPT** = 2 or 3, the intervals need not be equally spaced. If **IOPT** = 2, the intervals need not be equal in length. In this case, the intervals are defined by their boundaries, the "cutpoints", which are input in **DIV**. The number of cutpoints is one less than the number of intervals. The first cutpoint defines the upper bound of the first interval, and the last cutpoint defines the lower bound of the last interval.

If **IOPT**= 3, the intervals are all of length twice **CLHW**, and they are centered on the class marks input in **DIV**. This option can be used to exclude portions of the data.

The examples use all of these options with the same data set.

Examples

Example 1

The data for these examples are from Hinkley (1977) and Velleman and Hoaglin (1981). They are the measurements (in inches) of precipitation in Minneapolis/St. Paul during the month of March for 30 consecutive years. In the first example, we set **IOPT** = 0. This option may be appropriate if we do not know the range of the data. Notice that the midpoints of the class intervals, output in **DIV**, are not "pretty" numbers.

```
      USE OWFRQ_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    K, NOBS
      PARAMETER  (K=10, NOBS=30)
!
      INTEGER    NOUT
      REAL       DIV(K), TABLE(K), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
           2.05/
!
      CALL UMACH (2, NOUT)
!
      CALL OWFRQ (X, K, TABLE, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT ('  Midpoints: ', 10F5.2, /, '     Counts: ', 10F5.0)
      END
```

Output

```
  Midpoints:  0.54 0.98 1.43 1.87 2.31 2.76 3.20 3.64 4.09 4.53
     Counts:    4.   8.   5.   5.   3.   1.   3.   0.   0.   1.
```

Example 2

In this example, we set **IOPT** = 1 and choose **XLO** and **XHI** so that the intervals will be 0.0 to 0.5, 0.5 to 1.0, and so on. This means that the midpoints of the class intervals, output in **DIV**, will be 0.25, 0.75, and so on.

```
      USE OWFRQ_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    K, NOBS
      PARAMETER  (K=10, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIV(K), TABLE(K), X(NOBS), XHI, XLO
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!
      CALL UMACH (2, NOUT)
      IOPT = 1
      XLO  = 0.5
      XHI  = 4.5
!
      CALL OWFRQ (X, K, TABLE, iopt=iopt, xlo=xlo, xhi=xhi, div=div)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT ('  Midpoints:  ', 10F5.2, /, '     Counts: ', 10F5.0)
      END
```

Output

```
  Midpoints:   0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75
     Counts:    2.   7.   6.   6.   4.   2.   2.   0.   0.   1.
```

Example 3

In this example, we input class boundaries in **DIV**. We choose the same intervals as in the example above: 0.0 to 0.5, 0.5 to 1.0, and so on. **DIV** begins with the first cutpoint *between* classes.

```
      USE OWFRQ_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    K, NOBS
      PARAMETER  (K=10, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIV(K-1), TABLE(K), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA DIV/0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5/
```

```
!
      CALL UMACH (2, NOUT)
      IOPT = 2
!
      CALL OWFRQ (X, K, TABLE, IOPT=IOPT, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT ('  Cutpoints:    ', 9F5.1, /, '     Counts: ', 10F5.0)
      END
```

Output

```
 Cutpoints:      0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5
    Counts:    2.   7.   6.   6.   4.   2.   2.   0.   0.   1.
```

Example 4

In this example, we set `IOPT` = 3, and set the values in `DIV` and `CLHW` so that the intervals will be the same as in the previous two examples.

```
      USE OWFRQ_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    K, NOBS
      PARAMETER  (K=10, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       CLHW, DIV(K), TABLE(K), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90,&
          2.05/
      DATA DIV/0.25, 0.75, 1.25, 1.75, 2.25, 2.75, 3.25, 3.75, 4.25,&
          4.75/
!
      CALL UMACH (2, NOUT)
      IOPT = 3
      CLHW = 0.25
!
      CALL OWFRQ (X, K, TABLE, IOPT=IOPT, CLHW=CLHW, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT (' Class marks: ', 10F5.2, /, '      Counts: ', 10F5.0)
      END
```

Output

```
 Class marks:  0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75
      Counts:    2.   7.   6.   6.   4.   2.   2.   0.   0.   1.
```

TWFRQ

Tallies observations into a two-way frequency table.

Required Arguments

X — Vector of length NOBS containing the data for one variable. (Input)

Y — Vector of length NOBS containing the data for the other variable. (Input)

KX — Number of intervals for the variable X. (Input)

KY — Number of intervals for the variable Y. (Input)

TABLE — KX by KY matrix containing the counts. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

IOPT — Tallying option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Intervals of equal lengths for each variable, determined from the data, are used. Let XMIN and XMAX be the minimum and maximum values in X, respectively, with similar meanings for YMIN and YMAX. Then, TABLE(1, 1) is the tally of observations with the X value less than or equal to XMIN + (XMAX - XMIN)/KX, and the Y value less than or equal to YMIN + (YMAX - YMIN)/KY. The other table entries are determined similarly.
1	Intervals of equal lengths are used just as in the case of IOPT = 0, except the upper and lower bounds are taken as the user-supplied variables XLO, XHI, YLO, and YHI instead of the actual minima and maxima in the data. Therefore, the first and the last intervals for both variables are semi-infinite in length. KX and KY must be greater than 2.

IOPT	Action
2	KX - 1 cutpoints are input in DIVX, and KY - 1 cutpoints are input in DIVY. The tally in TABLE(1, 1) is the number of observations for which the X value is less than or equal to DIVX(1), and the Y value is less than or equal to DIVY(1). For I greater than 1 and less than KX and J greater than 1 and less than KY, the tally in TABLE(I, J) is the number of observations with X greater than DIVX(I - 1) and less than or equal to DIVX(I) and with Y greater than DIVY(J - 1) and less than or equal to DIVY(J). The tally in TABLE(KX, KY) is the number of observations for which the X value is greater than DIVX(KX - 1) and the Y value is greater than DIVY(KY - 1). KX and KY must be greater than 1.
3	Class marks are input in DIVX and DIVY and a constant class half-width are input in CLHWX and CLHWY. The total of the elements in TABLE may be less than NOBS. The tally in TABLE(I, J) is the number of observations with X value between DIVX(I) - CLHWX and DIVX(I) + CLHWX, and with Y value between DIVY(J) - CLHWY and DIVY(J) + CLHWY.

XLO — If IOPT = 1, XLO is the lower bound at which to begin forming the class intervals for X. (Input)
XLO is only used if IOPT = 1.

YLO — If IOPT = 1, YLO is the lower bound at which to begin forming the class intervals for Y. (Input)
YLO is only used if IOPT = 1.

XHI — If IOPT = 1, XHI is the upper bound to use in forming the class intervals for X. (Input)
XHI is only used if IOPT = 1.

YHI — If IOPT = 1, is the upper bound to use in forming the class intervals for Y. (Input)
YHI is only used if IOPT = 1.

CLHWX — If IOPT = 3, CLHWX is the half-width of the class intervals for X. (Input)
CLHWX is only used if IOPT = 3.

CLHWY —If IOPT = 3, CLHWY is the half-width of the class intervals for Y. (Input)
CLHWY is only used if IOPT = 3.

DIVX — Vector of varying length and contents depending on IOPT. (Input if IOPT= 2 or 3; output if IOPT = 0 or 1)
The contents of DIVX are in ascending order.

IOPT	Contents
0	DIV is of length KX containing interval midpoints for the X variable. (DIVX is output.)
1	DIV is of length KX containing interval midpoints for the X variable. Since the first and last intervals are semi-infinite in length, DIVX(1) contains XLO - half the interval length, and DIV(KX) contains XHI + half the interval length. (DIVX is output.)
2	DIVX is a vector of length KX - 1 containing cutpoints. (DIVX is input.)
3	DIVX is of length KX containing classmarks. (DIVX is input.)

DIVY — Vector of varying length and contents depending on **IOPT**. (Input if **IOPT**= 2 or 3; output if **IOPT** = 0 or 1)
The contents of **DIVY** are in ascending order. See **DIVX**.

LDTABL — Leading dimension of **TABLE** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDTABL** = size (**TABLE**,1).

FORTRAN 90 Interface

Generic: **CALL TWFRQ** (**X**, **Y**, **KX**, **KY**, **TABLE** [, ...])
Specific: The specific interface names are **S_TWFRQ** and **D_TWFRQ**.

FORTRAN 77 Interface

Single: **CALL TWFRQ** (**NOBS**, **X**, **Y**, **KX**, **KY**, **IOPT**, **XLO**, **YLO**, **XHI**, **YHI**, **CLHWX**, **CLHWY**, **DIVX**, **DIVY**, **TABLE**, **LDTABL**)
Double: The double precision name is **DTWFRQ**.

Description

The routine **TWFRQ** groups bivariate numerical data into categories, which can be defined in any of four different ways as chosen by **IOPT**. This routine is very similar to routine **OWFRQ** for univariate data. If **IOPT**= 0, **KX** intervals of equal length are formed for the first variable (in **X**) between the minimum and maximum values in **X** and similarly **KY** intervals are formed for the second variable (in **Y**). The data are then tallied in these intervals. The midpoints of the intervals for the first variable are output in **DIVX** and those of the second in **DIVY**.

If **IOPT** = 1, **K - 2** intervals of equal length are formed between **XLO** and **XHI** for the data in **X** and likewise for **Y**. The data are then tallied in these intervals. In this option, there is one group that consists of data less than **XLO** and one group of data greater than **XHI**. This option is similar to **IOPT** = 0, except in this case, the midpoints of the classes are under control of the user. The midpoints of the intervals are output in **DIVX** and **DIVY**.

For **IOPT** = 2 or 3, the intervals need not be equally spaced. If **IOPT** = 2, the intervals need not be equal in length. In this case, the intervals are defined by their boundaries, the “cutpoints”, which are input in **DIVX** and **DIVY**. The number of cutpoints is one less than the number of intervals. The first cutpoint defines the upper bound of the first interval, and the last cutpoint defines the lower bound of the last interval.

If **IOPT** = 3, the intervals are all of length twice **CLHWX** for **X** and twice **CLHWY** for **Y**, and they are centered on the class marks input in **DIVX** and **DIVY**. This option can be used to exclude portions of the data. The examples use all of these options with the same data set.

Examples

Example 1

The data for X in these examples are the same as those used in the routine for one-way frequency tabulation, OWFRQ. The data for Y were created by adding small integers to the data in X. In the first example, we set IOPT = 0. This option may be appropriate if we do not know the range of the data. Notice that the midpoints of the class intervals, output in DIVX and DIVY, are not "pretty" numbers. Routine WRRRN, (see Chapter 19, "Utilities") is used to print the frequencies. This printing routine puts column and row numbers above and to the left of the matrix being printed. For example, the "4" in the second row and second column of the output is the first number that represents a frequency. That frequency is the number of occurrences of pairs of observations in which both values are in the lowest groups.

```
      USE TWFRQ_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    KX, KY, LDTABL, NOBS
      PARAMETER  (KX=5, KY=6, LDTABL=5, NOBS=30)
!
      INTEGER    NOUT
      REAL       DIVX(KX), DIVY(KY), TABLE(LDTABL,KY), X(NOBS), Y(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA Y/1.77, 3.74, 3.81, 2.20, 3.95, 4.20, 1.47, 3.43, 6.37, &
          3.20, 5.00, 6.09, 2.51, 4.10, 3.52, 2.62, 3.31, 3.32, 1.59, &
          2.81, 5.81, 2.87, 3.18, 4.35, 5.75, 4.48, 3.96, 2.89, 2.90, &
          5.05/
!
      CALL UMACH (2, NOUT)
!
      CALL TWFRQ (X, Y, KX, KY, TABLE, DIVX=DIVX, DIVY=DIVY)
      WRITE (NOUT,99999) DIVX, DIVY
99999 FORMAT ('  Midpoints for X (Rows):    ', 5F5.2, /, '  Midpoints ' &
            , 'for Y (Columns): ', 6F5.2)
      CALL WRRRN ('Frequencies', TABLE)
      END
```

Output

```
 Midpoints for X (Rows):     0.76 1.65 2.53 3.42 4.31
 Midpoints for Y (Columns):  1.88 2.69 3.51 4.33 5.14 5.96

                  Frequencies
         1       2       3       4       5       6
1    4.000   2.000   4.000   2.000   0.000   0.000
2    0.000   4.000   3.000   2.000   1.000   0.000
3    0.000   0.000   1.000   2.000   0.000   1.000
4    0.000   0.000   0.000   0.000   1.000   2.000
```

```
    5   0.000   0.000   0.000   0.000   0.000   1.000
```

Example 2

In this example, we set **IOPT** = 1 and choose **XLO**, **XHI**, **YLO**, and **YHI** so that the intervals will be 0 to 1, 1 to 2, and so on for **X**, and 1 to 2, 2 to 3, and so on for **Y**. This means that the midpoints of the class intervals, output in **DIVX** and **DIVY**, will be 0.5, 1.5, 2.5, and so on. The "5" in the third row and fourth column of the printed output below, (i.e., the second row and the third column of the frequencies **TABLE**) represents five pairs of observations with the **X** value between 1.0 and 2.0 and the **Y** value between 3.0 and 4.0.

```
      USE TWFRQ_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    KX, KY, LDTABL, NOBS
      PARAMETER  (KX=5, KY=6, LDTABL=5, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIVX(KX), DIVY(KY), TABLE(LDTABL,KY), &
                 X(NOBS), XHI, XLO, Y(NOBS), YHI, YLO
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA Y/1.77, 3.74, 3.81, 2.20, 3.95, 4.20, 1.47, 3.43, 6.37, &
          3.20, 5.00, 6.09, 2.51, 4.10, 3.52, 2.62, 3.31, 3.32, 1.59,&
          2.81, 5.81, 2.87, 3.18, 4.35, 5.75, 4.48, 3.96, 2.89, 2.90, &
          5.05/
!
      CALL UMACH (2, NOUT)
      IOPT = 1
      XLO  = 1.0
      XHI  = 4.0
      YLO  = 2.0
      YHI  = 6.0
!
      CALL TWFRQ (X, Y, KX, KY, TABLE, iopt=iopt, xlo=xlo, ylo=ylo,  &
                  xhi=xhi, yhi=yhi, divx=divx, divy=divy)
      WRITE (NOUT,99999) DIVX, DIVY
99999 FORMAT ('  Midpoints for X (Rows):    ', 5F5.2, /, '  Midpoints ' &
            , 'for Y (Columns): ', 6F5.2)
      CALL WRRRN ('Frequencies', TABLE)
      END
```

Output

```
  Midpoints for X (Rows):     0.50 1.50 2.50 3.50 4.50
  Midpoints for Y (Columns):  1.50 2.50 3.50 4.50 5.50 6.50

                    Frequencies
          1       2       3       4       5       6
1   3.000   2.000   4.000   0.000   0.000   0.000
2   0.000   5.000   5.000   2.000   0.000   0.000
```

```
  3   0.000   0.000   1.000   3.000   2.000   0.000
  4   0.000   0.000   0.000   0.000   0.000   2.000
  5   0.000   0.000   0.000   0.000   1.000   0.000
```

Example 3

In this example, we input class boundaries in **DIVX** and **DIVY**. We choose the same intervals as in the example above: 0 to 1, 1 to 2, and so on. **DIVX** and **DIVY** begins with the first cutpoint *between* classes.

```
      USE TWFRQ_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    KX, KY, LDTABL, NOBS
      PARAMETER  (KX=5, KY=6, LDTABL=5, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIVX(4), DIVY(5), TABLE(LDTABL,KY), X(NOBS), Y(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA Y/1.77, 3.74, 3.81, 2.20, 3.95, 4.20, 1.47, 3.43, 6.37, &
          3.20, 5.00, 6.09, 2.51, 4.10, 3.52, 2.62, 3.31, 3.32, 1.59, &
          2.81, 5.81, 2.87, 3.18, 4.35, 5.75, 4.48, 3.96, 2.89, 2.90, &
          5.05/
      DATA DIVX/1.0, 2.0, 3.0, 4.0/
      DATA DIVY/2.0, 3.0, 4.0, 5.0, 6.0/
!
      CALL UMACH (2, NOUT)
      IOPT = 2
!
      CALL TWFRQ (X, Y, KX, KY, TABLE, IOPT=IOPT, DIVX=DIVX, DIVY=DIVY)
      WRITE (NOUT,99999) DIVX, DIVY
99999 FORMAT ('  Cutpoints for X (Rows):    ', 4F5.2, /, '  Cutpoints ' &
            , 'for Y (Columns): ', 5F5.2)
      CALL WRRRN ('Frequencies', TABLE)
      END
```

Output

```
 Cutpoints for X (Rows):     1.00 2.00 3.00 4.00
 Cutpoints for Y (Columns):  2.00 3.00 4.00 5.00 6.00

                   Frequencies
         1       2       3       4       5       6
1   3.000   2.000   4.000   0.000   0.000   0.000
2   0.000   5.000   5.000   2.000   0.000   0.000
3   0.000   0.000   1.000   3.000   2.000   0.000
4   0.000   0.000   0.000   0.000   0.000   2.000
5   0.000   0.000   0.000   0.000   1.000   0.000
```

Example 4

In this example, we set **IOPT** = 3, and set the values in **DIVX**, **DIVY**, **CLHWX**, and **CLHWY** so that the intervals will be the same as in the previous two examples.

```
      USE TWFRQ_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    KX, KY, LDTABL, NOBS
      PARAMETER  (KX=5, KY=6, LDTABL=5, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       CLHWX, CLHWY, DIVX(KX), DIVY(KY), TABLE(LDTABL,KY), &
                 X(NOBS), Y(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA Y/1.77, 3.74, 3.81, 2.20, 3.95, 4.20, 1.47, 3.43, 6.37, &
          3.20, 5.00, 6.09, 2.51, 4.10, 3.52, 2.62, 3.31, 3.32, 1.59, &
          2.81, 5.81, 2.87, 3.18, 4.35, 5.75, 4.48, 3.96, 2.89, 2.90, &
          5.05/
      DATA DIVX/0.5, 1.5, 2.5, 3.5, 4.5/
      DATA DIVY/1.5, 2.5, 3.5, 4.5, 5.5, 6.5/
!
      CALL UMACH (2, NOUT)
      IOPT  = 3
      CLHWX = 0.5
      CLHWY = 0.5
!
      CALL TWFRQ (X, Y, KX, KY, TABLE, IOPT=IOPT, CLHWX=CLHWX,  &
                  CLHWY=CLHWY, DIVX=DIVX, DIVY=DIVY)
      WRITE (NOUT,99999) DIVX, DIVY
99999 FORMAT ('  Class marks for X (Rows):    ', 5F5.2, /, '  Class ', &
            'marks for Y (Columns): ', 6F5.2)
      CALL WRRRN ('Frequencies', TABLE)
      END
```

Output

```
Class marks for X (Rows):     0.50 1.50 2.50 3.50 4.50
Class marks for Y (Columns):  1.50 2.50 3.50 4.50 5.50 6.50

                    Frequencies
          1       2       3       4       5       6
1    3.000   2.000   4.000   0.000   0.000   0.000
2    0.000   5.000   5.000   2.000   0.000   0.000
3    0.000   0.000   1.000   3.000   2.000   0.000
4    0.000   0.000   0.000   0.000   0.000   2.000
5    0.000   0.000   0.000   0.000   1.000   0.000
```

FREQ

Tallies multivariate observations into a multiway frequency table.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

INDCL — Index vector of length NCLVAR containing the column numbers in X that are the classification variables. (Input)

MAXTAB — An upper bound for the total number of cells in the frequency table. (Input)
This is the product of the number of distinct values taken by all of the classification variables since the table includes the empty cells.

MAXCL — An upper bound for the sum of the number of distinct values taken by all of the classification variables. (Input)

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Output, if IDO = 1; Input/Output, if IDO = 2.)
Each variable must have more than one level.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the values of the classification variables. (Output, if IDO= 1; input/output, if
IDO = 2.)
Since in general the length of CLVAL will not be known in advance, MAXCL is an upper bound for this length. The first NCLVAL(1) elements of CLVAL contain the values for the first classification variable. The next NCLVAL(2) contain the values for the second variable. The last NCLVAL(NCLVAR) positions contain the values for the last classification variable.

TABLE — Vector of length NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR) containing the frequencies in the cells of the table to be fit. (Output, if IDO = 1; input/output, if IDO = 2) Since, in general, the length of TABLE will not be known in advance, MAXTAB is an upper bound for this length. Empty cells are included in TABLE, and each element of TABLE is nonnegative. The cells of TABLE are sequenced so that the first variable cycles from 1 to NCLVAL(1) one time, the second variable cycles from 1 to NCLVAL(2) NCLVAL(1) times, and so on, up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) most rapidly (NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR - 1) times). That is to say, the second element of TABLE is the count for the first value for each classification variable except the last one and the second value of the last classification variable (assuming that variable takes more than one distinct value).

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 1.

IDO	Action
1	This is the first (or the only) invocation of FREQ for this data set. Initialization and updating for the data in X are performed.
2	This is an additional invocation of FREQ, and updating for the data in X is performed.

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

NCLVAR — Number of classification variables. (Input)
NCLVAR must be greater than one.
Default: NCLVAR = size (INDCL,1).

FORTRAN 90 Interface

Generic: CALL FREQ (X, INDCL, MAXTAB, MAXCL, NCLVAL, CLVAL, TABLE [, ...])
Specific: The specific interface names are S_FREQ and D_FREQ.

FORTRAN 77 Interface

Single: CALL FREQ (IDO, NOBS, NCOL, X, LDX, IFRQ, NCLVAR, INDCL, MAXTAB, MAXCL, NCLVAL, CLVAL, TABLE)
Double: The double precision name is DFREQ.

Description

The routine **FREQ** determines the distinct values in multivariate data and computes frequencies for the data. The routine accepts the data in the matrix **X**, but performs computations only for the variables (columns) in **X** specified in **INDCL**. In general, the variables for which frequencies should be computed are discrete; that is, they should take on a relatively small number of different values. Variables that are continuous can be grouped first.

The routine **OWFRQ** or **TWFRQ** can be used to group variables and determine the frequencies of groups. The routine **FREQ** fills the vector **CLVAL** with the unique values of the variables and tallies the number of unique values of each variable in the vector **NCLVAL**. Each combination of one value from each variable forms a cell in a multiway table. The frequencies of these cells are entered in **TABLE** so that the first variable cycles through its values exactly once and the last variable cycles through its values most rapidly. Some cells may not correspond to any observation in the data; that is, "missing cells" are included and have 0's in **TABLE**.

The length of the vectors **CLVAL** and **TABLE** depend on the data. The parameters **MAXCL** and **MAXTAB** are used as checks that the arrays sizes are not exceeded.

Comments

1. Workspace may be explicitly provided, if desired, by use of **F2EQ/DF2EQ**. The reference is

```
CALL F2EQ (IDO, NOBS, NCOL, X, LDX, IFRQ, NCLVAR, INDCL, MAXTAB,
   MAXCL, NCLVAL, CLVAL, TABLE, IWK, WK)
```

   The additional arguments are as follows:

   IWK — Workspace of length **NCLVAR**.

   WK — Workspace of length **NCLVAR**.

2. Informational errors

Type	Code	Description
4	1	MAXCL is too small. Increase the length of CLVAL.
4	2	MAXTAB is too small. Increase the length of TABLE.

Example

The data for this example are taken from the examples used in routine **TWFRQ**, but modified so that the values of all points within a given interval of Example 2 for **TWFRQ** are exactly equal to the class mark for that interval. The results from this example, therefore, are the same as for Example 2 for **TWFRQ**, except that **TABLE** is a vector. (The elements of the vector are sequenced as the columns of the matrix.)

```
        USE FREQ_INT
```

```
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LDX, MAXCL, MAXTAB, NCLVAR, NCOL, J
      PARAMETER  (LDX=30, MAXCL=15, MAXTAB=40, NCLVAR=2, NCOL=2)
!
      INTEGER    I, INDCL(NCLVAR), NCLVAL(NCLVAR), NOUT, &
                            NVAL1, NVAL2
      REAL       CLVAL(MAXCL), TABLE(MAXTAB), X(LDX,NCOL)
!
      DATA X/0.50, 1.50, 0.50, 1.50, 1.50, 1.50, 0.50, 1.50, 3.50, &
            2.50, 2.50, 3.50, 1.50, 2.50, 0.50, 1.50, 1.50, 0.50, &
            0.50, 0.50, 2.50, 1.50, 1.50, 1.50, 4.50, 2.50, 0.50, &
            1.50, 0.50, 2.50, &
            1.50, 3.50, 3.50, 2.50, 3.50, 4.50, 1.50, 3.50, 6.50, &
            3.50, 4.50, 6.50, 2.50, 4.50, 3.50, 2.50, 3.50, 3.50, &
            1.50, 2.50, 5.50, 2.50, 3.50, 4.50, 5.50, 4.50, 3.50, &
            2.50, 2.50, 5.50/
!
      CALL UMACH (2, NOUT)
      INDCL(1) = 1
      INDCL(2) = 2
      CALL FREQ (X, INDCL, MAXTAB, MAXCL, NCLVAL, CLVAL, TABLE)
      NVAL1 = NCLVAL(1)
      NVAL2 = NCLVAL(2)
      WRITE (NOUT,99999) (CLVAL(J),J=NVAL1+1,NVAL1+NVAL2), &
        (CLVAL(I),(TABLE((I-1)*NVAL2+J),J=1,NVAL2),I=1,NVAL1)
99999 FORMAT ('     Frequencies for All Combinations of Values', /, &
             8X,6F7.2,/,5(F7.2,6F7.0,/))
      END
```

Output

```
     Frequencies for All Combinations of Values
             1.50   2.50   3.50   4.50   5.50   6.50
0.50     3.     2.     4.     0.     0.     0.
1.50     0.     5.     5.     2.     0.     0.
2.50     0.     0.     1.     3.     2.     0.
3.50     0.     0.     0.     0.     0.     2.
4.50     0.     0.     0.     0.     1.     0.
```

UVSTA

Computes basic univariate statistics.

Required Arguments

X — |NROW| by NVAR + *m* matrix containing the data, where *m* is 0, 1, or 2 depending on whether any column(s) of X correspond to weights and/or frequencies. (Input)

STAT — 15 by NVAR matrix containing in each row statistics on all of the variables. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)

The columns of STAT correspond to the columns of X, except for the columns of X containing weights or frequencies. (The columns beyond the weights or frequencies column are shifted to the left.)

I	STAT(I, *)
1	contains means.
2	contains variances.
3	contains standard deviations.
4	contains coefficients of skewness.
5	contains coefficients of excess (kurtosis).
6	contains minima.
7	contains maxima.
8	contains ranges.
9	contains coefficients of variation, when they are defined. If the coefficient of variation is not defined for a given variable, STAT(9, *) contains a zero in the corresponding position.
10	contains numbers (counts) of nonmissing observations.
11	is used only when CONPRM is positive, and, in this case, contains the lower confidence limit for the mean (assuming normality).
12	is used only when CONPRM is positive, and, in this case, contains the upper confidence limit for the mean (assuming normality).
13	is used only when CONPRV is positive, and, in this case, contains the lower confidence limit for the variance (assuming normality).
14	is used only when CONPRV is positive, and, in this case, contains the upper confidence limit for the variance (assuming normality).
15	is used only when weighting is used (IWT is nonnegative), and, in this case, contains the sums of the weights.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of UVSTA for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to UVSTA will be made. Initialization and updating for the data in X are performed. The means are output correctly, but the other quantities output in STAT are intermediate quantities.
2	This is an intermediate invocation of UVSTA, and updating for the data in X is performed.
3	This is the final invocation of this routine. If NROW is not zero, updating is performed. The wrap-up computations for STAT are performed.

NROW — The absolute value of NROW is the number of rows of data currently input in X. (Input)
Default: NROW = size (X,1).
NROW may be positive, zero, or negative. Negative NROW means that the -NROW rows of data are to be deleted from some aspects of the analysis, and this should be done only if IDO is 2 or 3 and the wrap-up computations for STAT have not been performed. When a negative value is input for NROW, it is assumed that each of the -NROW rows of X has been input (with positive NROW) in a previous invocation of UVSTA. Use of negative values of NROW should be made with care and with the understanding that some quantities in STAT cannot be updated properly in this case. In particular, the minima, maxima, and ranges are not updated because of deletion. It is also possible that a constant variable in the remaining data will not be recognized as such.

NVAR — Number of variables (not including the weight or frequency variable, if used). (Input)
Default: NVAR = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column IWT of X contains the weights.
Default: IWT = 0.

MOPT — Missing value option. (Input)
NaN (not a number from routine AMACH(6)) is interpreted as the missing value code and any value in X equal to NaN is excluded from the computations.
Default: MOPT = 0.

MOPT	Action
0	The exclusion is listwise. (The entire row of X is excluded if any of the values of the row is equal to the missing value code.)
1	The exclusion is elementwise. (Statistics for variables with nonmissing values are updated.)

CONPRM — Confidence level for two-sided interval estimate of the means (assuming normality), in percent. (Input)
If CONPRM ≤ 0, no confidence interval for the mean is computed; otherwise, a CONPRM percent confidence interval is computed, in which case CONPRM must be between 0.0 and 100.0. CONPRM is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPRM = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPRM = .95.0.

CONPRV — Confidence level for two-sided interval estimate of the variances (assuming normality), in percent. (Input)
The confidence intervals are symmetric in probability (rather than in length). See also the description of CONPRM.
Default: CONPRV = .95.0.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
1	No printing is performed.
2	Statistics in STAT are printed if IDO = 0 or 3.
3	Intermediate means, sums of squares about the mean, minima, maxima, and counts are printed when IDO = 1 or 2, and all statistics in STAT are printed when IDO = 0 or 3.

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSTAT = size (STAT,1).

NRMISS — Number of rows of data encountered in calls to UVSTA that contain any missing values. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
Rows with a frequency of zero are not counted.

FORTRAN 90 Interface

Generic: CALL UVSTA (X, STAT [, ...])

Specific: The specific interface names are S_UVSTA and D_UVSTA.

FORTRAN 77 Interface

Single: CALL UVSTA (IDO, NROW, NVAR, X, LDX, IFRQ, IWT, MOPT, CONPRM, CONPRV, IPRINT, STAT, LDSTAT, NRMISS)

Double: The double precision name is DUVSTA.

Description

For the data in each column of X, except the columns containing frequencies or weights, UVSTA computes the sample mean, variance, minimum, maximum, and other basic statistics. It also computes confidence intervals for the mean and variance if the sample is assumed to be from a normal population.

Missing values, that is, values equal to NaN (not a number, the value returned by routine AMACH(6)), are excluded from the computations. If MOPT is positive, the exclusion is listwise; that is, the entire observation is excluded and no computations are performed even for the variables with valid values. If frequencies or weights are specified, any observation whose frequency or weight is missing is excluded from the computations.

Frequencies are interpreted as multiple occurrences of the other values in the observations. That is, a row of X with a frequency variable having a value of 2 has the same effect as two rows with frequencies of 1. The total of the frequencies is used in computing all of the statistics based on moments (mean, variance, skewness, and kurtosis). Weights are not viewed as replication factors. The sum of the weights is used only in computing the mean (of course, then the weighted mean is used in computing the central moments). Both weights and frequencies can be zero, but neither can be negative. In general, a zero frequency means that the row is to be eliminated from the analysis; no further processing, counting of missing values, or error checking is done on the row. Although it is not required that frequencies be integers, the logic of their treatment implicitly assumes that they are. Weights, on the other hand, are allowed to be continuous. A weight of zero results in the row being counted, and updates are made of statistics and of the number of missing values. A missing value for the frequency or a missing value for the weight when the frequency is nonzero results in the row being deleted from the analysis; but even in that case, if one is nonmissing, it is an error for that nonmissing weight or frequency to be negative.

The definitions of some of the statistics are given below in terms of a single variable x. The i-th datum is x_i, with corresponding frequency f_i and weight w_i. If either frequencies or weights are not specified, f_i and/or w_i are identically one. The summation in each case is over the set of valid observations, based on the setting of MOPT and the presence of missing values in the data.

Number of nonmissing observations, STAT(10, *)

$$n = \sum f_i$$

Mean, STAT(1, *)

$$\bar{x}_w = \frac{\sum f_i w_i x_i}{\sum f_i w_i}$$

Variance, STAT(2, *)

$$s_w^2 = \frac{\sum f_i w_i \left(x_i - \bar{x}_w \right)^2}{n - 1}$$

Skewness, STAT(4, *)

$$\frac{\sum f_i w_i \left(x_i - \bar{x}_w \right)^3 / n}{\left[\sum f_i w_i \left(x_i - \bar{x}_w \right)^2 / n \right]^{3/2}}$$

Excess or Kurtosis, STAT(5, *)

$$\frac{\sum f_i w_i \left(x_i - \bar{x}_w \right)^4 / n}{\left[\sum f_i w_i \left(x_i - \bar{x}_w \right)^2 / n \right]^2} - 3$$

Minimum, STAT(6, $*$)

$$x_{\min} = \min\left(x_i\right)$$

Maximum, STAT(7, $*$)

$$x_{\max} = \max\left(x_i\right)$$

Range, STAT(8, $*$)

$$x_{\max} - x_{\min}$$

Coefficient of Variation, STAT(9, $*$)

$$\frac{s_w}{\overline{x}_w} \quad \text{for} \quad \overline{x}_w \neq 0$$

The arguments **IDO** and **NROW** allow data to be input a few at a time and even to be deleted after having been included in the analysis. The minima, maxima, and ranges are not updated when observations are deleted.

Comments

Workspace may be explicitly provided, if desired, by use of **U2STA**/**DU2STA**. The reference is

```
CALL U2STA (IDO, NROW, NVAR, X, LDX, IFRQ, IWT, MOPT, CONPRM, CON-
   PRV, IPRINT, STAT, LDSTAT, NRMISS, WK)
```

The additional argument is

WK — Real work vector of length specified above. **WK** should not be changed between calls to **U2STA**.

Examples

Example 1

This example uses data from Draper and Smith (1981). There are 5 variables and 13 observations.

```
      USE UVSTA_INT
      USE GDATA_INT

      IMPLICIT   NONE
```

```
      INTEGER    LDSTAT, LDX, NVAR
      PARAMETER  (LDSTAT=15, LDX=13, NVAR=5)
!
      INTEGER    IPRINT, NR, NROW, NV
      REAL       CONPRM, CONPRV, STAT(LDSTAT,NVAR), X(LDX,NVAR)
!                                Get data for example.
      CALL GDATA (5, X, NR, NV)
!                                All data are input at once.
      NROW = NR
!                                No unequal frequencies or weights
!                                are used.
!                                Get 95% confidence limits.
!                                Delete any row containing a missing
!                                value.
!                                Print results.
      IPRINT = 1
      CALL UVSTA (X, STAT, NROW=NROW, IPRINT=IPRINT)
      END
```

Output

```
 Univariate Statistics from UVSTA

 Variable      Mean       Variance    Std. Dev.       Skewness      Kurtosis
    1        7.4615        34.6026      5.8824        0.68768       0.07472
    2       48.1538       242.1410     15.5609       -0.04726      -1.32257
    3       11.7692        41.0256      6.4051        0.61064      -1.07916
    4       30.0000       280.1667     16.7382        0.32960      -1.01406
    5       95.4231       226.3136     15.0437       -0.19486      -1.34244

 Variable    Minimum       Maximum        Range       Coef. Var.       Count
    1         1.0000       21.0000      20.0000          0.7884     13.0000
    2        26.0000       71.0000      45.0000          0.3231     13.0000
    3         4.0000       23.0000      19.0000          0.5442     13.0000
    4         6.0000       60.0000      54.0000          0.5579     13.0000
    5        72.5000      115.9000      43.4000          0.1577     13.0000

 Variable  Lower CLM     Upper CLM    Lower CLV      Upper CLV
    1         3.9068       11.0162      17.7930        94.2894
    2        38.7505       57.5572     124.5113       659.8163
    3         7.8987       15.6398      21.0958       111.7918
    4        19.8852       40.1148     144.0645       763.4335
    5        86.3322      104.5139     116.3726       616.6877
```

Example 2

In this example, we use some simple data to illustrate the use of frequencies, missing values, and the parameters `IDO` and `NROW`. In the data below, "NaN" represents a missing value.

f	***x***	***y***
2	3.0	5.0
1	9.0	2.0
3	1.0	NaN

We bring in the data one observation at a time in this example. Also, we bring in one false datum and then delete it on a subsequent call to UVSTA.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDSTAT, NVAR
      PARAMETER  (LDSTAT=15, NVAR=2)
!
      INTEGER    IDO, IFRQ, IPRINT, MOPT, NRMISS, NROW
      REAL       STAT(LDSTAT,NVAR), X1(1,NVAR+1)
!                                All data are input one observation
!                                at a time in the vector X1.
      NROW = 1
!                                Frequencies are in the first
!                                position.  No weights are used.
      IFRQ = 1
!                                Get 95% confidence limits.
!                                Elementwise deletion of missing
!                                values.
      MOPT = 1
!                                Print results, intermediate as well.
      IPRINT = 2
!                                Bring in the first observation.
      IDO   = 1
      X1(1,1) = 2.0
      X1(1,2) = 3.0
      X1(1,3) = 5.0
      CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, &
                  IPRINT=IPRINT, NRMISS=NRMISS)
!                                Bring in the second observation.
      IDO   = 2
      X1(1,1) = 1.0
      X1(1,2) = 9.0
      X1(1,3) = 2.0
      CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, &
                  IPRINT=IPRINT, NRMISS=NRMISS)
!                                Bring in a false observation.
      X1(1,1) = 3.0
      X1(1,2) = 6.0
      X1(1,3) = 3.0
      CALL UVSTA (X1, STAT, IDO=IDO, NVAR=NVAR, IFRQ=IFRQ, MOPT=MOPT, &
                  IPRINT=IPRINT, NRMISS=NRMISS)
!                                Delete the false observation.
!                                This may make the mimina, maxima,
!                                and range incorrect.
      NROW  = -1
      X1(1,1) = 3.0
      X1(1,2) = 6.0
      X1(1,3) = 3.0
      CALL UVSTA (X1, STAT, IDO=IDO, NROW=NROW, NVAR=NVAR, IFRQ=IFRQ, &
                  MOPT=MOPT, IPRINT=IPRINT, NRMISS=NRMISS)
      NROW = 1
!                                Bring in the final observation.
      IDO   = 3
      X1(1,1) = 3.0
      X1(1,2) = 1.0
      X1(1,3) = AMACH(6)
      CALL UVSTA (X1, STAT, IDO=IDO, NROW=NROW, NVAR=NVAR, IFRQ=IFRQ, &
                  MOPT=MOPT, IPRINT=IPRINT, NRMISS=NRMISS)
```

```
END
```

Output

```
                    Intermediate Statistics from UVSTA
 Variable          Mean       Sum Sqs.        Minimum        Maximum        Count
    1            3.0000         0.0000         3.0000         3.0000       2.0000
    2            5.0000         0.0000         5.0000         5.0000       2.0000

                    Intermediate Statistics from UVSTA
 Variable          Mean       Sum Sqs.        Minimum        Maximum        Count
    1            5.0000        24.0000         3.0000         9.0000       3.0000
    2            4.0000         6.0000         2.0000         5.0000       3.0000

                    Intermediate Statistics from UVSTA
 Variable          Mean       Sum Sqs.        Minimum        Maximum        Count
    1            5.5000        25.5000         3.0000         9.0000       6.0000
    2            3.5000         7.5000         2.0000         5.0000       6.0000

                    Intermediate Statistics from UVSTA
 Variable          Mean       Sum Sqs.        Minimum        Maximum        Count
    1            5.0000        24.0000         3.0000         9.0000       3.0000
    2            4.0000         6.0000         2.0000         5.0000       3.0000

                     Univariate Statistics from UVSTA
 Variable          Mean       Variance     Std. Dev.       Skewness     Kurtosis
    1            3.0000         9.6000         3.0984         1.4142       0.5000
    2            4.0000         3.0000         1.7321        -0.7071      -1.5000

 Variable       Minimum        Maximum          Range     Coef. Var.        Count
    1            1.0000         9.0000         8.0000         1.0328       6.0000
    2            2.0000         5.0000         3.0000         0.4330       3.0000

 Variable     Lower CLM      Upper CLM      Lower CLV      Upper CLV
     1          -0.2516         6.2516         3.7405        57.7470
     2          -0.3027         8.3027         0.8133       118.4935
```

RANKS

Computes the ranks, normal scores, or exponential scores for a vector of observations.

Required Arguments

X — Vector of length NOBS containing the observations to be ranked. (Input)

SCORE — Vector of length NOBS containing the rank or a transformation of that rank of each observation. (Output)
X and SCORE may occupy the same memory.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

FUZZ — Value used to determine ties. (Input)
If |X(I) - X(J)| is less than or equal to FUZZ, then X(I) and X(J) are said to be tied.
Default: FUZZ = 0.0.

ITIE — Option for determining the method used to assign a score to tied observations. (Input)
Default: ITIE = 0.

ITIE	Method
0	The average of the scores of the tied observations is used.
1	The highest score in the group of ties is used.
2	The lowest score in the group of ties is used.
3	The tied observations are to be randomly untied using an IMSL random number generator.

ISCORE — Option for specifying the type of values returned in SCORE. (Input)
Default: ISCORE = 0.

ISCORE	Type
0	Ranks
1	Blom version of normal scores
2	Tukey version of normal scores

ISCORE	Type
3	Van der Waerdan version of normal scores
4	Expected value of normal order statistics (For tied observations, the average of the expected normal scores are used.)
5	Savage scores (the expected value of exponential order statistics)

FORTRAN 90 Interface

Generic: CALL RANKS (X, SCORE [, ...])

Specific: The specific interface names are S_RANKS and D_RANKS.

FORTRAN 77 Interface

Single: CALL RANKS (NOBS, X, FUZZ, ITIE, ISCORE, SCORE)

Double: The double precision name is DRANKS.

Description

The routine RANKS determines the ranks, or various transformations of the ranks of the data in X. Ties in the data can be resolved in four different ways, as specified in ITIE.

ISCORE = 0: Ranks

For this option, the values output in SCORE are the ordinary ranks of the data in X. If X(I) has the smallest value among those in X and there is no other element in X with this value, then SCORE(I) = 1. If both X(I) and X(J) have the same smallest value, then

if ITIE = 0, SCORE(I) = SCORE(J) = 1.5

if ITIE = 1, SCORE(I) = SCORE(J) = 2.0

if ITIE = 2, SCORE(I) = SCORE(J) = 1.0

if ITIE = 3, SCORE(I) = 1.0 and SCORE(J) = 2.0
or SCORE(I) = 2.0 and SCORE(J) = 1.0.

When the ties are resolved by use of routine **RNUNF** (see Chapter 18, "Random Number Generation") to generate random numbers, different results may occur when running the same program at different times unless the "seed" of the random number generator is set explicitly by use of the routine **RNSET** (see Chapter 18, "Random Number Generation"). Ordinarily, there is no need to call the routine to set the seed, even if there are ties in the data.

ISCORE = 1: Normal Scores, Blom Version

Normal scores are expected values, or approximations to the expected values, of order statistics from a normal distribution. The simplest approximations are obtained by evaluating the inverse cumulative normal distribution function (routine **ANORIN**, see Chapter 18, "Random Number Generation") at the ranks scaled into the open interval (0, 1). In the Blom version (see Blom 1958), the scaling transformation for the rank $r_i (1 \le r_i \le n$, where n is the sample size, **NOBS)** is $(r_i$ **-** 3/8)/(n + 1/4). The Blom normal score corresponding to the observation with rank r_i is

$$\Phi^{-1}\left(\frac{r_i - 3/8}{n + 1/4}\right)$$

where **Φ**( ) is the normal cumulative distribution function.

Adjustments for ties are made after the normal score transformation. That is, if **X**(**I**) equals **X**(**J**) (within **FUZZ**) and their value is the k-th smallest in the data set, the Blom normal scores are determined for ranks of k and k + 1, and then these normal scores are averaged or selected in the manner specified by **ITIE**. (Whether the transformations are made first or ties are resolved first makes no difference except when averaging is done.)

ISCORE = 2: Normal Scores, Tukey Version

In the Tukey version (see Tukey 1962), the scaling transformation for the rank r_i is $(r_i$ **-** 1/3)/(n + 1/3). The Tukey normal score corresponding to the observation with rank r_i is

$$\Phi^{-1}\left(\frac{r_i - 1/3}{n + 1/3}\right)$$

Ties are handled in the same way as discussed above for the Blom normal scores.

ISCORE = 3: Normal Scores, Van der Waerden Version

In the Van der Waerden version (see Lehmann 1975, page 97), the scaling transformation for the rank r_i is r_i / (n + 1). The Van der Waerden normal score corresponding to the observation with rank r_i is

$$\Phi^{-1}\left(\frac{r_i}{n+1}\right)$$

Ties are handled in the same way as discussed above for the Blom normal scores.

ISCORE = 4: Expected Value of Normal Order Statistics

For this option, the values output in **SCORE** are the expected values of the normal order statistics from a sample of size **NOBS**. If the value in **X**(**I**) is the *k*-th smallest, then the value output in **SCORE**(**I**) is E(Z_k), where E( ) is the expectation operator and Z_k is the *k*-th order statistic in a sample of size **NOBS** from a standard normal distribution. Such expected values are computed by the routine **ENOS** (see Chapter 20, "Mathematical Support"). Ties are handled in the same way as discussed above for the Blom normal scores.

ISCORE = 5: Savage Scores

For this option, the values output in **SCORE** are the expected values of the exponential order statistics from a sample of size **NOBS**. These values are called Savage scores because of their use in a test discussed by Savage (1956) (see Lehman 1975). If the value in **X**(**I**) is the *k*-th smallest, then the value output in **SCORE**(**I**) is E(Y_k), where Y_k is the *k*-th order statistic in a sample of size **NOBS** from a standard exponential distribution. The expected value of the *k*-th order statistic from an exponential sample of size *n* (**NOBS**) is

$$\frac{1}{n}+\frac{1}{n-1}+\cdots+\frac{1}{n-k+1}$$

Ties are handled in the same way as discussed above for the Blom normal scores.

The example uses all of these options with the same data set, which contains some ties. The ties are handled different ways in this example.

Comments

1. Workspace may be explicitly provided, if desired, by use **R2NKS/DR2NKS**. The reference is:

   ```
   CALL R2NKS (NOBS, X, FUZZ, ITIE, ISCORE, SCORE, IWK)
   ```

   The additional argument is:

   IWK — Integer work vector of length **NOBS**.

2. The routine **RNSET** (see Chapter 18, "Random Number Generation") can be used to initialize the seed of the random number generator used to break ties. If the seed is not initialized by **RNSET**; different runs of the same program can yield different results if there are tied observations and **ITIE** = 3.

Example

The data for this example, from Hinkley (1977), are the same used in several examples in this chapter. There are 30 observations. Note that the fourth and sixth observations are tied and that the third and twentieth are tied.

```
      USE RANKS_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=30)
!
      INTEGER    ISCORE, ISEED, ITIE, NOUT
      REAL       FUZZ, SCORE(NOBS), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37,&
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!
      CALL UMACH (2, NOUT)
!                                Ranks.
      ISCORE = 0
!                                Average ties.
      ITIE = 0
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
      WRITE (NOUT,99994) SCORE
99994 FORMAT ('   Ranks', /, (1X,10F7.1))
!                                Blom normal scores.
      ISCORE = 1
!                                Take largest ranks for ties.
      ITIE = 1
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
      WRITE (NOUT,99995) SCORE
99995 FORMAT (/, '   Blom normal scores', /, (1X,10F7.3))
!                                Tukey normal scores.
      ISCORE = 2
!                                Take smallest ranks for ties.
      ITIE = 2
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
      WRITE (NOUT,99996) SCORE
99996 FORMAT (/, '   Tukey normal scores', /, (1X,10F7.3))
!                                Van der Waerden scores.
      ISCORE = 3
!                                Randomly resolve ties.
      ISEED = 123457
      CALL RNSET (ISEED)
      ITIE = 3
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
```

```
      WRITE (NOUT,99997) SCORE
99997 FORMAT (/, '   Van der Waerden scores', /, (1X,10F7.3))
!                                  Expected value of normal O. S.
      ISCORE = 4
!                                  Average ties.
      ITIE = 0
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
      WRITE (NOUT,99998) SCORE
99998 FORMAT (/, '   Expected values of normal order statistics', /,&
            (1X,10F7.3))
!                                  Savage scores.
      ISCORE = 5
!                                  Average ties.
      ITIE = 0
      FUZZ = 0.0
!
      CALL RANKS (X, SCORE, ISCORE=ISCORE, ITIE=ITIE, FUZZ=FUZZ)
      WRITE (NOUT,99999) SCORE
99999 FORMAT (/, '   Expected values of exponential order statistics', &
           /, (1X,10F7.2))
      END
```

Output

```
 Ranks
   5.0   18.0    6.5   11.5   21.0   11.5    2.0   15.0   29.0   24.0
  27.0   28.0   16.0   23.0    3.0   17.0   13.0    1.0    4.0    6.5
  26.0   19.0   10.0   14.0   30.0   25.0    9.0   20.0    8.0   22.0

 Blom normal scores
-1.024  0.209 -0.776 -0.294  0.473 -0.294 -1.610 -0.041  1.610  0.776
 1.176  1.361  0.041  0.668 -1.361  0.125 -0.209 -2.040 -1.176 -0.776
 1.024  0.294 -0.473 -0.125  2.040  0.893 -0.568  0.382 -0.668  0.568

 Tukey normal scores
-1.020  0.208 -0.890 -0.381  0.471 -0.381 -1.599 -0.041  1.599  0.773
 1.171  1.354  0.041  0.666 -1.354  0.124 -0.208 -2.015 -1.171 -0.890
 1.020  0.293 -0.471 -0.124  2.015  0.890 -0.566  0.381 -0.666  0.566

  Van der Waerden scores
-0.989  0.204 -0.753 -0.287  0.460 -0.372 -1.518 -0.040  1.518  0.753
 1.131  1.300  0.040  0.649 -1.300  0.122 -0.204 -1.849 -1.131 -0.865
 0.989  0.287 -0.460 -0.122  1.849  0.865 -0.552  0.372 -0.649  0.552
 Expected values of normal order statistics
-1.026  0.209 -0.836 -0.338  0.473 -0.338 -1.616 -0.041  1.616  0.777
 1.179  1.365  0.041  0.669 -1.365  0.125 -0.209 -2.043 -1.179 -0.836
 1.026  0.294 -0.473 -0.125  2.043  0.894 -0.568  0.382 -0.669  0.568

 Expected values of exponential order statistics
 0 18   0.89   0.24   0.47   1.17   0.47   0.07   0.68   2.99   1.54
 2.16   2.49   0.74   1.40   0.10   0.81   0.56   0.03   0.14   0.24
 1.91   0.98   0.40   0.61   3.99   1.71   0.35   1.07   0.30   1.28
```

LETTR

Produces a letter value summary.

Required Arguments

X — Vector of length **NOBS** containing the data. (Input)

SUMRY — Vector of length **NUM** containing the summary letter values. (Output)
If **NUM** is 5, for example, **SUMRY** contains the minimum, the lower hinge (quartile), the median, the upper hinge, and the maximum, in that order.

NMISS — Number of missing values. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**X**,1).

NUM — Number of summary values. (Input)
NUM must be an odd integer greater than or equal to 3. A common value for **NUM** is 5.
Default: **NUM** = 5.

FORTRAN 90 Interface

Generic: `CALL LETTR (X, SUMRY, NMISS [, ...])`
Specific: The specific interface names are **S_LETTR** and **D_LETTR**.

FORTRAN 77 Interface

Single: `CALL LETTR (NOBS, X, NUM, SUMRY, NMISS)`
Double: The double precision name is **DLETTR**.

Description

The routine **LETTR** computes the median ("*M*"), the minimum, the maximum, and other depths or "letter values"—hinges ("*H*"), eighths ("*E*"), sixteenths ("*D*"), etc.—as specified by **NUM**. If **NUM** = 9, for example, the values in **SUMRY** correspond to min, *D*, *E*, *H*, *M*, *H*, *E*, *D*, and max, in that order. The use of letter values in summarizing a set of data is due to Tukey. Examples and discussion of the use of letter values are given by Tukey (1977, Chapter 2) and by Velleman and Hoaglin (1981, Chapter 2).

Comments

1. Workspace may be explicitly provided, if desired, by use of **L2TTR**/**DL2TTR**. The reference is:

   ```
   CALL L2TTR (NOBS, X, NUM, SUMRY, NMISS, WK)
   ```

   The additional argument is:

   WK — Work vector of length **NOBS**.

2. Informational errors

Type	Code	Description
3	3	The results are likely not to be meaningful if NUM is larger than the number of valid observations, (NOBS - NMISS).
4	4	The number of valid observations (NOBS - NMISS) is not greater than zero.

Example

In this example, **LETTR** is used to compute a letter value summary of the measurements (in inches) of precipitation in Minneapolis/St. Paul during the month of March for 30 consecutive years. These data were studied by Hinkley (1977) and by Velleman and Hoaglin (1981), pages 50 - 53.

```
      USE LETTR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    I, NMISS, NOBS, NOUT, NUM
      REAL       SUMRY(11), X(30)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37,&
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!
      CALL UMACH (2, NOUT)
      NOBS = 30
      NUM  = 11
!
```

```
      CALL LETTR (X, SUMRY, NMISS, NUM=NUM)
      WRITE (NOUT,99998) SUMRY(6), (SUMRY(6-I),SUMRY(6+I),I=1,5)
99998 FORMAT ('         Letter Values', /, '       Lower      Upper', &
            /, '  M         ', F6.3, /, '  H  ', F6.3, 6X, F6.3, /, &
            '  E  ', F6.3, 6X, F6.3, /, '  D  ', F6.3, 6X, F6.3, /, &
            '  !  ', F6.3, 6X, F6.3, /, ' m/M ', F6.3, 6X, F6.3)
      WRITE (NOUT,99999) NMISS
99999 FORMAT ('  There are ', I2, ' missing values.')
      END
```

Output

```
        Letter Values
         Lower       Upper
 M           1.470
 H    0.900       2.100
 E    0.680       2.905
 D    0.495       3.230
 !    0.395       4.060
m/M   0.320       4.750
There are  0 missing values.
```

ORDST

Determines order statistics.

Required Arguments

X — Vector of length NOBS containing the data. (Input)

NOS — Number of order statistics. (Input)
NOS must be greater than or equal to one and less than or equal to NOBS.

OS — Vector of length NOS containing the order statistics. (Output)

NMISS — Number of missing values. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than or equal to one.
Default: NOBS = size (X,1).

IOPT — Option to choose the order statistics to be calculated. (Input)
Default: IOPT = 1.

IOPT	Action
0	Calculate the NOS order statistics listed in IOS.
1	Calculate the first NOS order statistics.
2	Calculate the last NOS order statistics.

IOS — If IOPT = 0, IOS is a vector of length NOS containing the ranks of the order statistics. (Input)
The elements of IOS must be greater than or equal to one and less than or equal to NOBS. If IOPT = 1 or 2, IOS is unreferenced and can be defined as a vector of length 1.

FORTRAN 90 Interface

Generic: CALL ORDST (X, NOS, OS, NMISS [, ...])
Specific: The specific interface names are S_ORDST and D_ORDST.

FORTRAN 77 Interface

Single: CALL ORDST (NOBS, X, NOS, IOPT, IOS, OS, NMISS)

Double: The double precision name is DORDST.

Description

The routine ORDST determines order statistics from the data in X and returns them in the vector OS. The routine ORDST first checks to see if X is sorted, in which case the order statistics are merely picked from X. If X is not sorted, ORDST does either a complete or partial sort, depending on how many order statistics are requested. Since either the largest few order statistics or the smallest few are often of interest, the option parameter IOPT allows the user to obtain the largest or the smallest order statistics easily; otherwise (when IOPT is set to 0), the user specifies in the vector IOS exactly which order statistics are to be returned. If IOS is used, the order statistics returned in OS are in the same order as the indicators in IOS.

Comments

1. Workspace may be explicitly provided, if desired, by use of O2DST/DO2DST. The reference is:

   ```
   CALL O2DST (NOBS, X, NOS, IOPT, IOS, OS, NMISS, WK)
   ```

   The additional argument is as follows:

   WK — Work vector of length NOBS.

2. Informational errors

Type	Code	Description
3	1	All of the observations are missing values. The elements of OS have been set to NaN (not a number).
3	2	NOS order statistics have been requested, but there are only NOBS - NMISS valid observations. Order statistics greater than NOBS - NMISS have been set to NaN (not a number).
3	3	Each value of IOS must be greater than 0 and less than or equal to the number of valid observations. The values of OS that are not defined have been set to NaN.

3. Missing values (NaN) are excluded from the analysis. Order statistics are based on the NOBS — NMISS nonmissing elements of X.

Examples

Example 1

The data for these examples are from Hinkley (1977) and Velleman and Hoaglin (1981). They are the measurements (in inches) of precipitation in Minneapolis/St. Paul during the month of March for 30 consecutive years. In the first example, the first five order statistics from a sample of size 30 are obtained. Since **IOPT** is set to 1, **IOS** is not used.

```
      USE ORDST_INT
      USE UMACH_INT
      USE WRRRN_INT
      USE AMACH_INT

      IMPLICIT    NONE
      INTEGER     NOBS, NOS
      PARAMETER  (NOBS=30, NOS=5)
!
      INTEGER    NMISS, NOUT
      REAL       OS(NOS), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!
      CALL UMACH (2, NOUT)
      CALL ORDST (X, NOS, OS, NMISS)
      CALL WRRRN ('First five order statistics:', OS, 1, NOS, 1)
      WRITE (NOUT,99999) NMISS
99999 FORMAT ('   There are', I2, ' missing values.')
      END
```

Output

```
 First five order statistics:
      1       2       3       4       5
0.3200  0.4700  0.5200  0.5900  0.7700
There are 0 missing values.
```

Example 2

In the second example, the last five order statistics from a sample of size 30 are obtained. This example uses the same data as in the first example, but this time the first two observations have been set to a missing value indicator (**AMACH**(6)). Note that since there are two missing values in the data set, the indices of the last five order statistics are numbers 24, 25, 26, 27, and 28. In this example, **NMISS** will be returned with a value of 2. The index of the last order statistic can be determined by **NOBS - NMISS**.

```
      USE ORDST_INT
      USE UMACH_INT
      USE WRRRN_INT
```

```
      USE AMACH_INT

      IMPLICIT   NONE
      INTEGER    IOPT, NOBS, NOS
      PARAMETER  (IOPT=2, NOBS=30, NOS=5)
!
      INTEGER    NMISS, NOUT
      REAL       OS(NOS), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!
      CALL UMACH (2, NOUT)
      X(1) = AMACH(6)
      X(2) = AMACH(6)
      CALL ORDST (X, NOS, OS, NMISS, IOPT=IOPT)
      CALL WRRRN ('Last five order statistics:', OS, 1, NOS, 1)
      WRITE (NOUT,99999) NMISS
99999 FORMAT ('   There are', I2, ' missing values.')
      END
```

Output

```
 Last five order statistics:
     1       2       3       4       5
2.810   3.000   3.090   3.370   4.750
There are 2 missing values.
```

Example 3

In this example, we illustrate the use of `IOS` to specify exactly which order statistics are to be computed. We request what would be the last five order statistics from a sample of size 30, that is, order statistics 26, 27, 28, 29, and 30. As in example two, the data set has two missing values. Order statistics 29 and 30 are not defined, but since they are specifically requested, a warning message is issued and `OS` contains two missing values on return.

```
      USE ORDST_INT
      USE UMACH_INT
      USE WRRRN_INT
      USE AMACH_INT

      IMPLICIT   NONE
      INTEGER    IOPT, NOBS, NOS
      PARAMETER  (IOPT=0, NOBS=30, NOS=5)
!
      INTEGER    IOS(NOS), NMISS, NOUT
      REAL       OS(NOS), X(NOBS)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA IOS/26, 27, 28, 29, 30/
!
      CALL UMACH (2, NOUT)
```

```
      X(1) = AMACH(6)
      X(2) = AMACH(6)
      CALL ORDST (X, NOS, OS, NMISS, IOS=IOS, IOPT=IOPT)
      CALL WRRRN ('Last five order statistics:', OS, 1, NOS, 1)
      WRITE (NOUT,99999) NMISS
99999 FORMAT ('   There are', I2, ' missing values.')
      END
```

Output

```
 *** WARNING  ERROR 3 from ORDST.  Each value of IOS must be greater than 0
 ***          and less than or equal to the number of valid observations,
 ***          NOBS-NMISS, which is 28.  IOS contains 2 values outside of
 ***          this range. The corresponding values of OS have been set to
 ***          NaN (not a number).

 Last five order statistics:
     1       2       3       4       5
 3.090   3.370   4.750     NaN     NaN
 There are 2 missing values.
```

EQTIL

Computes empirical quantiles.

Required Arguments

X — Vector of length **NOBS** containing the data. (Input)

NQPROP — Number of quantiles. (Input)
NQPROP must be greater than or equal to one.

QPROP — Vector of length **NQPROP** containing the quantile proportions. (Input)
The elements of **QPROP** must lie in the interval (0, 1).

Q — Vector of length **NQPROP** containing the empirical quantiles. (Output)
Q(*i*) corresponds to the empirical quantile at proportion **QPROP**(*i*). The quantiles are determined by linear interpolation between adjacent ordered sample values.

XLO — Vector of length **NQPROP** containing the largest element of **X** less than or equal to the desired quantile. (Output)

XHI — Vector of length **NQPROP** containing the smallest element of **X** greater than or equal to the desired quantile. (Output)

NMISS — Number of missing values. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than or equal to one.
Default: **NOBS** = size (**X**,1).

FORTRAN 90 Interface

Generic: `CALL EQTIL (X, NQPROP, QPROP, Q, XLO, XHI, NMISS [, ...])`
Specific: The specific interface names are `S_EQTIL` and `D_EQTIL`.

FORTRAN 77 Interface

Single: `CALL EQTIL (NOBS, X, NQPROP, QPROP, Q, XLO, XHI, NMISS)`

Double: The double precision name is **DEQTIL**.

Description

The routine **EQTIL** determines the empirical quantiles, as indicated in the vector **QPROP**, from the data in **X**. The routine **EQTIL** first checks to see if **X** is sorted; if **X** is not sorted, the routine does either a complete or partial sort, depending on how many order statistics are required to compute the quantiles requested.

The routine **EQTIL** returns the empirical quantiles and, for each quantile, the two order statistics from the sample that are at least as large and at least as small as the quantile. For a sample of size n, the quantile corresponding to the proportion p is defined as

$$Q(p) = (1 - f)x_j + f x_{j+1}$$

where $j = \lfloor p(n+1) \rfloor$, $f = p(n+1) - j$, and x_j is the j-th order statistic, if $1 \leq j < n$; otherwise, the empirical quantile is the smallest or largest order statistic.

Comments

1. Workspace may be explicitly provided, if desired, by use of **E2TIL/DE2TIL**. The reference is:

   ```
   CALL E2TIL (NOBS, X, NQPROP, QPROP, Q, XLO, XHI, NMISS, WK)
   ```

   The additional argument is:

   WK — Workspace of length **NOBS** containing the sorted data. (Output)

   If **X** is sorted in ascending order with all missing values at the end of **X**, then **X** and **WK** may share the same storage location.

2. Informational error

Type	Code	Description
3	1	All of the observations are missing values. The elements of Q, XLO, and XHI have been set to NaN (not a number).

3. Missing values (NaN) are excluded from the analysis. Empirical quantiles are based on the **NOBS - NMISS** nonmissing elements of **X**.

Example

In this example, five empirical quantiles from a sample of size 30 are obtained. Notice that the 0.5 quantile corresponds to the sample median. The data are from Hinkley (1977) and Velleman and Hoaglin (1981). They are the measurements (in inches) of precipitation in Minneapolis/St. Paul during the month of March for 30 consecutive years.

```
      USE EQTIL_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS, NQPROP
      PARAMETER  (NOBS=30, NQPROP=5)
!
      INTEGER    I, NMISS, NOUT
      REAL       QPROP(NQPROP), X(NOBS), XEMP(NQPROP), XHI(NQPROP),&
                 XLO(NQPROP)
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
      DATA QPROP/0.01, 0.50, 0.90, 0.95, 0.99/
!
      CALL UMACH (2, NOUT)
      CALL EQTIL (X, NQPROP, QPROP, XEMP, XLO, XHI, NMISS)
      WRITE (NOUT,99997)
99997 FORMAT ('              Smaller     Empirical     Larger', /, &
            '  Quantile     Datum      Quantile      Datum')
      DO 10  I=1, NQPROP
         WRITE (NOUT,99998) QPROP(I), XLO(I), XEMP(I), XHI(I)
   10 CONTINUE
99998 FORMAT (4X, F4.2, 8X, F4.2, 8X, F4.2, 8X, F4.2)
      WRITE (NOUT,99999) NMISS
99999 FORMAT (/, ' There are ', I2, ' missing values.')
      END
```

Output

```
              Smaller     Empirical     Larger
Quantile        Datum      Quantile      Datum
  0.01           0.32        0.32         0.32
  0.50           1.43        1.47         1.51
  0.90           3.00        3.08         3.09
  0.95           3.37        3.99         4.75
  0.99           4.75        4.75         4.75

There are  0 missing values.
```

TWOMV

Computes statistics for mean and variance inferences using samples from two normal populations.

Required Arguments

X — Vector of length **NROWX** containing observations from the first sample. (Input)

Y — Vector of length **NROWY** containing observations from the second sample. (Input)

STAT — Vector of length 25 containing the statistics.
(Output, if **IDO** = 0 or 1; input/output, if **IDO** = 2 or 3.) These are:

I	**STAT(I)**
1	Mean of the first sample
2	Mean of the second sample
3	Variance of the first sample
4	Variance of the second sample
5	Number of observations in the first sample
6	Number of observations in the second sample

(**STAT**(7) through **STAT**(14) depend on the assumption of equal variances.)

I	**STAT(I)**
7	Pooled variance
8	*t* value, assuming equal variances
9	Probability of a larger *t* in absolute value, assuming normality, equal means, and equal variance
10	Degrees of freedom assuming equal variances
11	Lower confidence limit for the mean of the first population minus the mean of the second, assuming equal variances
12	Upper confidence limit for the mean of the first population minus the mean of the second, assuming equal variances
13	Lower confidence limit for the common variance
14	Upper confidence limit for the common variance

(**STAT**(15) through **STAT**(19) use approximations that do not depend on an assumption of equal variances.)

I	STAT(I)
15	*t* value, assuming unequal variances.
16	Approximate probability of a larger *t* in absolute value, assuming normality, equal means, and unequal variances
17	Degrees of freedom assuming unequal variances, for Satterthwaite's approximation
18	Approximate lower confidence limit for the mean of the first population minus the mean of the second, assuming equal variances
19	Approximate upper confidence limit for the mean of the first population minus the mean of the second, assuming equal variances
20	*F* value (greater than or equal to 1.0)
21	Probability of a larger *F* in absolute value, assuming normality and equal variances
22	Lower confidence limit for the ratio of the variance of the first population to the second
23	Upper confidence limit for the ratio of the variance of the first population to the second
24	Number of missing values of first sample
25	Number of missing values of second sample

Optional Arguments

IDO — Processing option. (Input)

Default: `IDO` = 0.

IDO	Action
0	This is the only invocation of `TWOMV` for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to `TWOMV` will be made. Initialization and updating are performed. The means are output correctly, but most of the other quantities output in `STAT` are intermediate quantities.
2	This is an intermediate invocation of `TWOMV`, and updating for the data in `X` and `Y` is performed.
3	This is the final invocation of this routine. Updating for the data in `X` and `Y` and wrap-up computations are performed.

NROWX — Absolute value of NROWX is the number of observations currently input in X. (Input)
Default: NROWX = size (X,1).
NROWX may be positive, zero, or negative. Negative NROWX means delete the -NROWX observations in X from the analysis.

NROWY — Absolute value of NROWY is the number of observations currently input in Y. (Input)
Default: NROWY = size (Y,1).
NROWY may be positive, zero, or negative. Negative NROWY means delete the -NROWY observations in Y from the analysis.

CONPRM — Confidence level for two-sided interval estimate of the mean of X minus the mean of Y (assuming normality of both populations), in percent. (Input)
Default: CONPRM = 95.0.
If CONPRM = 0, no confidence interval for the difference in the means is computed; otherwise, a CONPRM percent confidence interval is computed, in which case CONPRM must be between 0.0 and 100.0. CONPRM is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPRM = 100.0 - 2.0 * (100.0 - ONECL).

CONPRV — Confidence level for inference on variances. (Input)
Default: CONPRV = 95.0.
Under the assumption of equal variances, the pooled variance is used to obtain a two-sided CONPRV percent confidence interval for the common variance in STAT(13) and STAT(14). Without making the assumption of equal variances, the ratio of the variances is of interest. A two-sided CONPRV percent confidence interval for the ratio of the variance of the first population (X) to that of the second population (assuming normality of both populations) is computed and stored in STAT(22) and STAT(23). The confidence intervals are symmetric in probability. See also the description of CONPRM.

IPRINT — Printing option. (Input)
If IPRINT = 0, no printing is performed; otherwise, various statistics in STAT are printed when IDO = 0 or 3.
Default: IPRINT = 0.

IPRINT	Action
0	No printing.
1	Simple statistics (STAT (1) to STAT(6), STAT(24), and STAT(25)).
2	Statistics for means, assuming equal variances.
3	Statistics for means, not assuming equal variances.
4	Statistics for variances.
5	All statistics.

FORTRAN 90 Interface

Generic: CALL TWOMV (X, Y, STAT [, ...])

Specific: The specific interface names are S_TWOMV and D_TWOMV.

FORTRAN 77 Interface

Single: CALL TWOMV (IDO, NROWX, X, NROWY, Y, CONPRM, CONPRV, IPRINT, STAT)

Double: The double precision name is DTWOMV.

Description

The routine TWOMV computes the statistics for making inferences about the means and variances of two normal populations, using independent samples in X and Y. For inferences concerning parameters of a single normal population, see routine UVSTA. For two samples that are paired, see routine ATWOB (see Chapter 3, "Correlation"), since the pairs can be considered to be blocks.

Let μ_X and

$$\sigma_X^2$$

be the mean and variance, respectively, of the first population, and μ_Y and

$$\sigma_Y^2$$

be the corresponding quantities of the second population. The routine TWOMV is used for testing $\mu_X = \mu_Y$ and

$$\sigma_X^2 = \sigma_Y^2$$

or for setting confidence intervals for $\mu_X - \mu_Y$ and

$$\sigma_X^2 / \sigma_Y^2$$

The basic quantities in STAT(1) through STAT(4) are

$$\bar{x} = \sum_{i=1}^{n_x} x_i / n_x, \qquad \bar{y} = \sum_{i=1}^{n_y} y_i / n_y$$

$$s_x^2 = \sum_{i=1}^{n_x} \left(x_i - \bar{x} \right)^2 / \left(n_x - 1 \right), \text{ and } s_y^2 = \sum_{i=1}^{n_y} \left(y_i - \bar{y} \right)^2 / \left(n_y - 1 \right)$$

where n_X and n_Y are the respective sample sizes (in **STAT**(5) and **STAT**(6)).

Inferences about the Means

The test for the equality of means of two normal populations depends on whether or not the variances of the two populations can be considered equal. If the variances are equal, the test is the *two-sample t test*, which is equivalent to an analysis of variance test see (Chapter 4, "Analysis of Variance"). In this case, the statistics returned in **STAT**(7) through **STAT**(12) are appropriate for testing $\mu_X = \mu_Y$. The pooled variance (in **STAT**(7)) is

$$s^2 = \frac{\left(n_x - 1 \right) s_x^2 + \left(n_y - 1 \right) s_y^2}{n_x + n_y - 2}$$

The *t* statistic (in **STAT**(8)) is

$$t = \frac{\bar{x} - \bar{y}}{s \sqrt{\left(1 / n_x \right) + \left(1 / n_y \right)}}$$

For testing $\mu_X = \mu_Y + c$, for some constant *c*, the confidence interval for μ_X **-** μ_Y can be used. (If the confidence interval includes *c*, the null hypothesis would not be rejected at the significance level 1 **-** **CONPRM**/100.)

If the population variances are not equal, the ordinary *t* statistic does not have a *t* distribution; and several approximate tests for the equality of means have been proposed. (See, for example, Anderson and Bancroft 1952, and Kendall and Stuart 1979.) The name *Fisher-Behrens* is associated with this problem, and one of the earliest tests devised for this situation is the Fisher-Behrens test, based on Fisher's concept of *fiducial probability*. Another test is called *Satterthwaite's procedure*. The routine **TWOMV** computes the statistics for this approximation, which was suggested by H.F. Smith and modified by F.E. Satterthwaite (Anderson and Bancroft 1952, page 83). The test statistic is

$$t' = \left(\bar{x} - \bar{y} \right) / s_d$$

where

$$s_d = \sqrt{\left(s_x^2 / n_x \right) + \left(s_y^2 / n_y \right)}$$

Under the null hypothesis of equal population means, this quantity has an approximate *t* distribution with degrees of freedom *f* (in STAT(17)), given by

$$f = \frac{s_d^4}{\frac{\left(s_x^2/n_x\right)^2}{n_x - 1} + \frac{\left(s_y^2/n_y\right)^2}{n_y - 1}}$$

Inferences about the Variances

The *F* statistic for testing the equality of variances is given by

$$F = s_1^2 / s_2^2, \quad \text{where} \quad s_1^2$$

is the larger of

$$s_x^2 \text{ and } s_y^2, \text{ and } s_2^2$$

is the smaller. If the variances are equal, this quantity has an *F* distribution with n_x - 1 and n_y - 1 degrees of freedom.

It is generally not recommended that the results of the *F* test be used to decide whether to use the regular *t* test or the modified *t′* on a single set of data. The more conservative approach is to use the modified *t′* (Satterthwaite's procedure) if there is doubt about the equality of the variances.

Examples

Example 1

This example is taken from Conover and Iman (1983, page 294). It involves scores on arithmetic tests of two grade school classes. The question is whether a group taught by an experimental method has a higher mean score. The data are shown below.

72	111
75	118
77	128
80	138
104	140
110	150

125 163
164
169

It is assumed that the variances of the two populations are equal so the statistics of interest are in STAT(8) and STAT(9). It is seen from the output below that there is strong reason to believe that the two means are different (*t*-value of -4.804). Since the lower 97.5% confidence limit does not include zero, the null hypothesis that $\mu_X \leq \mu_Y$ would be rejected at the 0.05 significance level. (The closeness of the values of the sample variances provides some qualitative substantiation of the assumption of equal variances.)

```
      USE TWOMV_INT

      IMPLICIT   NONE
      INTEGER    IPRINT
      REAL       CONPRV, STAT(25), X(7), Y(9)
!
      DATA X/72., 75., 77., 80., 104., 110., 125./Y/111., 118., 128., &
          138., 140., 150., 163., 164., 169./
!

      IPRINT = 2
      CONPRV = 0.0
      CALL TWOMV (X, Y, STAT, IPRINT=IPRINT, CONPRV=CONPRV)
      END
```

Output

```
 Mean Inferences Assuming Equal Variances
 Pooled Variance                                434.633
 t Value                                         -4.804
 Probability of a Larger t in Abs. Value          0.000
 Degrees of Freedom                              14.000
 Lower Confidence Limit Difference in Means     -73.010
 Upper Confidence Limit Difference in Means     -27.942
```

Example 2

For a second example, the same data set is used to illustrate the use of the IDO parameter to bring in the data one observation at a time. Since there are more "Y" values than "X" values, NROWX is set to zero on the later calls to TWOMV.

```
      USE TWOMV_INT

      IMPLICIT   NONE
      INTEGER    I, IDO, IPRINT, NROWX, NROWY
      REAL       STAT(25), X(7), Y(9)
!
```

```
      DATA X/72., 75., 77., 80., 104., 110., 125./Y/111., 118., 128., &
          138., 140., 150., 163., 164., 169./
!
      IPRINT = 5
      IDO    = 1
      NROWX  = 1
      NROWY  = 1
      DO 10  I=1, 7
!                                Bring in first seven observations
!                                on X and Y, one at a time.
         CALL TWOMV (X(I:), Y(I:), STAT, IDO=IDO, NROWX=NROWX,  &
                     NROWY=NROWY, IPRINT=IPRINT)
         IDO = 2
   10 CONTINUE
!                                Now bring in remaining observations
!                                on Y.
      NROWX = 0
      CALL TWOMV (X(1:), Y(8:), STAT, IDO=IDO, NROWX=NROWX,  &
                  NROWY=NROWY, IPRINT=IPRINT)
!                                Set IDO to indicate last observation.
      IDO = 3
      CALL TWOMV (X(1:), Y(9:), STAT, IDO=IDO, NROWX=NROWX,  &
                  NROWY=NROWY, IPRINT=IPRINT)
      END
```

Output

```
                Statistics from TWOMV
First Sample Mean                              91.857
Second Sample Mean                            142.333
First Sample Variance                         435.810
Second Sample Variance                        433.750
First Sample Valid Observations                 7.000
Second Sample Valid Observations                9.000
First Sample Missing Values                     0.000
Second Sample Missing Values                    0.000

       Mean Inferences Assuming Equal Variances
Pooled Variance                              434.63
t Value                                       -4.80
Probability of a Larger t in Abs. Value        0.00
Degrees of Freedom                            14.00
Lower Confidence Limit Difference in Means   -73.01
Upper Confidence Limit Difference in Means   -27.94
Lower Confidence Limit for Common Variance   232.97
Upper Confidence Limit for Common Variance  1081.04

       Mean Inferences Assuming Unequal Variances
t Value                                     -4.8028
Approx. Prob. of a Larger t in Abs. Value    0.0003
Degrees of Freedom                          13.0290
Lower Confidence Limit                     -73.1758
Upper Confidence Limit                     -27.7766

              Variance Inferences
F Value                                     1.00475
Probability of a Larger F in Abs. Value     0.96571
Lower Confidence Limit for Variance Ratio   0.21600
Upper Confidence Limit for Variance Ratio   5.62621
```

BINES

Estimates the parameter *p* of the binomial distribution.

Required Arguments

N — Total number of Bernoulli trials. (Input)
N is the parameter *N* in the binomial distribution from which one observation (K) has been drawn.

K — Number of successes in the *N* trials. (Input)

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
An approximate CONPER percent confidence interval is computed, hence, CONPER must be between 0.0 and 100.0. CONPER often will be 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set
CONPER = 100.0 - 2.0 * (100.0 - ONECL).

PHAT — Estimate of *p*. (Output)

PLOWER — Lower confidence limit for *p*. (Output)

PUPPER — Upper confidence limit for *p*. (Output)

FORTRAN 90 Interface

Generic: CALL BINES (N, K, CONPER, PHAT, PLOWER, PUPPER)
Specific: The specific interface names are S_BINES and D_BINES.

FORTRAN 77 Interface

Single: CALL BINES (N, K, CONPER, PHAT, PLOWER, PUPPER)
Double: The double precision name is DBINES.

Description

The routine BINES computes a point estimate and a confidence interval for the parameter, *p*, of a binomial distribution, using the number of "successes", *K*, in a sample of size *N* from a binomial distribution with probability function

$$f(x) = \binom{N}{x} p^x (1-p)^{N-x} \quad \text{for } x = 0, 1, \ldots, N$$

The point estimate for p is merely K/N.

The routine **BINES** makes use of the relationship between the binomial distribution and the beta distribution (see Johnson and Kotz 1969, Chapter 3) by solving the following equations equivalent to those in Comment 2:

$$p_L = \beta_{K,N-K+1,\alpha/2}$$

$$p_U = \beta_{K+1,N-K,1-\alpha/2}$$

where $\beta_{a,\,b,\tau}$ is the beta τ critical value with parameters a and b (that is, the inverse beta distribution function evaluated at 1 - τ). The routine **BETIN** see (Chapter 17, "Probability Distribution Function and Inverses") is used to evaluate the critical values.

Comments

1. Informational errors

Type	Code	Description
3	1	CONPER is 100.0 or too large for accurate computations. The confidence limits are set to 0.0 and 1.0.
3	2	CONPER is 0.0 or too small for accurate computations. The confidence limits are both set to PHAT.

2. Since the binomial is a discrete distribution, it is not possible to construct an exact **CONPER**% confidence interval for all values of **CONPER**. Let α = 1 - **CONPER**/100. Then, the approximate lower and upper confidence limits p_L and p_U (**PLOWER** and **PUPPER**) are solutions to the equations

$$\sum_{x=K}^{N} \binom{N}{x} p_L^x (1-p_L)^{N-x} = \alpha/2$$

$$\sum_{x=0}^{K} \binom{N}{x} p_U^x (1-p_U)^{N-x} = \alpha/2$$

These approximations are not just computational devices. Approximations to the confidence limits are necessary because the binomial distribution is discrete.

Example

In this example, we assume that the number of defective microchips in a given lot follows a binomial distribution. We estimate the proportion defective by taking a sample of 50. In this sample, 3 microchips were found to be defective. The routine `BINES` is used to estimate *p* and to compute a 95% confidence interval.

```
          USE BINES_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    K, N, NOUT
          REAL       CONPER, PHAT, PLOWER, PUPPER
    !
          CALL UMACH (2, NOUT)
          N      = 50
          K      = 3
          CONPER = 95.0
          CALL BINES (N, K, CONPER, PHAT, PLOWER, PUPPER)
          WRITE (NOUT,99999) PHAT, PLOWER, PUPPER
    99999 FORMAT ('  Point estimate of the proportion:   ', F5.3, /, &
                  '  95% confidence interval:   (', F5.3, ',', F5.3, &
                  ')')
          END
```

Output

```
    Point estimate of the proportion:    .060
    95% confidence interval:   ( .013, .165)
```

POIES

Estimates the parameter of the Poisson distribution.

Required Arguments

IX — Vector of length **NOBS** containing the data. (Input)
The data are assumed to be a random sample from a Poisson distribution; hence, all elements of **IX** must be nonnegative.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
An approximate **CONPER** percent confidence interval is computed; hence, **CONPER** must be between 0.0 and 100.0. **CONPER** often will be 90.0, 95.0, or 99.0. For a one sided confidence interval with confidence level **ONECL**, set
CONPER = 100.0 **-** 2.0 ***** (100.0 **-** **ONECL**).

THAT — Estimate of the parameter, theta (the mean). (Output)

TLOWER — Lower confidence limit for theta. (Output)

TUPPER — Upper confidence limit for theta. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**IX**,1).

FORTRAN 90 Interface

Generic: **CALL POIES** (**IX**, **CONPER**, **THAT**, **TLOWER**, **TUPPER** [, ...])
Specific: The specific interface names are **S_POIES** and **D_POIES**.

FORTRAN 77 Interface

Single: **CALL POIES** (**NOBS**, **IX**, **CONPER**, **THAT**, **TLOWER**, **TUPPER**)
Double: The double precision name is **DPOIES**.

Description

The routine **POIES** computes a point estimate and a confidence interval for the parameter, θ, of a Poisson distribution. It is assumed that the vector **IX** contains a random sample of size **NOBS** from a Poisson distribution with probability function

$$f(x) = e^{-\theta}\theta^{x} / x!, \text{ for } x = 0, 1, 2, \ldots$$

The point estimate for θ corresponds to the sample mean.

By exploiting the relationship between the Poisson distribution and the chi-squared distribution (see Johnson and Kotz, 1969, Chapter 4), the equations in Comment 2 can be written as

$$\theta_L = \tfrac{1}{2}\chi^2_{2k,\alpha/2}$$
$$\theta_U = \tfrac{1}{2}\chi^2_{2k+2,1-\alpha/2}$$

where

$$\chi^2_{\nu,\tau}$$

is the chi-squared τ critical value with degrees ν of freedom (that is, the inverse chi-squared distribution function evaluated at 1 **-** τ). The routine **CHIIN** (see Chapter 17, "Probability Distribution Functions and Inverses") is used to evaluate the critical values.

For more than one observation, the estimates are obtained as above and then divided by the number of observations, **NOBS**.

Comments

1. Informational error

Type	Code	Description
3	1	CONPER is 0.0 or too small for accurate computations. The confidence limits are both set to THAT.

2. Since the Poisson is a discrete distribution, it is not possible to construct an exact **CONPER**% confidence interval for all values of **CONPER**. Let α = 1 **-** **CONPER**/100, and let *k* be a single observation. Then, the approximate lower and upper confidence limits θ_L and θ_U (**TLOWER** and **TUPPER**) are solutions to the equations

$$\exp(-\theta_L)\sum_{x=k}^{\infty}\theta_L^x / x! = \alpha/2$$

$$\exp(-\theta_U)\sum_{x=0}^{k}\theta_U^x / x! = \alpha/2$$

Example

It is assumed that flight arrivals at a major airport during the middle of the day follow a Poisson distribution. It is desired to estimate the mean number of arrivals per minute and to obtain an upper one-sided 95% confidence interval for the mean. During a half-hour period, the number of arrivals each minute was recorded. These data are stored in `IX`, and `POIES` is used to obtain the estimates.

```
      USE POIES_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=30)
!
      INTEGER    IX(NOBS), NOUT
      REAL       CONPER, THAT, TLOWER, TUPPER
!
      DATA IX/2, 0, 1, 1, 2, 0, 3, 1, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, &
          0, 1, 2, 0, 2, 0, 0, 1, 2, 0, 2/
!
      CALL UMACH (2, NOUT)
!                                For a 95 percent one-sided ! .I.,
!                                CONPER = 100.0 - 2.0*(100.0-95.0)
      CONPER = 90.0
      CALL POIES (IX, CONPER, THAT, TLOWER, TUPPER)
      WRITE (NOUT,99999) THAT, TUPPER
99999 FORMAT ('  Point estimate of the Poisson mean:   ', F5.3, /, &
             '  Upper one-sided 95% confidence limit: ', F5.3)
      END
```

Output

```
 Point estimate of the Poisson mean: 0.800
 Upper one-sided 95% confidence limit: 1.125
```

NRCES

Computes maximum likelihood estimates of the mean and variance from grouped and/or censored normal data.

Required Arguments

XRT — Vector of length NOBS containing either the exact value of the data or the right endpoint of the censoring interval for interval-censored or right-censored data. (Input)
See the argument ICEN.

XLT — Vector of length NOBS containing the left endpoint of the censoring interval for interval-censored or left-censored data. (Input)
See the argument ICEN. XLT is not used if there is no left censoring.

ICEN — Vector of length NOBS containing the censoring codes. (Input)
The values in ICEN indicate the meaning of the values in XRT and/or XLT.

ICEN(I)	Censoring
0	Exact response at XRT(I).
1	Right censored. The response is greater than XRT(I).
2	Left censored. The response is less than or equal to XLT(I).
3	Interval censored. The response is greater than XRT(I), but less than or equal to XLT(I).

XMEAN — Estimate of the mean. (Input/Output if INIT = 0; output otherwise)

XSIGMA — Estimate of the standard deviation. (Input/Output if INIT = 0; output otherwise)

VXM — Estimate of the variance of the mean estimate. (Output)

VXS — Estimate of the variance of the variance estimate. (Output)

COVXMS — Estimate of the covariance of the mean and the variance estimates. (Output)

NUMBER — Vector of length 4 containing the numbers of observations having the various censoring properties. (Output)
NUMBER(1) is the number of exact observations. NUMBER(2) is the number of observations specified by a lower bound (right censored). NUMBER(3) is the number of observations specified by a upper bound (left censored). NUMBER(4) is the number of observations specified by an interval.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (XRT,1).

EPSM — Convergence criterion for the mean estimate. (Input)
See the argument EPSSIG. If EPSM is not positive, EPSM = 0.00001 is assumed.
Default: EPSM = .00001.

EPSSIG — Convergence criterion for the variance estimate. (Input)
Convergence is assumed when the relative change in the mean estimate is less than EPSM and the relative change in the variance estimate is less than EPSSIG . If EPSSIG is not positive, EPSSIG = 0.00001 is assumed.
Default: EPSSIG = .00001.

MAXITS — Maximum number of iterations allowed. (Input)
A typical value of MAXITS is 25.
Default: MAXITS = 25.

INIT — Initialization option. (Input)
Default: INIT = 1.

INIT	Action
0	On input, XMEAN and XSIGMA contain initial estimates of the parameters.
1	If there are enough exactly specified data, initial estimates are obtained from it; and, if there are not enough such data, fixed starting values (XRT(1) for the mean and 1.0 for the variance) are used.

FORTRAN 90 Interface

Generic: CALL NRCES (XRT, XLT, ICEN, XMEAN, XSIGMA, VXM, VXS, COVXMS, NUMBER [, ...])

Specific: The specific interface names are S_NRCES and D_NRCES.

FORTRAN 77 Interface

Single: CALL NRCES (NOBS, XRT, XLT, ICEN, EPSM, EPSSIG, MAXITS, INIT, XMEAN, XSIGMA, VXM, VXS, COVXMS, NUMBER)

Double: The double precision name is DNRCES.

Description

The routine **NRCES** computes maximum likelihood estimates of the mean and variance of a normal population, using a sample that may be censored. An observation whose value is known exactly is input in **XRT**, and the corresponding element in **ICEN** is set to 0. If an observation is known only by a lower bound, we say the observation is *right censored*; the lower bound is input in **XRT**, and the corresponding element in **ICEN** is set to 1. If an observation is known only by an upper bound, we say the observation is *left censored*; the upper bound is input in **XLT**, and the corresponding element in **ICEN** is set to 2. If an observation is known only by two bounds, we say the observation is *interval censored*; the lower bound is input in **XRT**, the upper bound is input in **XLT**, and the corresponding element in **ICEN** is set to 3.

Newton-Raphson iterations are used to find a stationary point of the likelihood function, and the Hessian at that point is used to estimate the variances and covariance of the estimates of the population mean and variance. If the numerical derivative of the estimate of the variance increases on nine consecutive iterations, the process is deemed divergent and a terminal error is issued. The iterations begin at user-supplied values if **INIT** is set to 0.

Example

This example uses an artificial data set consisting of 18 observations. The first 12 observations are known exactly; the next three are known only by a lower bound; the next two, by an upper bound; and the last one, by two bounds.

```
      USE NRCES_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=18)
!
      INTEGER    ICEN(NOBS), INIT, MAXITS, NOUT, NUMBER(4)
      REAL       COVXMS, EPSM, EPSSIG, VXM, VXS, XLT(NOBS), XMEAN, &
                XRT(NOBS), XSIGMA
!
      DATA XRT/4.5, 5.4, 3.9, 5.1, 4.6, 4.8, 2.9, 6.3, 5.5, 4.6, 4.1, &
          5.2, 3.2, 4.0, 3.1, 0.0, 0.0, 2.2/
      DATA XLT/0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
          0.0, 0.0, 0.0, 0.0, 5.1, 3.8, 2.5/
      DATA ICEN/0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 3/
!
      CALL UMACH (2, NOUT)
      EPSM   = 0.01
      EPSSIG = 0.01
      MAXITS = 25
      INIT   = 1
      CALL NRCES (XRT, XLT, ICEN, XMEAN, XSIGMA, VXM, VXS, COVXMS,  &
                  NUMBER, EPSM=EPSM, EPSSIG=EPSSIG)
      WRITE (NOUT,99999) XMEAN, XSIGMA, VXM, VXS, COVXMS, NUMBER
99999 FORMAT (' Estimate of mean:                          ', F8.4, &
            /, ' Estimate of variance:                      ', F8.4, &
```

```
               /, ' Estimate of variance of mean estimate:      ', F8.4, &
               /, ' Estimate of variance of variance estimate:  ', F8.4, &
               /, ' Estimate of covariance of mean and variance:', F8.4, &
               /, ' Number of exact observations:               ', I4, &
               /, ' Number of right-censored observations:      ', I4, &
               /, ' Number of left-censored observations:       ', I4, &
               /, ' Number of interval-censored observations:   ', I4)
          END
```

Output

```

    Estimate of mean:                              4.4990
    Estimate of standard deviation:                1.2304
    Estimate of variance of mean estimate:         0.0819
    Estimate of variance of variance estimate:    -0.0494
    Estimate of covariance of mean and variance: -0.0019
    Number of exact observations:                  12
    Number of right-censored observations:          3
    Number of left-censored observations:           2
    Number of interval-censored observations:       1
```

GRPES

Computes basic statistics from grouped data.

Required Arguments

TABLE — Vector of length NGROUP containing the frequencies within the groups. (Input)
The entries in TABLE are interpreted as counts. They must be nonnegative.

CLOW — The center (class mark) of the lowest class interval. (Input)

CWIDTH — The class width. (Input)
CWIDTH must be positive.

STAT — Vector of length 13 containing the statistics. (Output)

I	STAT(I)
1	The sum of the frequencies in TABLE.
2	Mean (arithmetic mean, first moment).
3	Sample standard deviation. (Uses STAT(1) - 1 as divisor).
4	Second moment about the mean, uncorrected for grouping. (Uses STAT(1) as divisor.)
5	Second moment about the mean, adjusted using Sheppard's correction.
6	Third moment about the mean, uncorrected for grouping.
7	Third moment about the mean, adjusted using Sheppard's correction.
8	Fourth moment about the mean, uncorrected for grouping.
9	Fourth moment about the mean, adjusted using Sheppard's correction.
10	Median.
11	Geometric mean; defined only if CLOW - CWIDTH/2 is nonnegative.
12	Harmonic mean; defined only if CLOW - CWIDTH/2 is nonnegative.
13	Mode; defined only if one element of TABLE is strictly greater than all other elements of TABLE.

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (TABLE,1).

IPRINT — Printing option. (Input)
If IPRINT = 0, no printing is performed; and if IPRINT = 1, the statistics in STAT are printed.
Default: IPRINT = 0.

FORTRAN 90 Interface

Generic: CALL GRPES (TABLE, CLOW, CWIDTH, STAT [, ...])
Specific: The specific interface names are S_GRPES and D_GRPES.

FORTRAN 77 Interface

Single: CALL GRPES (NGROUP, TABLE, CLOW, CWIDTH, IPRINT, STAT)

Double: The double precision name is **DGRPES**.

Description

The routine **GRPES** computes various statistics using data from equally spaced groups. The second, third, and fourth moments are computed both with and without Sheppard's corrections. These corrections for grouped data are most useful for distributions whose densities tail off smoothly (such as the normal distribution). Kendall, Stuart, and Ord (1987, Chapters 2 and 3) discuss these corrections.

The moments are computed using the sum of the frequencies as the divisor. The standard deviation (**STAT**(3)), on the other hand, is computed using as the divisor the sum of the frequencies minus one.

If any of the class marks are negative, the geometric and harmonic means are not computed, and NaN (not a number) is stored as the value of **STAT**(11). Likewise, if the mode does not exist (no group has a frequency greater than that of all other groups), NaN is stored as the value of **STAT**(13).

Examples

Example 1

This example is taken from Conover and Iman (1983, page 119). The objective is to compute some basic statistics relating to test scores, using the following data:

91 - 100	7
81 - 90	13
71 - 80	11
61 - 70	5
≤ 60	4

```
          USE GRPES_INT

          IMPLICIT   NONE
          INTEGER    IPRINT, NGROUP
          REAL       CLOW, CWIDTH, STAT(13), TABLE(5)
     !
          NGROUP   = 5
          CLOW     = 55.5
          CWIDTH   = 10.0
          TABLE(1) = 4.0
          TABLE(2) = 5.0
          TABLE(3) = 11.0
```

```
      TABLE(4) = 13.0
      TABLE(5) = 7.0
      IPRINT   = 1
      CALL GRPES (TABLE, CLOW, CWIDTH, STAT, IPRINT=IPRINT)
      END
```

Output

```
 Statistics from GRPES
 Sum freqs.        40.0
 Mean              79.0
 Std. dev.         12.1
 2nd moment       142.8
 2nd, adj.        134.4
 3rd moment      -741.8
 3rd, adj.      -2716.8
 4th moment     48242.3
 4th, adj.      47929.0
 Median            80.5
 Geometric         78.0
 Harmonic          77.0
 Mode              85.5
```

Example 2

In this example, there are negative values of some class marks, and there is no modal class.

-2.0	2
-1.0	5
0.0	7
1.0	7
2.0	2

```
      USE GRPES_INT

      IMPLICIT   NONE
      INTEGER    NGROUP, IPRINT
      REAL       TABLE(5), CLOW, CWIDTH, STAT(13)
!
      NGROUP = 5
      CLOW = -2.0
      CWIDTH = 1.0
      TABLE(1) = 2.0
      TABLE(2) = 5.0
      TABLE(3) = 7.0
      TABLE(4) = 7.0
      TABLE(5) = 2.0
      IPRINT = 1
      CALL GRPES (TABLE, CLOW, CWIDTH, STAT, IPRINT=IPRINT)
      END
```

Output

```
    Statistics from GRPES
    Sum freqs.     23.0000
    Mean            0.0870
    Std. dev.       1.1246
    2nd moment      1.2098
    2nd, adj.       1.1265
    3rd moment     -0.2293
    3rd, adj.      -0.2510
    4th moment      3.3292
    4th, adj.       2.7960
    Median          0.1429

    The mode is not defined, since no class has higher
    frequency than all others.
    The geometric and harmonic means are not defined, since
    the lower bound is negative.
```

CSTAT

Computes cell frequencies, cell means, and cell sums of squares for multivariate data.

Required Arguments

X — |NROW| by NCOL matrix containing the data. (Input)
Each column of X represents either a classification variable, a response variable, a weight, or a frequency.

KMAX — Maximum number of cells. (Input)
This quantity does not have to be exact, but must be at least as large as the actual number of cells, K.

CELIF — Matrix with min(KMAX, K) columns containing cell information.
(Output, if IDO = 0 or 1; input/output, if IDO = 2.)
The number of rows in CELIF depends on the eight cases tabled below.

	Contents	Rows in CELIF
1	MOPT ≤ 0, IFRQ = 0 and IWT = 0	NCOL + NR + 1
2	MOPT ≤ 0, IFRQ > 0 and IWT = 0	NCOL + NR
3	MOPT ≤ 0, IFRQ = 0 and IWT > 0	NCOL + NR + 1
4	MOPT ≤ 0, IFRQ > 0 and IWT > 0	NCOL + NR
5	MOPT > 0, IFRQ = 0 and IWT = 0	NCOL + 2 * NR + 1
6	MOPT > 0, IFRQ > 0 and IWT = 0	NCOL + 2 * NR
7	MOPT > 0, IFRQ = 0 and IWT > 0	NCOL + 3 * NR
8	MOPT > 0, IFRQ > 0 and IWT > 0	NCOL + 3 * NR - 1

Each column contains information on each unique combination of values of the *m* classification variables that occurs in the data. The first *m* rows give the values of the classification variables. Row *m* + 1 gives the number of observations that are in this cell. (For cases 2, 4, 6 and 8, this is the sum of the frequencies.) For case 3 and 4, row *m* + 2 contains the sum of the weights. For NR greater than zero, the remaining rows (beginning with row *m* + 3 in case 3 and 4 and with row *m* + 2 otherwise) contain information concerning the response variables. For cases 1, 2, 3 and 4, there are 2 ∗ NR remaining rows with the cell (weighted) mean and cell (weighted) sum of squares for each of the NR response variables. For cases 5 and 6, there are 3 ∗ NR remaining rows with the sample size, the

mean and sum of squares for each of the NR response variables. For case 7 and 8, there are 4 ∗ NR remaining rows with the sample size, the sum of weights, weighted means, and weighted sum of squares for each of the NR response variables.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of CSTAT for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to CSTAT will be made. Initialization and updating for the data in X are performed.
2	This is an intermediate invocation of CSTAT, and updating for the data in X is performed.

NROW — The absolute value of NROW is the number of rows of data currently input in X. (Input)
Default: NROW = size (X,1).
NROW may be positive or negative. Negative NROW means that the -NROW rows of data are to be deleted from some aspects of the analysis, and this should be done only if IDO is 2. When a negative value is input for NROW, it is assumed that each of the -NROW rows of X has been input (with positive NROW) in previous invocations of CSTAT.

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NR — Number of response variables. (Input)
NR = 0 means no response variables are input. Otherwise, cell means and sums of squares are computed for the response variables.
Default: NR = 0.

IRX — Vector of length NR. (Input if NR is greater than 0.)
The IRX(1), ..., IRX(NR) columns of X contain the response variables for which cell means and sums of squares are computed.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column IWT of X contains the weights.
Default: IWT = 0.

MOPT — Missing value option. (Input)
If MOPT is zero, the exclusion is listwise. If MOPT is positive, the following occurs: (1) if a classification variable's value is missing, the entire case is excluded, (2) if
IFRQ > 0 and the frequency variable's value is missing, the entire case is excluded, (3) if IWT > 0 and the weight variable's value is missing, the case is classified and the cell frequency updated, but no information with regard to the response variables is computed, and (4) if only some response variables' values are missing, all computations are performed except those pertaining to the response variables with missing values.
Default: MOPT = 0.

K — Number of cells or an upper bound for this number. (Input/Output)
On the first call K must be input K = 0. It should not be changed between calls to CSTAT. K is incremented by one for each new cell up to KMAX cells. Once KMAX cells are encountered, K is incremented by one for each observation that does not fall into one of the KMAX cells. In this case, K is an upper bound on the number of cells and can be used for KMAX in a subsequent run.
Default: K = 0.

LDCELI — Leading dimension of CELIF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCELI = size (CELIF,1).

FORTRAN 90 Interface

Generic: CALL CSTAT (X, KMAX, CELIF [, ...])
Specific: The specific interface names are S_CSTAT and D_CSTAT.

FORTRAN 77 Interface

Single: CALL CSTAT (IDO, NROW, NCOL, X, LDX, NR, IRX, IFRQ, IWT, MOPT, KMAX, K, CELIF, LDCELI)
Double: The double precision name is DCSTAT.

Description

The routine **CSTAT** computes cell frequencies, cell means, and cell sums of squares for multivariate data in **X**. The columns of **X** can contain data for four types of variables: classification variables, a frequency variable, a weight variable, and response variables. The frequency variable, the weight variable, and the response variables are all designated by indicators in **IFRQ**, **IWT**, and **IRX**. All other variables are considered to be classification variables; hence, there are *m* classification variables, where *m* = **NCOL - NR** if there is no weight or frequency variable, *m* = **NCOL - NR -** 1 if there is a weight or frequency variable but not both, and *m* = **NCOL - NR -** 2 if there are weight and frequency variables.

Each combination of values of the classification variables is stored in the first m rows of **CELIF**. For each combination of values of the classification variables, the frequencies are stored in the next row of **CELIF**. Then, for each combination, means and sums of squares for each of the response variables are computed and stored in the remaining rows of **CELIF**. If a weighting variable is specified, the sum of the weights for each combination is computed and stored. If missing values are deleted elementwise (that is, if **MOPT** is positive), the frequencies and sums of weights for each of the response variables are stored in the rows of **CELIF**.

Comments

1. If no nonmissing observations with positive weights or frequencies exist in a cell for a particular response variable, the mean and sum of squares are set to NaN (not a number).
2. In cases 3 and 6, if a zero weight is encountered, there is no contribution to the means or sums of squares, but the sample sizes are implemented by one for that observation.

Examples

Example 1

In this example, there are two classification variables, C_1 and C_2, and two response variables, R_1 and R_2. Their values are shown below.

		C_1			
		1		2	
		R_1	R_2	R_1	R_2
C_2	1	5.0 7.0	3.4 2.6	3.8 5.2 4.9	2.4 6.3 1.2
	2	4.3 3.2 1.7	9.8 7.1 6.3	6.5 3.1	3.4 5.1

```
      USE CSTAT_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    KMAX, LDCELI, LDX, NR, NCOL
      PARAMETER  (KMAX=4, LDCELI=15, LDX=10, NR=2, NCOL=4)
!
      INTEGER    IDO, IFRQ, IRX(NR), IWT, K, MIN0, MOPT, NROW
      REAL       CELIF(LDCELI,KMAX), X(LDX,NCOL)
      CHARACTER  CLABEL(1)*6, FMT*7, RLABEL(7)*6
      INTRINSIC  MIN0
!                                Get data for example
      DATA X/1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.0, &
          1.0, 2.0, 2.0, 2.0, 1.0, 1.0, 1.0, 2.0, 2.0, 5.0, 7.0, 4.3, &
          3.2, 1.7, 3.8, 5.2, 4.9, 6.5, 3.1, 3.4, 2.6, 9.8, 7.1, 6.3, &
          2.4, 6.3, 1.2, 3.4, 5.1/
!                                All data are input at once
      IDO  = 0
      NROW = 10
      K    = 0
!                                No unequal frequencies or weights
!                                are used
      IFRQ = 0
      IWT  = 0
!                                Response variables are in 3rd and 4th
!                                columns
      IRX(1) = 3
      IRX(2) = 4
!                                Delete any row containing a missing
!                                value
      MOPT = 0
!
      CALL CSTAT (X, KMAX, CELIF, NR=NR, IRX=IRX, K=K)
!                                Print the results
```

```
      CLABEL(1) = 'NONE'
      RLABEL(1) = ' '
      RLABEL(2) = ' '
      RLABEL(3) = 'Freq.'
      RLABEL(4) = 'Mean 1'
      RLABEL(5) = 'SS 1'
      RLABEL(6) = 'Mean 2'
      RLABEL(7) = 'SS 2'
      FMT       = '(W10.4)'
      CALL WRRRL ('Statistics for the Cells', CELIF, &
                  RLABEL, CLABEL, NRA=(NCOL+NR+1), &
                  NCA=MIN0(KMAX, K), FMT=FMT)
      END
```

Output

```
                Statistics for the Cells
                 1.00        1.00        2.00        2.00
                 1.00        2.00        1.00        2.00
Freq.            2.00        3.00        3.00        2.00
Mean 1           6.00        3.07        4.63        4.80
SS 1             2.00        3.41        1.09        5.78
Mean 2           3.00        7.73        3.30        4.25
SS 2             0.32        6.73       14.22        1.44
```

Example 2

This example uses the same data as in the first example, except some of the data are set to missing values. Also, a frequency variable is used. It is in the fourth column of **X**. The frequency variable indicates that the values of the classification and response variables in the first observation occur 3 times and that all other frequencies are 1. Since **MOPT** is greater than zero, statistics on one response variable are accumulated even if the other response variable has a missing value. If the frequency variable has a missing value, however, the entire observation is omitted.

The missing value is NaN (not a number) that can be obtained with the argument of 6 in the routine **AMACH** (Reference Material). For this example, we set the first response variable in the first cell ($C_1 = 1$, $C_2 = 1$) to a missing value; we set the second response variable in the (2, 1) cell to a missing value; and we set the frequency variable in the (1, 2) cell to a missing value. The data are now as shown below, with "NaN" in place of the missing values.

		c_1			
		1		2	
		R_1	R_2	R_1	R_2
c_2	1	NaN NaN NaN 7.0	3.4 3.4 3.4 2.6	3.8 5.2 4.9	NaN 6.3 1.2
	2	NaN 3.2 1.7	NaN 7.1 6.3	6.5 3.1	3.4 5.1

The first two rows output in **CELIF** are the values of the classification variables, and the third row is the frequencies of the cells, as before. The next three rows correspond to the first response variable, and the last three rows correspond to the second response variable. (This is "case 6" above, where the argument **CELIF** is described.)

```
      USE CSTAT_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    KMAX, LDCELI, LDX, NR, NCOL, NROW
      PARAMETER  (KMAX=4, LDCELI=15, LDX=10, NR=2, NCOL=5)
!
      INTEGER    IDO, IFRQ, IRX(NR), IWT, K, MIN0, MOPT
      REAL       CELIF(LDCELI,KMAX), X(LDX,NCOL), AMACH
      INTRINSIC   MIN0
!                                Get data for example.
      DATA X/1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.0, &
           1.0, 2.0, 2.0, 2.0, 1.0, 1.0, 1.0, 2.0, 2.0, 5.0, 7.0, 4.3, &
           3.2, 1.7, 3.8, 5.2, 4.9, 6.5, 3.1, 3.4, 2.6, 9.8, 7.1, 6.3, &
           2.4, 6.3, 1.2, 3.4, 5.1, 3.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, &
           1.0, 1.0, 1.0/
!                                All data are input at once.
      IDO  = 0
      NROW = 10
      K    = 0
!                                Frequencies are in the 5th column.
!                                All weights are equal
      IFRQ = 5
      IWT  = 0
!                                Response variables are in 3rd and 4th
!                                columns.
      IRX(1) = 3
      IRX(2) = 4
!                                Set some values to "missing" for
!                                this example.  Specify elementwise
!                                deletion of missing values of the
!                                response variables.
      MOPT   = 1
      X(1,3) = AMACH(6)
      X(6,4) = AMACH(6)
      X(3,5) = AMACH(6)
!
```

```
      CALL CSTAT (X, KMAX, CELIF, NR=NR, IRX=IRX, MOPT=MOPT, IFRQ=IFRQ, &
                  K=K)
!                                 Print the results.
      CALL WRRRN ('Statistics for the Cells', CELIF, NRA=(NCOL+2*NR), &
                  NCA=MIN0(KMAX, K))
      END
```

Output

```
                Statistics for the Cells
                  1       2       3       4
    1    1.00    1.00    2.00    2.00
    2    1.00    2.00    1.00    2.00
    3    4.00    2.00    3.00    2.00
    4    1.00    2.00    3.00    2.00
    5    7.00    2.45    4.63    4.80
    6    0.00    1.12    1.09    5.78
    7    4.00    2.00    2.00    2.00
    8    3.20    6.70    3.75    4.25
    9    0.48    0.32   13.01    1.44
```

MEDPL

Computes a median polish of a two-way table.

Required Arguments

TABLE — NROW by NCOL matrix containing the table. (Input)

MAXIT — Maximum number of polishing iterations to be performed. (Input)
An iteration is counted each time the rows or the columns are polished. The iterations begin by polishing the rows.

PTABLE — (NROW + 1) by (NCOL + 1) matrix containing the cell residuals from the fitted table and, in the last row and column, the marginal residuals. (Output)

Optional Arguments

NROW — Number of rows in the table. (Input)
Default: NROW = size (TABLE,1).

NCOL — Number of columns in the table. (Input)
Default: NCOL = size (TABLE,2).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

LDPTAB — Leading dimension of PTABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPTAB = size (PTABLE,1).

ITER — Number of iterations actually performed. (Output)

FORTRAN 90 Interface

Generic: CALL MEDPL (TABLE, MAXIT, PTABLE [, ...])

Specific: The specific interface names are S_MEDPL and D_MEDPL.

FORTRAN 77 Interface

Single: CALL MEDPL (NROW, NCOL, TABLE, LDTABL, MAXIT, PTABLE, LDPTAB, ITER)

Double: The double precision name is DMEDPL.

Description

The routine MEDPL performs a median polish on a two-way table. It first copies TABLE into PTABLE and fills the last row and last column of PTABLE with zeroes. It then computes the row-wise medians, adds these to the values in the last column and corresponding row, and subtracts them from the other entries in the corresponding row. Similar computations are then performed for all NCOL + 1 columns. The whole procedure is then repeated (using NROW + 1 rows) until convergence is achieved (until no changes occur), or until MAXIT iterations are performed. Convergence is known to have occurred if ITER is less than MAXIT.

As Emerson and Hoaglin (1983) discuss, it is not necessarily desirable to continue until convergence. If MAXIT is set to twice the maximum of the number of rows and columns plus five, it is likely that convergence will occur.

As Emerson and Hoaglin point out, median polish starting with rows can lead to a different fit from that obtained by starting with columns. Although MEDPL does not make provision for choosing which dimension to start with, TABLE can be transposed by use of routine
TRNRR (IMSL MATH/LIBRARY). Use of the transposed table in MEDPL would result in the iterations beginning with the columns of the original table. Further descriptions of median polish, which was first proposed by John Tukey, and examples of its use can be found in Tukey (1977, Chapter 11) and in Velleman and Hoaglin (1981, Chapter 8).

Comments

Workspace may be explicitly provided, if desired, by use of M2DPL/DM2DPL.

The reference is:

```
CALL M2DPL (NROW, NCOL, TABLE, LDTABL, MAXIT, PTABLE, LDPTAB, ITER, WK)
```

The additional argument is:

WK — Work vector of length max(NROW, NCOL).

Example

This example is taken from Emerson and Hoaglin (1983, page 168). It involves data on infant mortality in the United States, classified by father's education and by region of the country. In order to show the difference between making only one polishing pass over the rows and polishing until convergence, on the first invocation

MAXIT is set to one. On a second call, it is set large enough to have reasonable assurance of execution until convergence. In the first case, the last row and column of PTABLE are printed. The values in these are the medians before any polishing. These values approach zero as the polishing continues.

```
      USE MEDPL_INT
      USE UMACH_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    NCOL, NROW
      PARAMETER  (NCOL=5, NROW=4)
!
      INTEGER    ITER, LDPTAB, LDTABL, MAXIT, NOUT
      REAL       PTABLE(NROW+1,NCOL+1), TABLE(NROW,NCOL)
      CHARACTER  CLABEL(1)*5, RLABEL(1)*5
!
      DATA CLABEL/'NONE'/
      DATA RLABEL/'NONE'/
      DATA TABLE/25.3, 32.1, 38.8, 25.4, 25.3, 29.0, 31.0, 21.1, 18.2, &
          18.8, 19.3, 20.3, 18.3, 24.3, 15.7, 24.0, 16.3, 19.0, 16.8, &
          17.5/
!
      CALL UMACH (2, NOUT)
      MAXIT  = 1
      LDTABL = 4
      LDPTAB = 5
      CALL MEDPL (TABLE, MAXIT, PTABLE, ITER=ITER)
      CALL WRRRL ('Fitted table after one iteration over the rows', &
                  PTABLE, CLABEL, RLABEL, FMT='(W10.4)')
      MAXIT = 15
      CALL MEDPL (TABLE, MAXIT, PTABLE, ITER=ITER)
      CALL WRRRL ('%/Fitted table and marginal residuals', PTABLE, &
                  CLABEL, RLABEL, FMT='(W10.4)')
      WRITE (NOUT,99999) ITER
99999 FORMAT (/, ' Iterations taken: ', I2)
      END
```

Output

```
      Fitted table after one iteration over the rows
 7.0         7.0        -0.1         0.0        -2.0        18.3
 7.8         4.7        -5.5         0.0        -5.3        24.3
19.5        11.7         0.0        -3.6        -2.5        19.3
 4.3         0.0        -0.8         2.9        -3.6        21.1
 0.0         0.0         0.0         0.0         0.0         0.0

                  Fitted table and marginal residuals
-1.55        0.00        0.00       -1.15        0.60       -1.45
 1.55        0.00       -3.10        1.15       -0.40        2.25
10.85        4.60        0.00       -4.85        0.00       -0.35
-3.25       -6.00        0.30        2.75        0.00        0.35
 8.10        6.55       -0.55        0.70       -3.05       20.20
Iterations taken: 15
```

Regression

Routines

Usage Notes

Simple Linear Regression

The simple linear regression model is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad i = 1, 2, \ \ldots, n$$

where the observed values of the y_i's constitute the responses or values of the dependent variable, the x_i's are the settings of the independent (explanatory) variable, β_0 and β_1 are the intercept and slope parameters, respectively, and the ε_i's are independently distributed normal errors each with mean zero and variance σ^2.

Routine **RLINE** fits a straight line and computes summary statistics for the simple linear regression model. There are no options with this routine.

Routine **RONE** analyzes a simple linear regression model. Routine **RONE** requires a data matrix as input. There is an option for excluding the intercept β_0 from the model. The variables *x*, *y*, weights (optional), and frequencies (optional) must correspond to columns in this matrix. The simple linear regression model is fit, summary statistics are computed (including a test for lack of fit), and confidence intervals and diagnostics for individual cases are computed. There are options for printing and plotting the results.

Routines **RINCF** and **RINPF** solve the inverse regression (calibration) problem using a fitted simple linear regression. Routines **RLINE** or **RONE** can be used to compute the fit. Routine **RINCF** estimates settings of the independent variable that restrict, at a specified confidence percentage, *y* to a given specified range. Routine **RINPF** computes a confidence interval on the setting of the independent variable for a given response y_0.

Multiple Linear Regression

The multiple linear regression model is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_k x_{ik} + \varepsilon_i \quad i = 1,2,\ldots n$$

where the observed values of the y_i's constitute the responses or values of the dependent variable, the x_{i1}'s, x_{i2}'s, …, x_{ik}'s are the settings of the *k* independent (explanatory) variables, β_0, β_1, …, β_k are the regression coefficients, and the ε_i's are independently distributed normal errors each with mean zero and variance σ^2.

Routine RLSE fits the multiple linear regression model. There is an option for excluding the intercept β_0. There are no other options. The responses are input in a one-dimensional array **Y**, and the independent variables are input in a two-dimensional array **X** that contains the individual cases as the rows and the variables as the columns.

By specifying a single dependent variable, either RGIVN or RCOV can also be used to fit the multiple linear regression. (These routines are designed to fit any number of dependent variables simultaneously. See the section Multivariate General Linear Model.)

Routine **RGIVN** fits the model using fast Givens transformations. For large data sets that cannot be stored in a single array, **RGIVN** is designed to allow multiple invocations. In this case, only some of the rows from the entire data set are input at any one time. Alternatively, the data set can be input in a single array.

Routine **RCOV** fits the multiple linear regression model from the sum of squares and crossproducts matrix for the data $(x_1, x_2, \ldots, x_k, y)$. Routine CORVC in Chapter 3, "Correlation" can compute the required sums of squares and crossproducts matrix for input into **RCOV**. Routine RORDM in Chapter 19, "Utilities" can reorder this matrix, if required.

Three routines in the IMSL MATH/LIBRARY can be used for fitting the multiple linear regression model. Routine **LSQRR** (IMSL MATH/LIBRARY) computes the fit via the Householder QR decomposition. Routine **LSBRR** (IMSL MATH/LIBRARY) computes the fit via iterative refinement. Routine **LSVRR** (IMSL MATH/LIBRARY) computes the singular value decomposition of a matrix. Routines **LSQRR** and **LSBRR** use the regressors and dependent variable as two input arrays. Routine **LSVRR** computes the singular value decomposition of the matrix of regressors, from which the regression coefficients can be obtained. Kennedy and Gentle (1980, section 8.1) discuss some of the computational advantages and disadvantages of the various methods for least-squares computations.

No Intercept Model

Several routines provide the option for excluding the intercept from a model. In most practical applications, the intercept should be included in the model. For routines that use the sums of squares and crossproducts matrix as input, the no-intercept case can be handled by using the raw sums of squares and crossproducts matrix as input in place of the corrected sums of squares and crossproducts. The raw sum of squares and crossproducts matrix can be computed as$(x_1, x_2, \ldots, x_k, y)^T(x_1, x_2, \ldots, x_k, y)$ using the matrix multiplication routine **MXTXF** (IMSL MATH/LIBRARY).

Variable Selection

Variable selection can be performed by RBEST, which does all best subset regressions, or by RSTEP, which does stepwise regression. In either case, the sum of squares and crossproducts matrix must first be formed. The method used by **RBEST** is generally preferred over that used by **RSTEP** because **RBEST** implicitly examines all

possible models in the search for a model that optimizes some criterion while stepwise does not examine all possible models. However, the computer time and memory requirements for **RBEST** can be much greater than that for **RSTEP** when the number of candidate variables is large.

Two utility routines **GSWEP** and **RSUBM** are provided also for variable selection. Routine **GSWEP** performs a generalized sweep of a nonnegative define matrix. Routine **RSUBM** can be invoked after either **GSWEP** or **RSTEP** in order to extract the symmetric submatrix whose rows and columns have been swept, i.e., whose rows and columns have entered the stepwise model. Routines **GSWEP** and **RSUBM** can be invoked prior to **RBEST** in order to force certain variables into all the models considered by **RBEST**.

Polynomial Model

The polynomial model is

$$y_i = \beta_o + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_k x_i^k + \varepsilon_i \quad i = 1, 2, \ \ldots, n$$

where the observed values of the y_i's constitute the responses or values of the dependent variable, the x_i's are the settings of the independent (explanatory) variables, β_0, β_1, ..., β_k are the regression coefficients, and the ε_i's are independently distributed normal errors each with mean zero and variance σ^2.

Routine **RCURV** fits a specified degree polynomial. Routine **RPOLY** determines the degree polynomial to fit and analyzes this model. If only a decomposition of sum of squares for first, second, ..., *k*-th degree effects in a polynomial model is required, either **RCURV** or the service routine **RFORP** can be used to compute this decomposition. The other service routines (**RSTAP**, **RCASP**, **OPOLY**) can be used to perform other parts of the full analysis.

Multivariate General Linear Model

Routines for the multivariate general linear model use the model

$$Y = XB + \varepsilon$$

where Y is the $n \times q$ matrix of responses, X is the $n \times p$ matrix of regressors, B is the $p \times q$ matrix of regression coefficients, and ε is the $n \times q$ matrix of errors whose q-dimensional rows are identically and independently distributed multivariate normal with mean vector 0 and variance-covariance matrix Σ.

Specification of X for the General Linear Model

Variables used in the general linear model are either continuous or classification variables. Typically, multiple regression models use continuous variables, whereas analysis of variance models use classification variables. Although the notation used to specify analysis of variance models and multiple regression models may look quite different, the models are essentially the same. The term *general linear model* emphasizes that a common notational scheme is used for specifying a model that may contain both continuous and classification variables.

A general linear model is specified by its effects (sources of variation). We refer to an effect as a single variable or a product of variables. (The term *effect* is often used in a narrower sense, referring only to a single regression coefficient.) In particular, an effect is composed of one of the following:

1. a single continuous variable
2. a single classification variable
3. several different classification variables
4. several continuous variables, some of which may be the same
5. continuous variables, some of which may be the same, and classification variables, which must be distinct

Effects of the first type are common in multiple regression models. Effects of the second type appear as main effects in analysis of variance models. Effects of the third type appear as interactions in analysis of variance models. Effects of the fourth type appear in polynomial models and response surface models as powers and crossproducts of some basic variables. Effects of the fifth type appear in one-way analysis of covariance models as regression coefficients that indicate lack of parallelism of a regression function across the groups.

The specification of a general linear model is through arguments `INTCEP`, `NCLVAR`, `INDCL`, `NEF`, `NVEF`, and `INDEF`, whose meanings are as follows:

INTCEP — Intercept option. (Input)

`INTCEP`	**Action**
0	An intercept is not in the model.
1	An intercept is in the model.

NCLVAR — Number of classification variables. (Input)

INDCL — Index vector of length `NCLVAR` containing the column numbers of `X` that are the classification variables. (Input)

NEF — Number of effects (sources of variation) in the model excluding error. (Input)

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input)

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(NEF). (Input)
The first NVEF(1) elements give the column numbers of *X* for each variable in the first effect; the next NVEF(2) elements give the column numbers for each variable in the second effect; and so on. The last NVEF(NEF) elements give the column numbers for each variable in the last effect.

Suppose the data matrix has as its first 4 columns two continuous variables in columns 1 and 2 and two classification variables in columns 3 and 4. The data might appear as follows:

11.23	1.23	1.0	5.0
12.12	2.34	1.0	4.0
12.34	1.23	1.0	4.0
4.34	2.21	1.0	5.0
5.67	4.31	2.0	4.0
4.12	5.34	2.0	1.0
4.89	9.31	2.0	1.0
9.12	3.71	2.0	1.0

Each distinct value of a classification variable determines a level. The classification variable in column 3 has two levels. The classification variable in column 4 has three levels. (Integer values are recommended, but not required, for values of the classification variables. If real numbers are used, the values of the classification variables corresponding to the same level must be identical.) Some examples of regression functions and their specifications are as follows:

	INTCEP	NCLVAR	INDCL	NEF	NVEF	INDEF
$\beta_0 + \beta_1 x_1$	1	0		1	1	1
$\beta_0 + \beta_1 x_1 + \beta_2 x_1^2$	1	0		2	1,2	1,1,1
$\mu + \alpha_l$	1	1	3	1	1	3
$\mu + \alpha_i + \beta_j + \gamma_{ij}$	1	2	3,4	3	1,1,2	3,4,3,4
μ_{ij}	0	2	3,4	1	2	3,4
$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$	1	0		3	1,1,2	1,2,1,2
$\mu + \alpha_i + \beta x_{1i} + \beta_i x_{1i}$	1	1	3	3	1,1,2	3,1,1,3

Routines for Fitting the Model

Routine RGLM fits a multivariate general linear model. If the data set is too large to be stored in a single array, RGLM is designed so that multiple invocations can be made. In this case, one or more rows of the entire data set can be input at each invocation. Alternatively, the data set can be input all at once in a single array. Index vectors are used to specify the column numbers of the data matrix used as classification variables, effects, and dependent variables. This is useful if several models with different effects need to be fit from the same data matrix.

Routine RLEQU can be called after RGIVN or RGLM to impose linear equality restrictions $AB = Z$ on the regression parameters. RLEQU checks consistency of the restrictions. Routine RLEQU is useful for fitting spline functions where restrictions on the regression parameters arise from continuity and differentiability conditions on the regression function.

Routine RLEQU can be used to test the multivariate general linear hypothesis $AB = Z$ by fitting the restricted model after the full model is fit. The additional degrees of freedom for error (and the additional sum of squares and crossproducts for error) gained in the restricted model can be used for computing a test statistic. However, a more efficient approach for computing the sum of squares and crossproducts for a multivariate general linear hypothesis is provided by RHPSS. See the next section entitled "Multivariate General Linear Hypothesis" for a brief description of the problem and related routines.

Two utility routines GCLAS and GRGLM are provided to determine the values of the classification variables and then to use those values and the specified general linear model to generate the regressors in the model. These routines would not be required if you use RGLM to fit the model since RGLM does this automatically. However, if other routines in this chapter are used that require the actual regressors and not the classification variables, then these routines could be used.

Linear Dependence and the *R* Matrix

Linear dependence of the regressors frequently arises in regression models—sometimes by design and sometimes by accident. The routines in this chapter are designed to handle linear dependence of the regressors, i.e., the $n \times p$ matrix X (the matrix of regressors) in the general linear model can have rank less than p. Often, the models are referred to as nonfull rank models.

As discussed in Searle (1971, Chapter 5) some care must be taken to use correctly the results of the fitted nonfull rank regression model for estimation and hypothesis testing. In the nonfull rank case, not all linear combinations of the regression coefficients can be estimated. Those linear combinations that can be estimated are called "estimable functions." If routines in this chapter are used to attempt to estimate linear combinations that cannot be estimated, error messages are issued. A good general discussion of estimable functions is given by Searle (1971, pages 180-188).

The check used by routines in this chapter for linear dependence is sequential. The j-th regressor is declared linearly dependent on the preceding $j - 1$ regressors if

$$\sqrt{1 - R^2_{j\cdot 1,2,\ldots j-1}}$$

is less than or equal to **TOL**. Here, $R_{j\cdot 1,\ 2,\ \ldots,\ j-1}$ is the multiple correlation coefficient of the *j*-th regressor with the first *j* - 1 regressors. Also, **TOL** is a tolerance that must be input by the user. When a routine declares the *j*-th regressor to be linearly dependent on the first *j* - 1 regressors, the *j*-th regression coefficient is set to zero. Essentially, this removes the *j*-th regressor from the model.

The reason a sequential check is used is that frequently practitioners include the variables that they prefer to remain in the model first. Also, the sequential check is based on many of the computations already performed as this does not degrade the overall efficiency of the routines. There is no perfect test for linear dependence when finite precision arithmetic is used. The input of the tolerance **TOL** allows the user some control over the check for linear dependence. If you know your model is full rank, you can input **TOL** = 0.0. However, generally **TOL** should be input as approximately 100 times the machine epsilon. The machine epsilon is **AMACH**(4) in single precision and **DMACH**(4) in double precision. (See routines **AMACH** and **DMACH** in Reference Material)

Routines in this chapter performing least squares are based on *QR* decomposition of *X* or on a Cholesky factorization R^TR of X^TX. Maindonald (1984, chapters 1-5) discusses these methods extensively. The *R* matrix used by the regression routines is taken to be a $p \times p$ upper triangular matrix, i.e., all elements below the diagonal are zero. The signs of the diagonal elements of *R* are used as indicators of linearly dependent regressors and as indicators of parameter restrictions imposed by fitting a restricted model. The rows of *R* can be partitioned into three classes by the sign of the corresponding diagonal element:

1. A positive diagonal element means the row corresponds to data.
2. A negative diagonal element means the row corresponds to a linearly independent restriction imposed on the regression parameters by *AB* = *Z* in a restricted model.
3. A zero diagonal element means a linear dependence of the regressors was declared. The regression coefficients in the corresponding row of $\hat{B}$ are set to zero. This represents an arbitrary restriction which is imposed to obtain a solution for the regression coefficients. The elements of the corresponding row of *R* are also set to zero.

Multivariate General Linear Hypothesis

Routine **RHPSS** computes the matrix of sums of squares and crossproducts for the general linear hypothesis *H B U* = *G* for the multivariate general linear model *Y* = *XB* + **ε** with possible linear equality restrictions *AB* = *Z*. The *R* matrix and $\hat{B}$ from the routines that fit the model are required for input to **RHPSS**.

The rows of *H* must be linear combinations of the rows of *R*, i.e., *H B* = *G* must be completely testable. If the hypothesis is not completely testable, routine **CESTI** can be used to construct an equivalent completely testable hypothesis.

Routine **RHPTE** computes several test statistics and approximate *p*-values for the multivariate general linear hypothesis. The test statistics computed included are Wilks' lambda, Roy's maximum root, Hotelling's trace, and Pillai's trace. Seber (1984, pages 409-416) and Morrison (1976, pages 222-224) discuss the procedures and compare the test statistics. The error sum of squares and crossproducts matrix (**SCPE**) output from the fit of the model is required for input to **RHPTE**. In addition, the hypothesis sum of squares and crossproducts matrix (**SCPH**), which can be computed using **RHPSS**, is required for input to **RHPTE**.

Nonlinear Regression Model

The nonlinear regression model is

$$y_i = f\left(x_i;\theta\right) + \varepsilon_i \quad i = 1, 2, \ \ldots, n$$

where the observed values of the y_i's constitute the responses or values of the dependent variable, the x_i's are the known vectors of values of the independent (explanatory) variables, f is a known function of an unknown regression parameter vector $\boldsymbol{\theta}$, and the $\boldsymbol{\varepsilon}_i$'s are independently distributed normal errors each with mean zero and variance $\boldsymbol{\sigma}^2$.

Routine **RNLIN** performs the least-squares fit to the data for this model. The routine **RCOVB** can be used to compute the large sample variance-covariance matrix of the estimated nonlinear regression parameters from the output of **RNLIN**.

Weighted Least Squares

Routines throughout the chapter generally allow weights to be assigned to the observations. The argument **IWT** is used throughout to specify the weighting option. (**IWT** = 0 means ordinary least squares; a positive **IWT** means weighted least squares with weights in column **IWT** of the data set.) All of the weights must be nonnegative. For routines requiring a sum of squares and crossproducts matrix for input, a weighted analysis can be performed by using as input a weighted sum of squares and crossproducts matrix. Routine **CORVC** in Chapter 3, "Correlation" can compute the required weighted sum of squares and crossproducts matrix.

Computations that relate to statistical inference, e.g., *t* tests, *F* tests, and confidence intervals, are based on the multiple regression model except that the variance of $\boldsymbol{\varepsilon}_i$ is assumed to equal $\boldsymbol{\sigma}^2$ (or $\boldsymbol{\Sigma}$ in the multivariate case) times the reciprocal of the corresponding weight.

If a single row of the data matrix corresponds to n_i observations, the argument **IFRQ** can be used to specify the frequency option. **IFRQ** = 0 means that for all rows, $n_i = 1$; a positive **IFRQ** means the frequencies are entered into column **IFRQ** of the data matrix. Degrees of freedom for error are affected by frequencies, but are unaffected by weights.

Summary Statistics

Summary statistics for a single dependent variable are computed by several routines in the regression chapter. The routines RONE, RLSE, RSTEP, and RPOLY output some summary statistics with the fit of the model. For additional summary statistics, the routines RSTAT and RSTAP can be used.

Routine RSTAT can be used to compute and print statistics related to a regression for each of the q dependent variables fitted by RGIVN, RGLM, RLEQU, or RCOV. Routine RSTAT computes summary statistics that include the model analysis of variance table, sequential sums of squares and
F-statistics, coefficient estimates, estimated standard errors, t-statistics, variance inflation factors, and estimated variance-covariance matrix of the estimated regression coefficients. If only the variance-covariance matrix of the estimated regression coefficients in needed, routine RCOVB can be used.

The summary statistics are computed under the model $y = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors with rank(X) = r, $\boldsymbol{\beta}$ is the $p \times 1$ vector of regression coefficients, and $\boldsymbol{\varepsilon}$ is the $n \times 1$ vector of errors whose elements are independently normally distributed with mean 0 and variance σ^2/w_i.

Given the results of a weighted least-squares fit of this model (with the w_i's as the weights), most of the computed summary statistics are output in the following variables:

AOV — a one-dimensional array usually of length 15. In RSTEP, AOV is of length 13 because the last two elements of the array cannot be computed from the input. The array contains statistics related to the analysis of variance. The sources of variation examined are the regression, error, and total. The first 10 elements of AOV and the notation frequently used for these is described in the following table:

Source of Variation	Degrees of Freedom	Sum of Squares	Mean Square	*F*	*p*-value
Regression	DFR=A0V(1)	SSR=A0V(4)	MSR=A0V(7)	A0V(9)	A0V(10)
Error	DFE=A0V(2)	SSE=A0V(5)	s^2 = A0V(8)		
Total	DFT=A0V(3)	SST=A0V(6)			

In the case an intercept is indicated (INTCEP = 1), the total sum of squares is the sum of squares of the deviations of y_i from its (weighted) mean

$$\overline{y}$$

— the so-called *corrected total sum of squares*, it is denoted by

$$SST = \sum_{i=1}^{n} w_i \left(y_i - \overline{y} \right)^2$$

In the case an intercept is not indicated (**INTCEP** = 0), the total sum of squares is the sum of squares of y_i —the so-called *corrected total sum of squares*, it is denoted by

$$SST = \sum_{i=1}^{n} w_i y_i^2$$

The error sum of squares is given by

$$SSE = \sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i \right)^2$$

The error degrees of freedom is defined by

$$\mathrm{DFE} = n - r$$

The estimate of σ^2 is given by

$$s^2 = \mathrm{SSE}/\mathrm{DFE}$$

which is the error mean square.

The computed *F* statistic for the null hypothesis $H_0 : \beta_1 = \beta_2 = \ldots = \beta_k = 0$ versus the alternative that at least one coefficient is nonzero is given by

$$F = \mathrm{MSR}/s^2$$

The *p*-value associated with the test is the probability of an *F* larger than that computed under the assumption of the model and the null hypothesis. A small *p*-value (less that 0.05) is customarily used to indicate that there is sufficient evidence from the data to reject the null hypothesis.

The remaining 5 elements in **AOV** frequently are displayed together with the actual analysis of variance table. The quantities *R*-squared (R^2 = **AOV**(11)) and adjusted *R*-squared

$$R_a^2 = \mathrm{AOV}\left(12 \right)$$

are expressed as a percentage and are defined by

$$R^2 = 100(\mathrm{SSR}/\mathrm{SST}) = 100(1 - \mathrm{SSE}/\mathrm{SST})$$

$$R_a^2 = 100\max\left\{0,1 - \frac{s^2}{\mathrm{SST}/\mathrm{DFT}}\right\}$$

The square root of s^2(s = **A0V**(13)) is frequently referred to as the estimated standard deviation of the model error.

The overall mean of the responses

$$\overline{y}$$

is output in (**A0V**(14)).

The coefficient of variation (CV = **A0V**(15)) is expressed as a percentage and is defined by

$$\mathrm{CV} = 100s/\overline{y}$$

COEF — a two dimensional array containing the regression coefficient vector

$$\hat{\beta}$$

as one column and associated statistics (including the estimated standard error, *t* statistic and *p*-value) in the remaining columns.

SQSS — a two dimensional array containing sequential sums of squares as one column and associated statistics (including degrees of freedom, *F* statistic, and *p*-value) in the remaining columns.

COVB — the estimated variance-covariance matrix of the estimated regression coefficients.

Tests for Lack of Fit

Tests for lack of fit are computed for simple linear regression by **RONE**, for the polynomial regression by routines **RPOLY** and **RSTAP** and for multiple regression by routines **RLOFE** and **RLOFN**.

In the case of polynomial regression, the two-dimensional output array **TLOF** contains the lack of fit *F* tests for each degree polynomial 1, 2, ..., *k*, that is fit to the data. These tests are useful for indicating the degree of the polynomial required to fit the data well.

In the case of simple and multiple regression, the one-dimensional output array **TESTLF** of length 10 contains the analysis of variance table for the test of lack of fit. Two routines **RLOFE** and **RLOFN** can be used to compute a test for lack of fit. Routine **RLOFE** requires exact replicates of the independent variables, i.e., there must be at least two cases in the data set that have the same settings of all the independent variables, while **RLOFN** does not require exact replicates. Customarily, one would require there to be several sets of duplicate settings of the independent variables in order to use **RLOFE**.

For **RLOFE**, the 10 elements of **TESTLF** and the notation frequently used for these is described in the following table:

Source of Variation	Degrees of Freedom	Sum of Squares	Mean Square	*F*	*p*-value
Lack of Fit	TESTLF(1)	TESTLF(4)	TESTLF(7)	TESTLF(9)	TESTLF(10)
Error	DFPE = TESTLF(2)	SSPE = TESTLF(5)	TESTLF(8)		
Pure Error	DFE = TESTLF(3)	SSE = TESTLF(6)			

For **RLOFN**, the 10 elements of **TESTLF** are similar to those in the previous table. However, since there may not be exact replicates in the data, the data are grouped into sets of near replicates. Then, instead of computing a pure error (or within) sum of squares using a one-way analysis of variance model, an expanded one-way analysis of covariance model using the clusters of near replicates as the groups is computed. The error from this expanded model replaces the pure error in the preceding table in order to compute an exact *F* test for lack of fit conditional on the selected clusters.

Diagnostics for Individual Cases

Diagnostics for individual cases (observations) are computed by several routines in the regression chapter. Routines `RONE`, and `RPOLY` output diagnostics for individual cases with the fit. If the fit of the model is done by other routines, `RCASE` and `RCASP` can be used to compute the diagostics.

Routine **RCASE** computes confidence intervals and diagnostics for individual cases in the data matrix. The cases can be stored in a single data matrix or multiple invocations can be made in which one or more rows of the entire data set are input at any one time. Statistics computed by **RCASE** include predicted values, confidence intervals, and diagnostics for detecting outliers and cases that greatly influence the fitted regression.

If not all of the statistics computed by **RCASE** are needed, `ROTIN` can be used to obtain some of the statistics.

The diagnostics are computed under the model $y = X\,\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors with rank(X) = r, $\boldsymbol{\beta}$ is the $p \times 1$ vector of regression coefficients, and $\boldsymbol{\varepsilon}$ is the $n \times 1$ vector of errors whose elements are independently normally distributed with mean 0 and variance σ^2/w_i.

Given the results of a weighted least-squares fit of this model (with the w_i's as the weights), the following five diagnostics are computed: (1) leverage, (2) standardized residual, (3) jackknife residual, (4) Cook's distance, and (5) DFFITS. These diagnostics are stored in the FORTRAN matrix **CASE**. The definition of these terms is given in the discussion that follows:

Let x_i be a column vector containing the elements of the *i*-th row of *X*. A case could be unusual either because of x_i or because of the response y_i. The *leverage* h_i is a measure of unusualness of the x_i. The leverage is defined by

$$h_i = \left[x_i^T \left(X^T W X \right)^- x_i \right] w_i$$

where $W = \text{diag}(w_1, w_2, \ldots, w_n)$ and $(X^T W X)^-$ denotes a generalized inverse of $X^T WX$. The average value of the h_i's is r/n. Regression routines declare x_i unusual if $h_i > 2r/n$. A row label **X** is printed beside a case that is unusual because of of x_i. Hoaglin and Welsch (1978) call a data point highly influential (i.e., a leverage point) when this occurs.

Let e_i denote the residual

$$y_i - \hat{y}_i$$

for the *i*-th case. The estimated variance of e_i is $(1 - h_i)s^2/w_i$ where s^2 is the residual mean square from the fitted regression. The *i*-th *standardized residual* (also called the internally studentized residual) is by definition

$$r_i = e_i \sqrt{\frac{w_i}{s^2 \left(1 - h_i \right)}}$$

and r_i follows an approximate standard normal distribution in large samples.

The *i*-th *jackknife residual* or *deleted residual* involves the difference between y_i and its predicted value based on the data set in which the *i*-th case is deleted. This difference equals $e_i/(1 - h_i)$. The jackknife residual is obtained by standardizing this difference. The residual mean square for the regression in which the *i*-th case is deleted is

$$s_i^2 = \frac{\left(n - r \right) s^2 - w_i e_i^2 / \left(1 - h_i \right)}{n - r - 1}$$

The jackknife residual is defined to be

$$t_i = e_i \sqrt{\frac{w_i}{s_i^2 \left(1 - h_i \right)}}$$

and t_i follows a *t* distribution with $n - r - 1$ degrees of freedom. The regression routines declare y_i unusual (an outlier) if a jackknife residual greater than 2.0 in absolute value is computed. A row label **Y** is printed beside a case that is unusual because of y_i.

Cook's distance for the *i*-th case is a measure of how much an individual case affects the estimated regression coefficients. It is given as

$$D_i = \frac{w_i h_i e_i^2}{r s^2 \left(1 - h_i\right)^2}$$

Weisberg (1985) states that if D_i exceeds the 50-th percentile of the $F(r, n - r)$ distribution, it should be considered large. (This value is about 1. This statistic does not have an F distribution.)

DFFITS, like Cook's distance, is also a measure of influence. For the i-th case, DFFITS is computed by the formula

$$\text{DFFITS}_i = e_i \sqrt{\frac{w_i h_i}{s_i^2 \left(1 - h_i\right)^2}}$$

Hoaglin and Welsch (1978) suggest that DFFITS_i is greater than

$$2\sqrt{r/n}$$

is large.

Transformations

Transformations of the independent variables are sometimes useful in order to satisfy the regression model. The inclusion of squares and crossproducts of the variables

$$\left(x_1,\ x_2,\ x_1^2,\ x_2^2,\ x_1 x_2\right)$$

is often needed. Logarithms of the independent variables are also often used. (See Draper and Smith, 1981, pages 218-222, Box and Tidwell, 1962, Atkinson, 1985, pages 177-180, Cook and Weisberg, 1982, pages 78-86.)

When the responses are described by a nonlinear function of the parameters, a transformation of the model equation can often be selected so that the transformed model is linear in the regression parameters. For example, the exponential model

$$y = e^{\beta_0 + \beta_1 x_1} \varepsilon$$

by taking natural logarithms on both sides of the equation, can be transformed to a model that satisfies the linear regression model provided the $\boldsymbol{\varepsilon}_i$'s have a log normal distribution (Draper and Smith, pages 222-225).

When the responses are nonnormal and their distribution is known, a transformation of the responses can often be selected so that the transformed responses closely satisfy the regression model assumptions. The square root transformation for counts with a Poisson distribution and the arc-sine transformation for binomial proportions are common examples (Snedecor and Cochran, 1967, pages 325-330, Draper and Smith, pages 237-239).

If the distribution of the responses is not known, the data can be used to select a transformation so that the transformed responses may more closely obey the regression model. For a positive response variable $y > 0$, the family of power transformations indexed by λ

$$y^{(\lambda)} = \begin{cases} \frac{y^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \ln y & \text{if } \lambda = 0 \end{cases}$$

and generalizations of this family are useful. Routine **BCTR** (See Chapter 8, "Time Series Analysis and Forecasting") can be used to perform the transformation. A method to estimate and to compute an approximate test for $\lambda = 1$ is given by Atkinson (1973). Also, Atkinson (1986) discusses transformation deletion statistics for computing the estimate and test leaving out a single observation since the evidence for a transformation of the response may sometimes depend crucially on one or a few observations.

Alternatives to Least Squares

The method of least squares has desirable characteristics when the errors are normally distributed, e.g., a least-squares solution produces maximum likelihood estimates of the regression parameters. However, when errors are not normally distributed, least squares may yield poor estimators. The least absolute value (LAV, L_1) criterion yields the maximum likelihood estimate when the errors follow a Laplace distribution. Routine **RLAV** is often used when the errors have a heavy tailed distribution or when a fit is needed that is resistant to outliers.

A more general approach, minimizing the L_p norm ($p \geq 1$), is given by routine **RLLP**. Although the routine requires about 30 times the **CPU** time for the case $p = 1$ than would the use of **RLAV**, the generality of **RLLP** allows the user to try several choices for $p \geq 1$ by simply changing the input value of p in the calling program. The **CPU** time decreases as p gets larger. Generally, choices of p between 1 and 2 are of interest. However, the L_p norm solution for values of p larger than 2 can also be computed.

The minimax (LMV, L_∞, Chebyshev) criterion is used by **RLMV**. Its estimates are very sensitive to outliers, however, the minimax estimators are quite efficient if the errors are uniformly distributed.

Routine **PLSR** provides a fourth alternative useful when there are many inter-related regression variables and relatively few observations. **PLSR** finds linear combinations of the predictor variables that have highest covariance with Y.

Missing Values

NaN (not a number) is the missing value code used by the regression routines. Use function **AMACH**(6) (or function **DMACH**(6) with double precision regression routines) to retrieve NaN. (See the section Machine-Dependent Constants in Reference Material.) Any element of the data matrix that is missing must be set to **AMACH**(6) (or

DMACH(6) for double precision). In fitting regression models, any row of the data matrix containing NaN for the independent, dependent, weight, or frequency variables is omitted from the computation of the regression parameters.

Often predicted values and confidence intervals are desired for combinations of settings of the independent variables not used in computing the regression fit. This can be accomplished by including additional rows in the data matrix. These additional rows should contain the desired settings of the independent variables along with the responses set equal to NaN. The cases with NaN will not be used in determining the estimates of the regression parameters, and a predicted value and confidence interval will be computed from the given settings of the independent variables.

RLINE

Fits a line to a set of data points using least squares.

Required Arguments

XDATA — Vector of length NOBS containing the *x*-values. (Input)

YDATA — Vector of length NOBS containing the *y*-values. (Input)

B0 — Estimated intercept of the fitted line. (Output)

B1 — Estimated slope of the fitted line. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (XDATA,1).

STAT — Vector of length 12 containing the statistics described below. (Output)

I	STAT(I)
1	Mean of XDATA
2	Mean of YDATA
3	Sample variance of XDATA
4	Sample variance of YDATA
5	Correlation
6	Estimated standard error of B0
7	Estimated standard error of B1
8	Degrees of freedom for regression
9	Sum of squares for regression
10	Degrees of freedom for error
11	Sum of squares for error
12	Number of (*x*, *y*) points containing NaN (not a number) as either the *x* or *y* value

FORTRAN 90 Interface

Generic: CALL RLINE (XDATA, YDATA, B0, B1 [, ...])

Specific: The specific interface names are S_RLINE and D_RLINE.

FORTRAN 77 Interface

Single: CALL RLINE (NOBS, XDATA, YDATA, B0, B1, STAT)

Double: The double precision name is DRLINE.

Description

Routine RLINE fits a line to a set of (*x*, *y*) data points using the method of least squares. Draper and Smith (1981, pages 1-69) discuss the method. The fitted model is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

where $\hat{\beta}_0$ (stored in B0) is the estimated intercept and $\hat{\beta}_1$ (stored in B1) is the estimated slope. In addition to the fit, RLINE produces some summary statistics, including the means, sample variances, correlation, and the error (residual) sum of squares. The estimated standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$ are computed under the simple linear regression model. The errors in the model are assumed to be uncorrelated and with constant variance.

If the *x* values are all equal, the model is degenerate. In this case, **RLINE** sets $\hat{\beta}_1$ to zero and $\hat{\beta}_0$ to the mean of the *y* values.

Comments

Informational error

Type	Code	Description
4	1	Each (x, y) point contains NaN (not a number). There are no valid data.

Example

This example fits a line to a set of data discussed by Draper and Smith (1981, Table 1.1, pages 9-33). The response *y* is the amount of steam used per month (in pounds), and the independent variable *x* is the average atmospheric temperature (in degrees Fahrenheit).

```
      USE RLINE_INT
      USE UMACH_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=25)
!
      INTEGER    NOUT
      REAL       B0, B1, STAT(12), XDATA(NOBS), YDATA(NOBS)
      CHARACTER  CLABEL(13)*15, RLABEL(1)*4
!
      DATA XDATA/35.3, 29.7, 30.8, 58.8, 61.4, 71.3, 74.4, 76.7, 70.7, &
          57.5, 46.4, 28.9, 28.1, 39.1, 46.8, 48.5, 59.3, 70.0, 70.0, &
          74.5, 72.1, 58.1, 44.6, 33.4, 28.6/
      DATA YDATA/10.98, 11.13, 12.51, 8.4, 9.27, 8.73, 6.36, 8.5, &
          7.82, 9.14, 8.24, 12.19, 11.88, 9.57, 10.94, 9.58, 10.09, &
          8.11, 6.83, 8.88, 7.68, 8.47, 8.86, 10.36, 11.08/
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Mean of X', 'Mean of Y', &
          'Variance X', 'Variance Y', 'Corr.', 'Std. Err. B0', &
          'Std. Err. B1', 'DF Reg.', 'SS Reg.', 'DF Error', &
          'SS Error', 'Pts. with NaN'/
!
      CALL RLINE (XDATA, YDATA, B0, B1, STAT=STAT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) B0, B1
99999 FORMAT (' B0 = ', F7.2, '  B1 = ', F9.5)
      CALL WRRRL ('%/STAT', STAT, RLABEL, CLABEL, 1, 12, 1, &
                  FMT='(12W10.4)')
!
      END
```

Output

```
B0 =   13.62  B1 =  -0.07983

                                        STAT
Mean of X   Mean of Y  Variance X  Variance Y       Corr.  Std. Err. B0
     52.6       9.424       298.1       2.659     -0.8452        0.5815

Std. Err. B1     DF Reg.     SS Reg.    DF Error    SS Error  Pts. with NaN
     0.01052           1       45.59          23       18.22             0
```

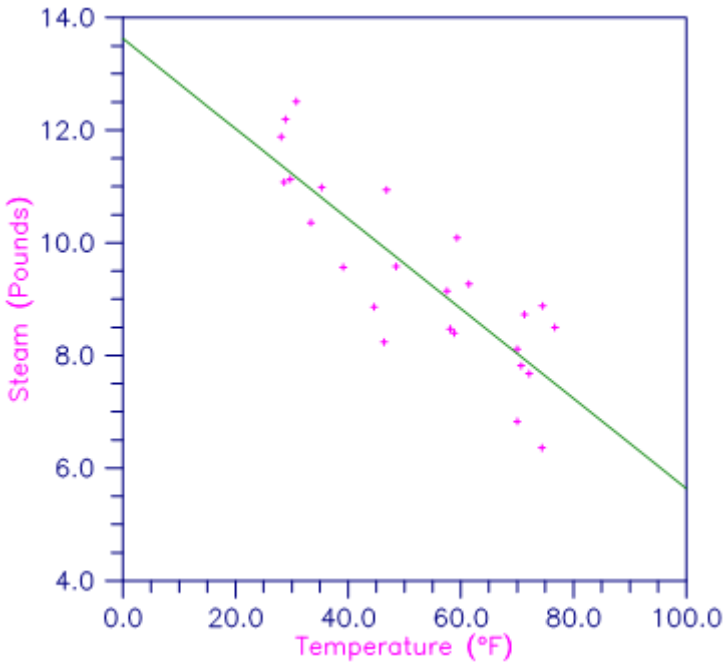


Figure 1, Plot of the Data and the Least Squares Line

RONE

Analyzes a simple linear regression model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

IND — Column number IND of X contains the data for the independent (explanatory) variable. (Input)

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

If INTCEP = 1, the regression and total are corrected for the mean. If INTCEP = 0, the regression and total are not corrected for the mean, and AOV(14) and AOV(15) are set to NaN (not a number).

COEF — INTCEP + 1 by 5 matrix containing statistics relating the regression coefficients. (Output)
If INTCEP = 1, the first row corresponds to the intercept. Row INTCEP + 1 corresponds to the coefficient for the slope. The statistics in the columns are

Col.	Description
1	Coefficient estimate
2	Estimated standard error of the coefficient estimate
3	*t*-statistic for the test that the coefficient is zero
4	*p*-value for the two-sided *t* test
5	Variance inflation factor

COVB — INTCEP + 1 by INTCEP + 1 matrix that is the estimated variance-covariance matrix of the estimated regression coefficients. (Output)

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the model. (Output)

Elem	Description
1	Degrees of freedom for lack of fit
2	Degrees of freedom for pure error
3	Degrees of freedom for error (TESTLF(1) + TESTLF(2))
4	Sum of squares for lack of fit
5	Sum of squares for pure error
6	Sum of squares for error
7	Mean square for lack of fit
8	Mean square for pure error
9	*F* statistic
10	*p*-value

If there are no replicates in the data set, a test for lack of fit cannot be performed. In this case, elements 7, 8, 9, and 10 of TESTLF are set to NaN (not a number).

CASE — NOBS by 12 matrix containing case statistics. (Output)
Columns 1 through 12 contain the following:

Col.	Description
1	Observed response
2	Predicted response
3	Residual
4	Leverage
5	Standardized residual
6	Jackknife residual
7	Cook's distance
8	DFFITS
9, 10	Confidence interval on the mean
11, 12	Prediction interval

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies. If X(I, IFRQ) = 0.0, none of the remaining elements of row I of X are referenced, and updating of statistics is skipped for row I.
Default: IFRQ = 0.

IWT — Weighting option. (Input)

IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.

Default: IWT = 0.

IPRED — Prediction interval option. (Input)

IPRED = 0 means that prediction intervals are computed for a single future response. For positive IPRED, a prediction interval is computed on the average of future responses, and column number IPRED of X contains the number of future responses in each average.

Default: IPRED =0.

CONPCM — Confidence level for two-sided interval estimates on the mean, in percent. (Input)

CONPCM percent confidence intervals are computed, hence, CONPCM must be greater than or equal to 0.0 and less than 100.0. CONPCM often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, where ONECL is greater than or equal to 50.0 and less than 100.0, set CONPCM = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPCM = 95.0.

CONPCP — Confidence level for two-sided prediction intervals, in percent. (Input)

CONPCP percent prediction intervals are computed, hence, CONPCP must be greater than or equal to 0.0 and less than 100.0. CONPCP often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, where ONECL is greater than or equal to 50.0 and less than 100.0, set CONPCP = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPCP = 95.0.

IPRINT — Printing option. (Input)

Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	AOV, COEF, TESTLF, and unusual rows of CASE are printed.
2	AOV, COEF, TESTLF, and unusual rows of CASE are printed. A plot of the data with the regression line is printed.
3	All printing is performed. A plot of the data with the regression line, a plot of the standardized residuals versus the independent variable, and a half-normal probability plot of the standardized residuals are printed.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)

Default: LDCOEF = size (COEF,1).

LDCOVB — Leading dimension of **COVB** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOVB** = size (**COVB**,1).

LDCASE — Leading dimension of **CASE** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCASE** = size (**CASE**,1).

NRMISS — Number of rows of data encountered containing missing values for the independent, dependent, weight, or frequency variables. (Output)
NaN (not a number) is used as the missing value code. Any row of **X** containing NaN as a value of the independent, dependent, weight, or frequency variables is omitted from the computations for fitting the model.

FORTRAN 90 Interface

Generic: `CALL RONE (X, IRSP, IND, AOV, COEF, COVB, TESTLF, CASE [, ...])`
Specific: The specific interface names are **S_RONE** and **D_RONE**.

FORTRAN 77 Interface

Single: `CALL RONE (NOBS, NCOL, X, LDX, INTCEP, IRSP, IND, IFRQ, IWT, IPRED, CONPCM, CONPCP, IPRINT, AOV, COEF, LDCOEF, COVB, LDCOVB, TESTLF, CASE, LDCASE, NRMISS)`
Double: The double precision name is **DRONE**.

Description

Routine **RONE** performs an analysis for the simple linear regression model. In addition to the fit, summary statistics (analysis of variance, *t* tests, lack-of-fit test), and confidence intervals and diagnostics for individual cases are computed. With the printing option, diagnostic plots can also be produced. Draper and Smith (1981, chapter 1) give formulas for many of the statistics computed by **RONE**. For definitions of the case diagnostics (stored in **CASE**), see the "Usage Notes" of this chapter.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2NE/DR2NE**. The reference is:

   `CALL R2NE (NOBS, NCOL, X, LDX, INTCEP, IRSP, IND, IFRQ, IWT, IPRED, CONPCM, CONPCP, IPRINT, AOV, COEF, LDCOEF, COVB, LDCOVB, TESTLF, CASE, LDCASE, NRMISS, IWK, WK)`

The additional arguments are as follows:

IWK — Work vector of length NOBS.

WK — Work vector of length 3 * NOBS.

2. Informational errors

Type	Code	Description
3	5	CONPCM is less than 50.0. Confidence percentages commonly used are 90.0, 95.0, and 99.0.
3	6	CONPCP is less than 50.0. Confidence percentages commonly used are 90.0, 95.0, and 99.0.
4	1	Negative weight encountered.
4	2	Negative frequency encountered.
4	7	Each row of X contains NaN.

Examples

Example 1

This example fits a line to a set of data discussed by Draper and Smith (1981, pages 9-33). The response *y* is the amount of steam used per month (in pounds), and the independent variable *x* is the average atmospheric temperature (in degrees Fahrenheit). The IPRINT = 1 option is selected. Hence, plots are not produced and only unusual cases are printed. Note in the case analysis, with the default page width, the observation number and the associated 12 statistics require two lines of output. (Routine PGOPT, Chapter 19, "Utilities", can be invoked to increase the page width to put all 12 statistics on the same line.) Also note that observation 11 is labeled with a "Y" to indicate an unusual *y* (response). The residual for this case is about 2 standard deviations from zero.

```
      USE RONE_INT

      IMPLICIT   NONE
      INTEGER    INTCEP, LDCASE, LDCOEF, LDCOVB, LDX, NCOEF, NCOL, NOBS
      INTEGER    J
      PARAMETER  (NOBS=25, LDX=25, LDCASE=25, INTCEP=1, NCOEF=INTCEP+1, &
                 LDCOEF=NCOEF, LDCOVB=NCOEF, NCOL=2)
!
      INTEGER    IND, IPRINT, IRSP, NRMISS
      REAL       AOV(15), CASE(LDCASE,12), COEF(LDCOEF,5), CONPCP, &
                 COVB(LDCOVB,NCOEF), TESTLF(10), X(LDX,NCOL)
!
      DATA (X(1,J),J=1,2)  /35.3, 10.98/
      DATA (X(2,J),J=1,2)  /29.7, 11.13/
      DATA (X(3,J),J=1,2)  /30.8, 12.51/
      DATA (X(4,J),J=1,2)  /58.8,  8.40/
      DATA (X(5,J),J=1,2)  /61.4,  9.27/
      DATA (X(6,J),J=1,2)  /71.3,  8.73/
      DATA (X(7,J),J=1,2)  /74.4,  6.36/
```

```
      DATA (X(8,J),J=1,2)  /76.7,  8.50/
      DATA (X(9,J),J=1,2)  /70.7,  7.82/
      DATA (X(10,J),J=1,2) /57.5,  9.14/
      DATA (X(11,J),J=1,2) /46.4,  8.24/
      DATA (X(12,J),J=1,2) /28.9, 12.19/
      DATA (X(13,J),J=1,2) /28.1, 11.88/
      DATA (X(14,J),J=1,2) /39.1,  9.57/
      DATA (X(15,J),J=1,2) /46.8, 10.94/
      DATA (X(16,J),J=1,2) /48.5,  9.58/
      DATA (X(17,J),J=1,2) /59.3, 10.09/
      DATA (X(18,J),J=1,2) /70.0,  8.11/
      DATA (X(19,J),J=1,2) /70.0,  6.83/
      DATA (X(20,J),J=1,2) /74.5,  8.88/
      DATA (X(21,J),J=1,2) /72.1,  7.68/
      DATA (X(22,J),J=1,2) /58.1,  8.47/
      DATA (X(23,J),J=1,2) /44.6,  8.86/
      DATA (X(24,J),J=1,2) /33.4, 10.36/
      DATA (X(25,J),J=1,2) /28.6, 11.08/
!
      IRSP   = 2
      IND    = 1
      CONPCP = 99.0
      IPRINT = 1
      CALL RONE (X, IRSP, IND, AOV, COEF, COVB, TESTLF, CASE, &
                 CONPCP=CONPCP, IPRINT=IPRINT, NRMISS=NRMISS)
!
      END
```

Output

```
 R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 (percent)  R-squared  of Model Error        Mean  Var. (percent)
    71.444     70.202          0.8901       9.424           9.445

                   * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Regression            1       45.59       45.59     57.543    0.0000
 Residual             23       18.22        0.79
 Corrected Total      24       63.82

                * * * Inference on Coefficients * * *
                    Standard                  Prob. of    Variance
 Coef.    Estimate       Error  t-statistic  Larger |t|   Inflation
     1       13.62      0.5815        23.43      0.0000       10.67
     2       -0.08      0.0105        -7.59      0.0000        1.00

                 * * * Test for Lack of Fit * * *
                           Sum of        Mean             Prob. of
 Source             DF     Squares      Square  Overall F  Larger F
 Lack of fit        22       17.40      0.7911      0.966    0.6801
 Pure error          1        0.82      0.8192
 Residual           23       18.22

                         * * * Case Analysis * * *
      Obs.   Observed  Predicted   Residual   Leverage  Std. Res.  Jack Res.
             Cook's D     DFFITS   95.0% CI   95.0% CI   99.0% PI   99.0% PI
 Y      11     8.2400     9.9189    -1.6789     0.0454    -1.9305    -2.0625
               0.0886    -0.4497     9.5267    10.3112     7.3640    12.4739
```

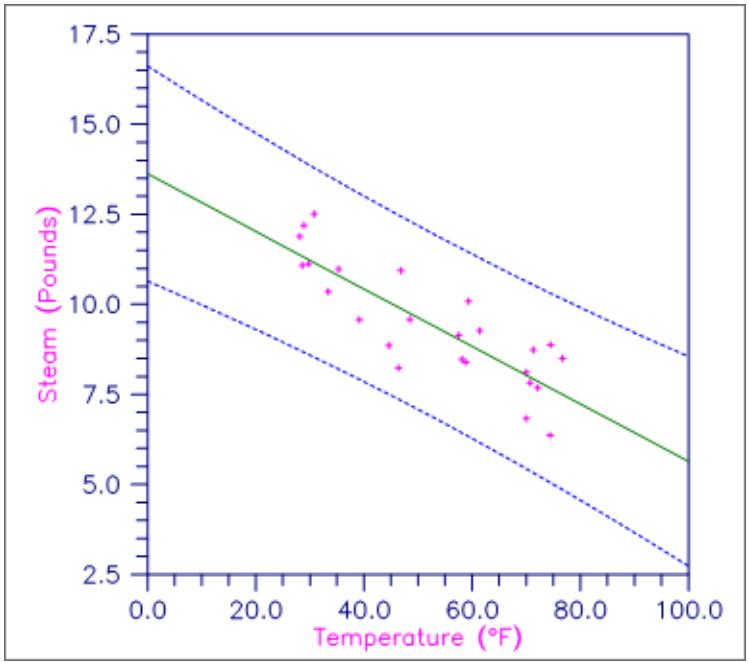


Figure 2, Plot of Line and 99% One-at-a-Time Prediction Intervals

Example 2

This example fits a line to a data set discussed by Draper and Smith (1981, pages 38-40). The data set contains several repeated *x* values in order to assess lack of fit of the straight line. The **IPRINT** = 1 option is selected. Hence, plots are not produced and only unusual cases are printed. Note in the case analysis that observations 1 and 2 are labeled with an "**X**" to indicate an unusual *x* value. Each have leverage 0.1944 that exceeds the average leverage of *p*/*n* = 2/24 by a factor of 2.

```
          USE RONE_INT

          IMPLICIT    NONE
          INTEGER     LDCASE, LDCOEF, LDCOVB, LDX, NCOEF, NCOL, NOBS,J
          INTEGER     INTCEP, NRMISS
          PARAMETER   (INTCEP=1, NCOL=2, NOBS=24, LDCASE=NOBS, LDX=NOBS, &
                       NCOEF=INTCEP+1, LDCOEF=NCOEF, LDCOVB=NCOEF)
     !
          INTEGER     IFRQ, IND, IPRED, IPRINT, IRSP
          REAL        AOV(15), CASE(LDCASE,12),COEF(LDCOEF,5), &
                      COVB(LDCOVB,NCOEF), TESTLF(10), X(LDX,NCOL)
     !
          DATA (X(1,J),J=1,2)  /2.3, 1.3/
          DATA (X(2,J),J=1,2)  /1.8, 1.3/
          DATA (X(3,J),J=1,2)  /2.8, 2.0/
          DATA (X(4,J),J=1,2)  /1.5, 2.0/
```

```
      DATA (X(5,J),J=1,2)  /2.2, 2.7/
      DATA (X(6,J),J=1,2)  /3.8, 3.3/
      DATA (X(7,J),J=1,2)  /1.8, 3.3/
      DATA (X(8,J),J=1,2)  /3.7, 3.7/
      DATA (X(9,J),J=1,2)  /1.7, 3.7/
      DATA (X(10,J),J=1,2) /2.8, 4.0/
      DATA (X(11,J),J=1,2) /2.8, 4.0/
      DATA (X(12,J),J=1,2) /2.2, 4.0/
      DATA (X(13,J),J=1,2) /5.4, 4.7/
      DATA (X(14,J),J=1,2) /3.2, 4.7/
      DATA (X(15,J),J=1,2) /1.9, 4.7/
      DATA (X(16,J),J=1,2) /1.8, 5.0/
      DATA (X(17,J),J=1,2) /3.5, 5.3/
      DATA (X(18,J),J=1,2) /2.8, 5.3/
      DATA (X(19,J),J=1,2) /2.1, 5.3/
      DATA (X(20,J),J=1,2) /3.4, 5.7/
      DATA (X(21,J),J=1,2) /3.2, 6.0/
      DATA (X(22,J),J=1,2) /3.0, 6.0/
      DATA (X(23,J),J=1,2) /3.0, 6.3/
      DATA (X(24,J),J=1,2) /5.9, 6.7/
!
      IRSP = 1
      IND  = 2
      IPRINT = 1
      CALL RONE (X, IRSP, IND, AOV, COEF, COVB, TESTLF, CASE, &
                 IPRINT=IPRINT, NRMISS=NRMISS)
      END
```

Output

```
 R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 (percent)  R-squared  of Model Error       Mean  Var. (percent)
    22.983     19.483          0.9815      2.858          34.34

                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Regression            1        6.32       6.325      6.565    0.0178
 Residual             22       21.19       0.963
 Corrected Total      23       27.52

                  * * * Inference on Coefficients * * *
                     Standard                  Prob. of    Variance
 Coef.    Estimate       Error  t-statistic  Larger |t|   Inflation
     1       1.436      0.5900        2.435      0.0235       8.672
     2       0.338      0.1319        2.562      0.0178       1.000

                   * * * Test for Lack of Fit * * *
                            Sum of        Mean             Prob. of
 Source              DF     Squares      Square  Overall F  Larger F
 Lack of fit         11        8.72       0.793      0.700    0.7183
 Pure error          11       12.47       1.134
 Residual            22       21.19

                        * * * Case Analysis * * *
       Obs.  Observed  Predicted   Residual   Leverage  Std. Res.  Jack Res.
             Cook's D     DFFITS   95.0% CI   95.0% CI   95.0% PI   95.0% PI
 X       1     2.3000     1.8756     0.4244     0.1944     0.4817     0.4731
               0.0280     0.2324     0.9783     2.7730    -0.3489     4.1002
```

X	2	1.8000	1.8756	-0.0756	0.1944	-0.0859	-0.0839
		0.0009	-0.0412	0.9783	2.7730	-0.3489	4.1002
Y	13	5.4000	3.0245	2.3755	0.0460	2.4780	2.8515
		0.1481	0.6264	2.5877	3.4612	0.9426	5.1063
Y	24	5.9000	3.7002	2.1998	0.1537	2.4363	2.7855
		0.5391	1.1873	2.9021	4.4983	1.5138	5.8866

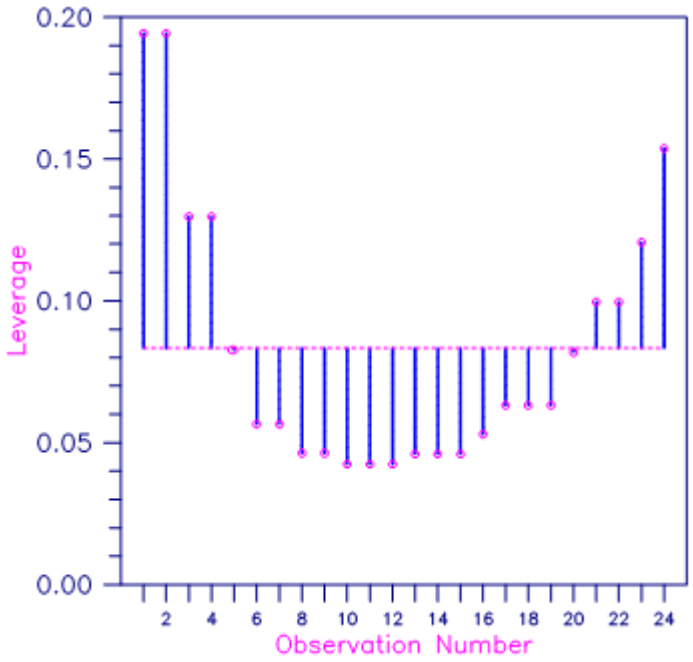


Figure 3, Plot of Leverages h_i and the Average (p/n = 2/24)

RINCF

Performs response control given a fitted simple linear regression model.

Required Arguments

SUMWTF — Sum of products of weights with frequencies from the fitted regression. (Input, if INTCEP = 1)
In the ordinary case when weights and frequencies are all one, SUMWTF equals the number of observations.

DFE — Degrees of freedom for error from the fitted regression. (Input)

B — Vector of length INTCEP + 1 containing a least-squares solution for the intercept and slope. (Input)

INTCEP	Intercept	Slope
0		B(1)
1	B(1)	B(2)

XYMEAN — Vector of length 2 containing the variable means. (Input)
XYMEAN(1) is the independent variable mean. XYMEAN(2) is the dependent variable mean. If INTCEP = 0, XYMEAN is not referenced and can be a vector of length one.

SSX — Sum of squares for the independent variable. (Input)
If INTCEP = 1, SSX is the sums of squares of deviations of the independent variable from its mean. Otherwise, SSX is not corrected for the mean.

S2 — s^2, the estimate of σ^2 from the fitted regression. (Input)

YLOWER — Lower limit for the response. (Input)

YUPPER — Upper limit for the response. (Input)

XLOWER — Lower limit on the independent variable for controlling the response. (Output)

XUPPER — Upper limit on the independent variable for controlling the response. (Output)

Optional Arguments

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

SWTFY0 — S2/SWTFY0 is the estimated variance of the future response (or future response mean) that is to be controlled. (Input)
In the ordinary case, when weights and frequencies are all one, SWTFY0 is the number of observations in the response mean that is to be controlled. SWTFY0 = 0.0 means the true response mean is to be controlled.
Default: SWTFY0 = 0.0.

CONPER — Confidence level for a two-sided response control, in percent. (Input)
CONPER percent limits are computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER often will be 90.0, 95.0, or 99.0. For one-sided control with confidence level ONECL, where ONECL is greater than or equal to 50.0 and less than 100.0, set CONPCM = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL RINCF (SUMWTF, DFE, B, XYMEAN, SSX, S2, YLOWER, YUPPER, XLOWER, XUPPER [, ...])

Specific: The specific interface names are S_RINCF and D_RINCF.

FORTRAN 77 Interface

Single: CALL RINCF (SUMWTF, DFE, INTCEP, B, XYMEAN, SSX, S2, SWTFY0, CONPER, YLOWER, YUPPER, XLOWER, XUPPER)

Double: The double precision name is DRINCF.

Description

Routine RINCF estimates settings of the independent variable that restrict, at a specified confidence percentage, the average of *k* randomly drawn responses to a given acceptable range (or the true mean response to a given acceptable range), using a fitted simple linear regression model. The results of routine RLINE or RONE can be used for input into RINCF. The simple linear regression model is assumed:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad i = 1, 2, \ldots, n + k$$

where the ε_i's are independently distributed normal errors with mean zero and variance σ^2/w_i. Here, n is the total number of observations used in the fit of the line, i.e., n = **DFE** + **INTCEP** + 1. Also, k is the number of additional responses whose average is to be restricted to the specified range. The w_i's are the weights.

The methodology is based on Graybill (1976, pages 280-283). The estimate of σ^2, s^2 (stored in **S2**), is the usual estimate of σ^2 from the fitted regression based on the first n observations. First, a test of the hypothesis $H_0 : \beta_1 = 0$ vs. $H_a : \beta_1 \neq 0$ at level α = 1 - **CONPER**/100 is performed. If H_0 is accepted, the model becomes $y_i = \beta_0 + \varepsilon_i$, and limits for x to control the response are meaningless since x is no longer in the model. In this case, a type 4 fatal error is issued. If H_0 is rejected and $\hat{\beta}_1$ is positive, a lower limit (upper limit) for x stored in **XLOWER**(**XUPPER**) is computed for the case where **SWTFY0** is positive by

$$\bar{x} + \frac{\hat{\beta}_1\left(y_0 - \bar{y}\right)}{a} \pm \frac{ts}{a}\left[\frac{a}{\sum_{i=1}^{n} w_i} + \frac{a}{\sum_{i=n+1}^{n+k} w_i} + \frac{\left(y_0 - \bar{y}\right)^2}{\sum_{i=1}^{n} w_i\left(x_i - \bar{x}\right)^2}\right]^{1/2}$$

where y_0 is the value stored in **YLOWER**(**YUPPER**) and where

$$a = \hat{\beta}_1^2 - \frac{t^2 s^2}{\sum_{i=1}^{n} w_i\left(x_i - \bar{x}\right)^2}$$

and t is the 50 + CONPER/2 percentile of the t distribution with DFE degrees of freedom. In the formula, the symbol ± is used to indicate that + is used to compute **XLOWER** with y_0 = **YLOWER**, and - is used to compute **XUPPER** with y_0 = **YUPPER**. If H_0 is rejected and $\hat{\beta}_1$ is negative, a lower limit (upper limit) for x stored in **XLOWER**(**XUPPER**) is computed for the case where **SWTFY0** is positive by a small modification. In particular, the symbol ± is then taken so that + is used to compute **XLOWER** with y_0 = **YUPPER**, and - is used to compute **XUPPER** with y_0 = **YLOWER**. These limits actually have a confidence coefficient less than that specified by **CONPER**.

In the weighted case, which was discussed earlier, the means (stored in **XYMEAN**) and the sum of squares for x (stored in **SSX**) are all weighted. When the variances of the ε_i's are all equal, ordinary least squares must be used, this corresponds to all w_i = 1.

The previous discussion can be generalized to the case where an intercept is not in the model. The necessary modifications are to let $\beta_0 = 0,\ \hat{\beta}_0 = 0$ and to replace the first term under the square root symbol by zero, $\bar{x}$ by zero, and $\bar{y}$ by zero.

In order to restrict the true mean response to a specified range, i.e, when SWTFY0 is zero, the formulas are modified by replacing the second term under the square root symbol with zero.

Comments

Informational errors

Type	Code	Description
4	1	The slope is not significant at the (100 - CONPER) percent level. Control limits cannot be obtained.
4	2	The computed lower limit, XLOWER, exceeds the computed upper limit, XUPPER. No satisfactory settings of the independent variable exist to control the response as specified.

Example

This example estimates the settings of the independent variable that restrict, at 97.5% confidence, the true mean response to a upper bound of -4.623, using a fitted simple linear regression model. The fitted model excludes the intercept term. To accomplish one-sided control, CONPER is set to 100 - 2(100 - 97.5) = 95, and YLOWER is set to an arbitrary value less than YUPPER. The output for XLOWER furnishes the lower bound for *x* necessary to control *y*.

```
          USE RINCF_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    INTCEP
          PARAMETER  (INTCEP=0)
     !
          INTEGER    NOUT
          REAL       B(INTCEP+1), CONPER, DFE, ONECL, S2, SSX, SUMWTF, &
                     SWTFY0, XLOWER, XUPPER, XYMEAN(1), YLOWER, YUPPER
     !
          DATA B/-.079829/
     !
          SUMWTF = 25.0
          DFE    = 24.0
          SSX    = 76323.0
          S2     = 0.7926
          SWTFY0 = 0.0
          ONECL  = 97.5
          CONPER = 100.0 - 2*(100.0-ONECL)
          YUPPER = -4.623
          YLOWER = -9.0
          CALL RINCF (SUMWTF, DFE, B, XYMEAN, SSX, S2, YLOWER, YUPPER,  &
                      XLOWER, XUPPER, INTCEP=INTCEP, CONPER=CONPER)
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) 'XLOWER = ', XLOWER, '  XUPPER = ', XUPPER
          END
```

Output

```
    XLOWER = 63.1747   XUPPER = 104.07
```

RINPF

Performs inverse prediction given a fitted simple linear regression model.

Required Arguments

SUMWTF — Sum of products of weights with frequencies from the fitted regression. (Input, if `INTCEP` = 1)
In the ordinary case when weights and frequencies are all one, `SUMWTF` equals the number of observations used in the fit of the model.

DFS2 — Degrees of freedom for estimate of σ^2. (Input)
If `IY0` = 1, `DFS2` is the degrees of freedom for error from the fitted regression. If `IY0` = 0, `DFS2` is the pooled degrees of freedom from the estimate of sigma-squared based on the fitted regression and the additional responses used to compute the mean `Y0`.

B — Vector of length `INTCEP` + 1 containing a least-squares solution for the intercept and slope. (Input)

`INTCEP`	Intercept	Slope
0		B(1)
1	B(1)	B(2)

XYMEAN — Vector of length 2 with the mean of the independent and dependent variables, respectively. (Input, if `INTCEP` = 1)
If `INTCEP` = 0, `XYMEAN` is not referenced and can be a vector of length 1.

SSX — Sum of squares for *x*. (Input)
If `INTCEP` = 1, `SSX` is the sum of squares of deviations of *x* from its mean. If `INTCEP` = 0, `SSX` must not be corrected for the mean.

S2 — s^2, the estimate of the variance of the error in the model. (Input)
If `IY0` = 1, `S2` is the estimate of σ^2 from the fitted regression. If `IY0` = 0, `S2` is the pooled estimate of σ^2 based on the fitted regression, and the additional responses used to compute the mean `Y0`.

Y0 — Value of the response variable for which an interval estimate of the corresponding independent variable value is desired. (Input)

X0HAT — Point estimate of the independent variable. (Output)

XLOWER — Lower limit of the interval estimate for the independent variable. (Output)

XUPPER — Upper limit of the interval estimate for the independent variable. (Output)

Optional Arguments

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

CONPER — Confidence level for the interval estimation. (Input)
CONPER must be expressed as a percentage between 0.0 and 100.0. CONPER often will be 90.0, 95.0, 99.0. For one-sided confidence intervals with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

IY0 — Option for Y0. (Input)
Default: IY0 = 1.

IY0	Action
0	Y0 is a sample mean of one or more responses.
1	Y0 is the true mean response.

SWTFY0 — Sum of products of weights with frequencies for Y0. (Input, if IY0 = 0)
In the ordinary case, when weights and frequencies are all one, SWTFY0 is the number of observations used to obtain the mean Y0. If IY0 = 1, SWTFY0 is not referenced.

FORTRAN 90 Interface

Generic: CALL RINPF (SUMWTF,DFS2 , B , XYMEAN , SSX , S2 , Y0 , X0HAT, XLOWER , XUPPER [, ...])
Specific: The specific interface names are S_RINPF and D_RINPF.

FORTRAN 77 Interface

Single: CALL RINPF (SUMWTF,DFS2 , INTCEP, B , XYMEAN , SSX , S2 , CONPER , IY0, SWTFY0, Y0, X0HAT, XLOWER , XUPPER)
Double: The double precision name is DRINPF.

Description

Routine **RINPF** computes a confidence interval on the independent variable setting x_0 for a given response y_0 from the results of a straight line fit. Here, y_0 may represent the mean of k responses or the true mean response. The results of routine **RLINE** or **RONE** can be used for input into **RINPF**. The simple linear regression model is assumed,

$$y_i = \beta_0 + \beta_1 x + \varepsilon_i \quad i = 1,2,...n+k$$

where the ε_i's are independently distributed normal errors with mean zero and variance σ^2/w_i. Here, n is the total number of observations used in the fit of the line, i.e., n = **DFE** + **INTCEP** + 1 where **DFE** is the degrees of freedom from the fitted regression. Also, k is the number of additional responses used to determine y_0. The w_i's are the weights that must be used in the fit of the model. The methodology is discussed by Graybill (1976, pages 280-283). For the case when **IY0** = 1, the estimate of σ^2, s^2 (stored in **S2**), is the usual estimate of σ^2 from the fitted regression based on the first n observations. If **IY0** = 0, the estimate of σ^2 is a pooled estimator based on the fitted regression and the k responses that produce $\overline{y}_0$.

This pooled estimator (stored in **S2**) is given by

$$s^2 = \frac{\sum_{i=1}^{n} w_i\left(y_i - \hat{\beta}_o - \hat{\beta}_1 x_i\right)^2 + \sum_{i=n+1}^{n+k} w_i\left(y_i - \overline{y}_0\right)^2}{(n-2)+(k-1)}$$

where (n - 2) + (k - 1) (stored in **DFS2**) is the pooled degrees of freedom for s^2.

First, a point estimate $\hat{x}_0$ (stored in **X0HAT**) is computed by

$$\hat{x}_0 = \frac{y_0 - \hat{\beta}_0}{\hat{\beta}_1}$$

Then, a test of the hypothesis H_0 : $\beta_1 = 0$ vs. H_a : $\beta_1 \neq 0$ is performed. If H_0 is accepted, the model becomes $y_i = \beta_0 + \varepsilon_i$, and therefore no confidence interval exists for x_0 because it is no longer in the model. In this case, a type 3 warning error is issued. If H_0 is rejected, a confidence interval exists and is computed for the case **IY0** = 1 by

$$\bar{x} + \frac{\hat{\beta}_1\left(y_0 - \bar{y}\right)}{a} \pm \frac{ts}{a}\sqrt{\frac{a}{\sum_{i=1}^{n} w_i} + \frac{a}{\sum_{i=n+1}^{n+k} w_i} + \frac{\left(y_0 - \bar{y}\right)^2}{\sum_{i=1}^{n} w_i\left(x_i - \bar{x}\right)^2}}$$

where

$$a = \hat{\beta}_1^2 - \frac{t^2 s^2}{\sum_{i=1}^{n} w_i\left(x_i - \bar{x}\right)^2}$$

and t is the 50 + **CONPER**/2 percentile of the t distribution with **DFS2** degrees of freedom. The interval actually has a confidence coefficient less than that specified by **CONPER**.

In the weighted case, which was discussed earlier, the means (stored in **XYMEAN**) and the sum of squares for x (stored in **SSX**) are all weighted. When the variances of the ε_i's are all equal, ordinary least squares must be used, this corresponds to all $w_i = 1$.

Modifications are necessary to the preceding discussion for other cases. For the case when an intercept is not in the model, let $\beta_0 = 0, \hat{\beta}_0 = 0$ the pooled degrees of freedom of s^2 equal to (n - 1) + (k - 1), and replace the first term under the square root symbol with zero, $\bar{x}$ with zero, and $\bar{y}$ with zero.

For the case of the true response mean, i.e, when **IY0** = 1, replace the second term under the square root symbol by zero.

Comments

Informational errors

Type	Code	Description
3	2	The slope is not significant at the (100 - CONPER)% level. Confidence limits XLOWER and XUPPER cannot be obtained.

Example

This example fits a line to a set of data discussed by Draper and Smith (1981, Table 1.1, page 9). The response y is the amount of steam used per month (in pounds), and the independent variable x is the average atmospheric temperature (in degrees Fahrenheit). A 95% confidence interval for the temperature x_0 is computed given a single response of $y_0 = 10$.

```
        USE RINPF_INT
```

```
      USE RLINE_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=25)
!
      INTEGER    INTCEP, IY0, NOUT
      REAL       B(2), B0, B1, CONPER, DFS2, S2, SSX, STAT(12), &
                 SUMWTF, SWTFY0, X0HAT, XDATA(NOBS), XLOWER, XUPPER, &
                 XYMEAN(2), Y0, YDATA(NOBS)
!
      DATA XDATA/35.3, 29.7, 30.8, 58.8, 61.4, 71.3, 74.4, 76.7, 70.7, &
          57.5, 46.4, 28.9, 28.1, 39.1, 46.8, 48.5, 59.3, 70.0, 70.0, &
          74.5, 72.1, 58.1, 44.6, 33.4, 28.6/
      DATA YDATA/10.98, 11.13, 12.51, 8.4, 9.27, 8.73, 6.36, 8.5, &
          7.82, 9.14, 8.24, 12.19, 11.88, 9.57, 10.94, 9.58, 10.09, &
          8.11, 6.83, 8.88, 7.68, 8.47, 8.86, 10.36, 11.08/
!
      CALL RLINE (XDATA, YDATA, B0, B1, STAT=STAT)
      SUMWTF    = NOBS
      DFS2      = STAT(10)
      INTCEP    = 1
      B(1)      = B0
      B(2)      = B1
      XYMEAN(1) = STAT(1)
      XYMEAN(2) = STAT(2)
      SSX       = STAT(3)*(NOBS-1)
      S2        = STAT(11)/STAT(10)
      CONPER    = 95.0
      IY0       = 0
      SWTFY0    = 1.0
      Y0        = 10.0
      CALL RINPF (SUMWTF, DFS2, B, XYMEAN, SSX, S2, Y0, X0HAT, XLOWER,  &
                  XUPPER, IY0=IY0, SWTFY0=SWTFY0)
      CALL UMACH (2, NOUT)


      WRITE (NOUT,*) 'X0HAT = ', X0HAT
      WRITE (NOUT,*) '(XLOWER,XUPPER) = (', XLOWER, ',', XUPPER, ')'
      END
```

Output

```
 X0HAT = 45.3846
 (XLOWER,XUPPER) = (20.2627,69.347)
```

RLSE

Fits a multiple linear regression model using least squares.

Required Arguments

Y — Vector of length **NOBS** containing the dependent (response) variable. (Input)

X — **NOBS** by **NIND** matrix containing the independent (explanatory) variables. (Input)

B — Vector of length **INTCEP** + **NIND** containing a least-squares solution $\hat{\beta}$ for the regression coefficients. (Output)

For **INTCEP** = 0, the fitted value for observation **I** is
B(1) * X(**I**, 1) + B(2) * X(**I**, 2) + … + B(**NIND**) * X(**I**, **NIND**).

For **INTCEP** = 1, the fitted value for observation **I** is
B(1) + B(2) * X(**I**, 1) + … + B(**NIND** + 1) * X(**I**, **NIND**).

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**Y**,1).

NIND — Number of independent (explanatory) variables. (Input)
Default: **NIND** = size (**X**,2).

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

INTCEP — Intercept option. (Input)
Default: **INTCEP** = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

SST — Total sum of squares. (Output)
If **INTCEP** = 1, the total sum of squares is corrected for the mean.

SSE — Sum of squares for error. (Output)

FORTRAN 90 Interface

Generic: CALL RLSE (Y, X, B [, ...])

Specific: The specific interface names are S_RLSE and D_RLSE.

FORTRAN 77 Interface

Single: CALL RLSE (NOBS, Y, NIND, X, LDX, INTCEP, B, SST, SSE)

Double: The double precision name is DRLSE.

Description

Routine RLSE fits a multiple linear regression model with or without an intercept. If INTCEP = 1, the multiple linear regression model is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_k x_{ik} + \varepsilon_i \quad i = 1, 2, \ \ldots, n$$

where the observed values of the y_i's (input in Y) constitute the responses or values of the dependent variable, the x_{i1}'s, x_{i2}'s, ..., x_{ik}'s (input in X) are the settings of the k (input in NIND) independent variables, β_0, β_1, ..., β_k are the regression coefficients whose estimated values are output in B, and the ε_i's are independently distributed normal errors each with mean zero and variance σ^2. Here, n is the number of valid rows in the augmented matrix (X, Y), i.e. n equals NOBS - NRMISS (the number of rows that do not contain NaN). If INTCEP = 0, β_0 is not included in the model.

Routine RLSE computes estimates of the regression coefficients by minimizing the sum of squares of the deviations of the observed response y_i from the fitted response

$$\hat{y}_i$$

for the n observations. This minimum sum of squares (the error sum of squares) is output and denoted by

$$SSE = \sum_{i=1}^{n} \left(y_i - \hat{y}_i \right)^2$$

In addition, the total sum of squares is output. For the case, INTCEP = 1, the total sum of squares is the sum of squares of the deviations of y_i from its mean

$$\bar{y}$$

— the so-called *corrected total sum of squares*; it is denoted by

$$SST = \sum_{i=1}^{n} \left(y_i - \overline{y} \right)^2$$

For the case **INTCEP** = 0, the total sum of squares is the sum of squares of y_i — the so-called *uncorrected total sum of squares*; it is denoted by

$$SST = \sum_{i=1}^{n} y_i^2$$

See Draper and Smith (1981) for a good general treatment of the multiple linear regression model, its analysis, and many examples.

In order to compute a least-squares solution, **RLSE** performs an orthogonal reduction of the matrix of regressors to upper triangular form. If the user needs the upper triangular matrix output for subsequent computing, the routine **R2SE** can be invoked in place of **RLSE**. (See the description of **R** in Comment 1). The reduction is based on one pass through the rows of the augmented matrix (**X**, **Y**) using fast Givens transformations. (See routines **SROTMG** and **SROTM** Golub and Van Loan, 1983, pages 156-162, Gentleman, 1974.) This method has the advantage that the loss of accuracy resulting from forming the crossproduct matrix used in the normal equations is avoided.

With **INTCEP** = 1, the current means of the dependent and independent variables are used to internally center the data for improved accuracy. Let x_j be a column vector containing the *j*-th row of data for the independent variables. Let $\overline{x}_i$ represent the mean vector for the independent variables given the data for rows 1, 2, ..., *i*. The current mean vector is defined to be

$$\overline{x}_i = \frac{\sum_{j=1}^{i} x_j}{i}$$

The *i*-th row of data has $\overline{x}_i$ subtracted from it and is then weighted by *i*/(*i* - 1). Although a crossproduct matrix is not computed, the validity of this centering operation can be seen from the following formula for the sum of squares and crossproducts matrix:

$$\sum_{i=1}^{n} \left(x_i - \overline{x}_n \right) (x_i - \overline{x}_n)^T = \sum_{i=2}^{n} \frac{i}{i-1} \left(x_i - \overline{x}_i \right) (x_i - \overline{x}_i)^T$$

An orthogonal reduction on the centered matrix is computed. When the final computations are performed, the first row of **R** and the first element of **B** are updated so that they reflect the statistics for the original (uncentered) data. This means that the estimate of the intercept and the **R** matrix are for the uncentered data.

As part of the final computations, **RLSE** checks for linearly dependent regressors. If the *i*-th regressor is a linear combination of the first *i* - 1 regressors, the *i*-th diagonal element of **R** is close to zero (exactly zero if infinite precision arithmetic could be used) prior to the final computations. In particular, linear dependence of the regressors is declared if any of the following three conditions is satisfied:

- A regressor equals zero.
- Two or more regressors are constant.
- The result of

$$\sqrt{1 - R^2_{i \cdot 1, 2, \ldots, i-1}}$$

  is less than or equal to 100 × **ε** where **ε** is the machine epsilon. (For **RLSE**, **ε** = **AMACH**(4) and for **DRLSE**, **ε** = **DMACH**(4). See routines **AMACH** and **DMACH** in Reference Material).

Here, $R_{i.1, 2, \ldots, i-1}$ is the multiple correlation coefficient of the *i*-th independent variable with the first *i* - 1 independent variables. If no intercept is in the model (**INTCEP** = 0), the "multiple correlation" coefficient is computed without adjusting for the mean.

On completion of the final computations, if the *i*-th regressor is declared to be linearly dependent upon the previous *i* - 1 regressors, then the *i*-th element of **B** and all elements in the *i*-th row of **R** are set to zero. Finally, if a linear dependence is declared, an informational (error) message, code 1, is issued indicating the model is not full rank.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2SE/DR2SE**. The reference is:

   ```
   CALL R2SE (NOBS, Y, NIND, X, LDX, INTCEP, B, SST, SSE, R, LDR, DFE, NRMISS, WK)
   ```

   The additional arguments are as follows:

   R — **INTCEP** + **NIND** by **INTCEP** + **NIND** upper triangular matrix containing the *R* matrix from a *QR* decomposition of the matrix of regressors. (Output)
   All of the diagonal element of *R* are taken to be nonnegative. The rank of the matrix of regressors is the number of positive diagonal elements, which equals
   NOBS - NRMISS - DFE.

   LDR — Leading dimension of *R* exactly as specified in the dimension statement in the calling program. (Input)

   DFE — Degrees of freedom for error. (Output)

   NRMISS — Number of rows in the augmented matrix (**X**, **Y**) containing NaN (not a number). (Output)
   If a row contains NaN, that row is excluded from all other computations.

   WK — Work vector of length 5 * **NIND** + 4 * **INTCEP** + 2.

2. Informational error

Type	Code	Description
3	1	The model is not full rank. There is not a unique least-squares solution. If the I-th diagonal element of R is zero, B(I) is set to zero in order to compute a solution.

Examples

Example 1

A regression model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i \qquad i = 1,2, \ldots, 9$$

is fitted to data taken from Maindonald (1984, pages 203-204).

```
      USE RLSE_INT
      USE WRRRN_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    INTCEP, LDX, NCOEF, NIND, NOBS, J
      PARAMETER  (INTCEP=1, NIND=3, NOBS=9, LDX=NOBS, &
                 NCOEF=INTCEP+NIND)
!
      INTEGER    NOUT
      REAL       B(NCOEF), SSE, SST, X(LDX,NIND), Y(NOBS)
!
      DATA (X(1,J),J=1,NIND)/ 7.0,  5.0, 6.0/, Y(1)/ 7.0/
      DATA (X(2,J),J=1,NIND)/ 2.0, -1.0, 6.0/, Y(2)/-5.0/
      DATA (X(3,J),J=1,NIND)/ 7.0,  3.0, 5.0/, Y(3)/ 6.0/
      DATA (X(4,J),J=1,NIND)/-3.0,  1.0, 4.0/, Y(4)/ 5.0/
      DATA (X(5,J),J=1,NIND)/ 2.0, -1.0, 0.0/, Y(5)/ 5.0/
      DATA (X(6,J),J=1,NIND)/ 2.0,  1.0, 7.0/, Y(6)/-2.0/
      DATA (X(7,J),J=1,NIND)/-3.0, -1.0, 3.0/, Y(7)/ 0.0/
      DATA (X(8,J),J=1,NIND)/ 2.0,  1.0, 1.0/, Y(8)/ 8.0/
      DATA (X(9,J),J=1,NIND)/ 2.0,  1.0, 4.0/, Y(9)/ 3.0/
!
      CALL RLSE (Y, X, B, SST=SST, SSE=SSE)
      CALL WRRRN ('B', B)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'SST = ', SST, '  SSE = ', SSE
99999 FORMAT (A7, F7.2, A7, F7.2)
      END
```

Output

```
      B
1   7.733
2  -0.200
3   2.333
```

```
    4  -1.667

    SST =  156.00  SSE =   4.00
```

Example 2

A weighted least-squares fit is computed using the model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i \qquad i = 1, 2, \ldots, 4$$

and weights $1/i^2$ discussed by Maindonald (1984, pages 67 - 68). In order to compute the weighted least-squares fit, using an ordinary least squares routine (**RLSE**), the regressors (including the column of ones for the intercept term as well as the independent variables) and the responses must be transformed prior to invocation of **RLSE**. The transformed regressors and responses can be computed by using routine **SHPROD** (IMSL MATH/LIBRARY). For the *i*-th case the corresponding response and regressors are multiplied by a square root of the *i*-th weight. Because the column of ones corresponding to the intercept term in the untransformed model, is transformed by the weights, this transformed column of ones must be input to the least squares subroutine as an additional independent variable along with the option **INTCEP** = 0.

In terms of the original, untransformed regressors and responses, the minimum sum of squares for error output in **SSE** is

$$\text{SSE} = \sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i \right)^2$$

where here the weight $w_i = 1/i^2$. Also, since **INTCEP** = 0, the uncorrected total sum of squares is output in **SST**. In terms of the original untransformed responses,

$$\text{SST} = \sum_{i=1}^{n} w_i y_i^2$$

```
          USE RLSE_INT
          USE SHPROD_INT
          USE WRRRN_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    INTCEP, LDX, NCOEF, NIND, NOBS, J
          PARAMETER  (INTCEP=0, NIND=3, NOBS=4, LDX=NOBS, &
                     NCOEF=INTCEP+NIND)
    !
          INTEGER    I, NOUT
          REAL       B(NCOEF), SQRT, SSE, SST, W(NOBS), X(LDX,NIND), &
                     Y(NOBS)
          INTRINSIC  SQRT
    !
          DATA (X(1,J),J=1,NIND)/1.0, -2.0, 0.0/, Y(1)/-3.0/
```

```
      DATA (X(2,J),J=1,NIND)/1.0, -1.0, 2.0/, Y(2)/ 1.0/
      DATA (X(3,J),J=1,NIND)/1.0,  2.0, 5.0/, Y(3)/ 2.0/
      DATA (X(4,J),J=1,NIND)/1.0,  7.0, 3.0/, Y(4)/ 6.0/
!
      DO 10  I=1, NOBS
!                                  Assign weights
         W(I) = 1.0/I**2
!                                  Store square roots of weights
         W(I) = SQRT(W(I))
   10 CONTINUE
!                                  Transform regressors
      DO 20  J=1, NIND
         CALL SHPROD (NOBS, W, 1, X(:,J), 1, X(:,J), 1)
   20 CONTINUE
!                                  Transform response
      CALL SHPROD (NOBS, W, 1, Y, 1, Y, 1)
!
      CALL RLSE (Y, X, B, INTCEP=INTCEP, SST=SST, SSE=SSE)
!
      CALL WRRRN ('B', B)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,99999) 'SST = ', SST, '  SSE = ', SSE
99999 FORMAT (A7, F7.2, A7, F7.2)
      END
```

Output

```
      B
1  -1.431
2   0.658
3   0.748

SST =   11.94  SSE =   1.01
```

RCOV

Fits a multivariate linear regression model given the variance-covariance matrix.

Required Arguments

COV — NIND + NDEP by NIND + NDEP matrix containing the variance-covariance matrix or sum of squares and crossproducts matrix. (Input)
Only the upper triangle of COV is referenced. The first NIND rows and columns correspond to the independent variables, and the last NDEP rows and columns correspond to the dependent variables. If INTCEP = 0, COV contains raw sums of squares and crossproducts. If INTCEP = 1, COV contains sums of squares and crossproducts corrected for the mean. If weighting is desired, COV contains weighted sums of squares and crossproducts.

XYMEAN — Vector of length NIND + NDEP containing variable means. (Input, if INTCEP = 1)
The first NIND elements of XYMEAN are for the independent variables in the same order in which they appear in COV. The last NDEP elements of XYMEAN are for the dependent variables in the same order in which they appear in COV. If weighting is desired, XYMEAN contains weighted means. If INTCEP = 0, XYMEAN is not referenced and can be a vector of length one.

SUMWTF — Sum of products of weights with frequencies. (Input, if INTCEP = 1)
In the ordinary case when weights and frequencies are all one, SUMWTF equals the number of observations.

B — INTCEP + NIND by NDEP matrix containing a least-squares solution $\hat{B}$ for the regression coefficients. (Output)
Column *j* is for the *j*-th dependent variable. If INTCEP = 1, row 1 is for the intercept. Row INTCEP + *i* is for the *i*-th independent variable. Elements of the appropriate row(s) of $\hat{B}$ are set to 0.0 if linear dependence of the regressors is declared.

Optional Arguments

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

NIND — Number of independent (explanatory) variables. (Input)
Default: NIND = size (B,1) - INTCEP.

NDEP — Number of dependent (response) variables. (Input)
Default: NDEP = size (B,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

TOL — Tolerance used in determining linear dependence. (Input)
For RCOV, TOL = 100 * AMACH(4) is a common choice. See documentation for routine AMACH in *Reference Material*.
Default: TOL = 1.e-5 for single precision and 2.d -14 for double precision.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

R — INTCEP + NIND by INTCEP + NIND upper triangular matrix containing the *R* matrix from a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. (Output)
Elements of the appropriate row(s) of *R* are set to 0.0 if linear dependence of the regressors is declared.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

IRANK — Rank of *R*. (Output)
IRANK less than INTCEP + NIND indicates that linear dependence of the regressors was declared. In this case, some rows of $\hat{B}$ are set to zero.

SCPE — NDEP by NDEP matrix containing the error (residual) sums of squares and crossproducts. (Output)

LDSCPE — Leading dimension of SCPE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCPE = size (SCPE,1).

FORTRAN 90 Interface

Generic: CALL RCOV (COV, XYMEAN, SUMWTF, B [, ...])

Specific: The specific interface names are S_RCOV and D_RCOV.

FORTRAN 77 Interface

Single: CALL RCOV (INTCEP, NIND, NDEP, COV, LDCOV, XYMEAN, SUMWTF, TOL, B, LDB, R, LDR, IRANK, SCPE, LDSCPE)

Double: The double precision name is DRCOV.

Description

Routine RCOV fits a multivariate linear regression model given the variance-covariance matrix (or sum of squares and crossproducts matrix) for the independent and dependent variables. Typically, an intercept is to be in the model, and the corrected sum of squares and crossproducts matrix is input for COV. Routine CORVC in Chapter 3, "Correlation" can be invoked to compute the corrected sum of squares and crossproducts matrix. Routine RORDM in Chapter 19, "Utilities" can reorder this matrix, if required. If an intercept is not to be included in the model, a raw (uncorrected) sum of squares and crossproducts matrix must be input for COV; and SUMWTF and XYMEAN are not used in the computations. Routine MXTXF (IMSL MATH/LIBRARY) can be used to compute the raw sum of squares and crossproducts matrix.

Routine RCOV is based on a Cholesky factorization of COV. Let k (input in NIND) be the the number of independent variables, and d (input in SUMWTF) the denominator used in computing the x means (input in the first k locations of XYMEAN). The matrix R is formed by computing a Cholesky factorization of the first k rows and columns of COV. If INTCEP equals one, the k rows from this factorization are appended to the initial row

$$\sqrt{d}, \sqrt{d}\bar{x}_1, \ \ldots, \sqrt{d}\bar{x}_k$$

The resulting R matrix is the Cholesky factor of the X^TX matrix where X contains a column of ones as its first column and the independent variable settings as its remaining k columns.

Maindonald (1984, Chapter 3) discusses the Cholesky factorization as it applies to regression computations.

The routine RCOV checks sequentially for linear dependent regressors. Linear dependence of the regressors is declared if

$$1 - R^2_{i\cdot 1,\,2,\,\ldots,\,i-1}$$

is less than or equal to **TOL**. Here, $R_{i\cdot 1,\,2,\,\ldots,\,i-1}$ is the multiple correlation coefficient of the *i*-th independent variable with the first *i* - 1 independent variables. If no intercept is in the model (**INTCEP** = 0), the "multiple correlation" coefficient is computed without adjusting for the mean. When a dependence is declared, elements of the corresponding rows of *R* and *B* are set to zero. Maindonald (1984, Sections 3.3, 3.4, and 3.9) discusses these implementation details of the Cholesky factorization in regression problems.

Comments

1. Informational error

Type	Code	Description
3	1	COV is not a variance-covariance matrix within the tolerance defined by TOL.

Example

This example uses a data set from Draper and Smith (1981, pages 629 - 630). This data set is put into the matrix **X** by routine **GDATA** (Chapter 19, "Utilities"). The first four columns are for the independent variables, and the last column is for the dependent variable. Routine **CORVC** in Chapter 3, "Correlation" is invoked to compute the corrected sum of squares and crossproducts matrix. Then, **RCOV** is invoked to compute the regression coefficient estimates, the *R* matrix, and the sum of squares for error.

```
      USE RCOV_INT
      USE GDATA_INT
      USE CORVC_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDX, NDX, NIND, NDEP, LDCOV, LDSCPE, INTCEP
      INTEGER    LDB, LDR, NROW, NVAR, IRANK, NOUT
      PARAMETER (LDX=13, NDX=5, NIND=4, NDEP=1, LDCOV=NIND+NDEP, &
                 LDSCPE=NDEP)
      PARAMETER (INTCEP=1, LDB=INTCEP+NIND, LDR=INTCEP+NIND)
      REAL       XYMEAN(NIND+NDEP)
      REAL       X(LDX,NDX), B(LDB,NDEP), R(LDR,INTCEP+NIND)
      REAL       COV(LDCOV,NIND+NDEP), SCPE(LDSCPE,NDEP), SUMWTF
      INTEGER    INCD(1,1), ICOPT
!
      CALL GDATA (5, X, NROW, NVAR)
!
      ICOPT = 1
      CALL CORVC (NVAR, X, COV, ICOPT=ICOPT, XMEAN=XYMEAN, SUMWT=SUMWTF)
!
      CALL RCOV (COV, XYMEAN, SUMWTF, B, R=R, IRANK=IRANK, &
```

```
                    SCPE=SCPE)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  SCPE(1,1) = ', SCPE(1,1)
      CALL WRRRN ('B', B, 1, INTCEP+NIND, 1)
      CALL WRRRN ('R', R)
      END
```

Output

```
 IRANK =   5  SCPE(1,1) =     47.8638

                   B
     1       2       3       4       5
 62.40    1.55    0.51    0.10   -0.14

                     R
          1       2       3       4       5
 1      3.6    26.9   173.6    42.4   108.2
 2      0.0    20.4    12.3   -18.3   -14.2
 3      0.0     0.0    52.5     1.1   -54.6
 4      0.0     0.0     0.0    12.5   -12.9
 5      0.0     0.0     0.0     0.0     3.4
```

RGIVN

Fits a multivariate linear regression model via fast Givens transformations.

Required Arguments

X — |NROW| by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)

IIND	Meaning
< 0	The first -IIND columns of X contain the independent (explanatory) variables.
> 0	The IIND independent variables are specified by the column numbers in INDIND.
= 0	There are no independent variables.

The regressors are the intercept (if INTCEP = 1) and the independent variables. There are INTCEP + |IIND| regression coefficients for each dependent variable.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent variables. (Input, if IIND is positive)
If IIND is nonpositive, INDIND is not referenced and can be a vector of length one.

IDEP — Dependent variable option. (Input)

IDEP	Meaning
< 0	The last -IDEP columns of X contain the dependent (response) variables. That is, columns NCOL + IDEP + 1, NCOL + IDEP + 2, …, NCOL contain the dependent variables.
> 0	The IDEP dependent (response) variables are specified by the column numbers in INDDEP..
= 0	There are no dependent variables. (Generally, this option is not used. The *R* matrix from a *QR* decomposition of a matrix of regressors is computed.)

INDDEP — Index vector of length IDEP containing the column numbers of X that are the dependent variables. (Input, if IDEP is positive)
If IDEP is nonpositive, INDDEP is not referenced and can be a vector of length one.

B — INTCEP + |IIND| by |IDEP| matrix containing a least-squares solution $\hat{B}$ for the regression coefficients on return from the final invocation of this routine. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
If INTCEP = 1, row 1 is for the intercept. Row INTCEP + I is for the I-th independent variable. Column *j* is for the *j*-th dependent variable.

IDO	Action
1 or 2	A current least-squares solution is given by a solution *x* to the equation *Rx* = B.
0 or 3	A least-squares solution for the regression coefficients is returned in B. Elements of the appropriate row(s) of B are set to 0.0 if linear dependence of the regressors is declared.

If IDEP = 0, B is not referenced and can be a vector of length 1.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RGIVN for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to RGIVN will be made. Initialization and updating for the data in X are performed.
2	This is an intermediate invocation of RGIVN, and updating for the data in X is performed.
3	This is the final invocation of this routine. Updating for the data in X and wrap-up computations are performed.

NROW — The absolute value of NROW is the number of rows of data currently input in X. (Input)
NROW may be positive, zero, or negative. Negative NROW means that the -NROW rows of data are to be deleted from some aspects of the analysis, and this should be done only if IDO is 2 or 3 and the wrap-up computations have not been performed. When a negative value is input for NROW, it is assumed that each of the -NROW rows of X has been input (with positive NROW) in previous invocations of RGIVN. Use of negative values of NROW should be made with care and with the understanding that XMIN and XMAX cannot be updated properly in this case. It is also possible that a constant variable in the remaining data will not be recognized as such.
Default: NROW = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies. If X(I, IFRQ) = 0.0, none of the remaining elements of row I of X are referenced, and updating of statistics is skipped for row I.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

ICEN — Data centering option. (Input)
If INTCEP = 0, ICEN must equal 0.
Default: ICEN = 1.

ICEN	Action
0	No centering. This option should be used when (1) the data are already centered; (2) there is no intercept in the model; or (3) the independent variables for a large percentage of the data are zero, and sparsity of the problem needs to be preserved in order that the Givens rotations are performed quickly.
1	Variables are centered using the method of provisional means for improved accuracy of the computations. The final estimate for the intercept and the *R* matrix are given for the uncentered data. This option is generally recommended.

TOL — Tolerance used in determining linear dependence. (Input)
For RGIVN, TOL = 100 * AMACH(4) is a common choice. See the documentation for routine AMACH in *Reference Material*.
Default: TOL = 1.e-5 for single precision and 2.D-14 for double precision.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

R — INTCEP + |IIND| by INTCEP + |IIND| upper triangular matrix containing the *R* matrix from a *QR* decomposition of the matrix of regressors on return from the final invocation of this routine. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

IDO	Action
1 or 2	The current matrix of raw sums of squares and crossproducts for the regressors can be found as $R^T \cdot \text{diag(D)} \cdot R$ where diag(D) is the diagonal matrix whose diagonal elements are the elements of the vector D.
0 or 3	The matrix of raw sums of squares and crossproducts for the regressors can be found as $R^T R$. Elements of the appropriate row(s) of *R* are set to 0.0 if linear dependence of the regressors is declared.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

D — Vector of length INTCEP + |IIND| containing scale factors for fast Givens transformations. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

IDO	Action
1 or 2	D contains the current scale factors associated with the fast Givens transformations.
0 or 3	Each element of D is set to 1.0.

IRANK — Rank of *R*. (Output, if IDO = 0 or 3)
IRANK less than INTCEP + |IIND| indicates linear dependence of the regressors was declared.

DFE — Degrees of freedom for error on return from the final invocation of this routine. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
Prior to the final invocation of RGIVN, DFE is the sum of the frequencies.

SCPE — |IDEP| by |IDEP| matrix containing error (residual) sums of squares and crossproducts. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
SCPE(*m*, *n*) contains the current sum of crossproducts of residuals for the *m*-th and *n*-th dependent variables. If IDEP = 0, SCPE is not referenced and can be a 1 by 1 array.

LDSCPE — Leading dimension of SCPE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCPE = size (SCPE,1).

NRMISS — Number of rows of data encountered in calls to **RGIVN** that contain any missing values for the independent, dependent, weight, or frequency variables. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)
NaN (not a number) is used as the missing value code. Any row of **X** containing NaN as a value of the independent, dependent, weight, or frequency variables is omitted from the analysis.

XMIN — Vector of length **INTCEP** + |**IIND**| containing the minimum values for each of the regressors. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)

XMAX — Vector of length **INTCEP** + |**IIND**| containing the maximum values for each of the regressors. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)

FORTRAN 90 Interface

Generic: **CALL RGIVN** (X, **IIND**, **INDIND**, **IDEP**, **INDDEP**, **B** [, ...])
Specific: The specific interface names are **S_RGIVN** and **D_RGIVN**.

FORTRAN 77 Interface

Single: **CALL RGIVN** (**IDO**, **NROW**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IDEP**, **INDDEP**, **IFRQ**, **IWT**, **ICEN**, **TOL**, **B**, **LDB**, **R**, **LDR**, **D**, **IRANK**, **DFE**, **SCPE**, **LDSCPE**, **NRMISS**, **XMIN**, **XMAX**)
Double: The double precision name is **DRGIVN**.

Description

Routine **RGIVN** fits a multivariate linear regression model. (See the chapter introduction for a description of the multivariate linear regression model.) The routine **RGIVN** is designed so that multiple invocations can be made. In this case, zero, one, or several rows of the data set can be input for each invocation of **RGIVN** (with **IDO** = 1, 2, 2, ..., 2, 3). Alternatively, one invocation of **RGIVN** (with **IDO** = 0) can be made with the entire data set contained in **X**. Routine RSTAT can be invoked after the wrap-up computations are performed by **RGIVN** to compute and print summary statistics related to the fitted regression.

Routine **RGIVN** performs an orthogonal reduction of the matrix of regressors to upper triangular form. The reduction is based on fast Givens transformations. (See routines **SROTMG** and **SROTM**, Golub and Van Loan 1983, pages 156-162, Gentleman 1974.) This method has two main advantages: (1) the loss of accuracy resulting from forming the crossproduct matrix used in the normal equations is avoided, (2) data can be conveniently added or deleted to take advantage of the previous computations performed.

With **ICEN** = 1, the current means of the independent and dependent variables are used to center the data for improved accuracy. Let x_i be a column vector containing the *i*-th row of data for the independent variables. Let $\bar{x}_i$ represent the mean vector for the independent variables given the data for observations 1, 2, ..., *i*. The mean vector is defined to be

$$\bar{x}_i = \frac{\sum_{j=1}^{i} w_j f_j x_j}{\sum_{j=1}^{i} w_j f_j}$$

where the w_j's and f_j's are the weights and frequencies, respectively. The *i*-th row of data has $\bar{x}_i$ subtracted from it, and then $w_i f_i$ is multiplied by the factor a_i/a_{i-1} where

$$a_i = \sum_{j=1}^{i} w_j f_j$$

Although a crossproduct matrix is not computed, the validity of this centering operation can be seen from the following formula for the sum of squares and crossproducts matrix:

$$\sum_{i=1}^{n} w_i f_i (x_i - \bar{x}_n)(x_i - \bar{x}_n)^T = \sum_{i=2}^{n} \frac{a_i}{a_{i-1}} w_i f_i (x_i - \bar{x}_i)(x_i - \bar{x}_i)^T$$

An orthogonal reduction on the centered matrix is computed. When wrap-up computations (**IDO** = 3 or **IDO** = 0) are performed, the first rows of *R* and *B* are updated so that they reflect the statistics for the original (uncentered) data. This means that the estimate of the intercept and the *R* matrix are for the uncentered data.

If the *i*-th regressor is a linear combination of the first *i* - 1 regressors, the *i*-th diagonal element of *R* will be close to zero (exactly zero if infinite precision arithmetic could be used) prior to the wrap-up computations. When performing the wrap-up computations, **RGIVN** checks sequentially for linear dependent regressors. Linear dependence of the regressors is declared if any of the following three conditions is satisfied:

- A regressor equals zero, as determined from **XMIN** and **XMAX**.
- Two or more regressors are constant, as determined from **XMIN** and **XMAX**.
- $\sqrt{1 - R^2_{i \cdot 1,2,\ldots i-1}}$ is less than or equal to **TOL**. Here, $R_{i \cdot 1,2,\ldots,i-1}$ is the multiple correlation coefficient of the *i*-th independent variable with the first *i* - 1 independent variables. If no intercept is in the model (**INTCEP** = 0) the "multiple correlation" coefficient is computed without adjusting for the mean.

When a dependence is declared, *R* is changed in the wrap-up computations so as to reflect the deletion of the *i*-th regressor from the model. On completion of the wrap-up computations, if the *i*-th regressor is declared to be dependent upon the previous *i* - 1 regressors, then the *R* and $\hat{B}$ matrices will have all elements in their *i*-th rows set to zero.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2IVN/DR2IVN**. The reference is:

   CALL R2IVN (IDO, NROW, NCOL, X, LDX, INTCEP, IIND, INDIND, IDEP, INDDEP, IFRQ, IWT, ICEN, TOL, B, LDB, R, LDR, D, IRANK, DFE, SCPE, LDSCPE, NRMISS, XMIN, XMAX, WK)

   The additional argument is:

   WK — Work vector of length INTCEP + |IIND| + |IDEP|

2. Informational errors

Type	Code	Description
4	1	Negative weight encountered.
4	2	Negative frequency encountered.

Examples

Example 1

The first example uses a data set from Draper and Smith (1981, pages 629-630). This data set is put into the matrix **X** by routine GDATA in Chapter 19, "Utilities". There is 1 dependent variable and 4 independent variables. RGIVN is invoked to fit the regression model with the IDO = 0 option, so all computations are performed in one call.

```
          USE RGIVN_INT
          USE GDATA_INT
          USE WRRRN_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    LDB, LDCOEF, LDR, LDSCPE, LDX, NCOEF, NCOL, NDEP, NRX
          PARAMETER  (LDSCPE=1, NCOEF=5, NCOL=5, NDEP=1, NRX=13, &
                     LDB=NCOEF, LDCOEF=NCOEF, LDR=NCOEF, LDX=NRX)
     !
          INTEGER    I, IDEP, IIND, INDDEP(1), INDIND(1), &
                     IRANK, NOBS, NOUT, NRMISS, NVAR
          REAL       B(LDB,NDEP), D(NCOEF), DFE, R(LDR,NCOEF), &
                     SCPE(LDSCPE,NDEP), X(LDX,NCOL), XMAX(NCOEF), &
                     XMIN(NCOEF)
```

```
!
      CALL GDATA (5, X, NOBS, NVAR)
!
      IIND   = -4
      IDEP   = -1
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, r=r, d=d, &
                  irank=irank, dfe=dfe, scpe=scpe, nrmiss=nrmiss, &
                  xmin=xmin, xmax=xmax)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('R', R)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Regressor   XMIN     XMAX'
      DO 10  I=1, NCOEF
         WRITE (NOUT,'(1X,I5,2X,2F9.1)') I, XMIN(I), XMAX(I)
   10 CONTINUE
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'DFE = ', DFE, '  SCPE(1,1) = ', SCPE(1,1)
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
      B
1   62.41
2    1.55
3    0.51
4    0.10
5   -0.14
                    R
         1       2       3       4       5
1      3.6    26.9   173.6    42.4   108.2
2      0.0    20.4    12.3   -18.3   -14.2
3      0.0     0.0    52.5     1.1   -54.6
4      0.0     0.0     0.0    12.5   -12.9
5      0.0     0.0     0.0     0.0     3.4

Regressor       XMIN     XMAX
        1        1.0      1.0
        2        1.0     21.0
        3       26.0     71.0
        4        4.0     23.0
        5        6.0     60.0

IRANK =   5
DFE =     8.00000  SCPE(1,1) =     47.8637
NRMISS =   0
```

Example 2

The data for the second example are taken from Maindonald (1984, pages 203-204). The data are saved in the matrix X. Here, the data are input into RGIVN a row at a time. The data set is small for clarity. However, the approach is generally useful when the data set is large and the entire data set cannot be stored in X. A multivariate regression model containing two dependent variables and three independent variables is fit.

```
      USE RGIVN_INT
      USE WRRRN_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    INTCEP, LDB, LDR, LDSCPE, LDX, NCOEF, NCOL, NDEP, &
                 NIND, NOBS, J
      PARAMETER  (INTCEP=1, NCOL=5, NDEP=2, NIND=3, NOBS=9, &
                LDSCPE=NDEP, LDX=NOBS, NCOEF=INTCEP+NIND, LDB=NCOEF, &
                LDR=NCOEF)
!
      INTEGER    I, IDEP, IDO, IIND, INDDEP(1), INDIND(1), IRANK, &
                 NOUT, NRMISS, NROW
      REAL       B(LDB,NDEP), D(NCOEF), DFE, R(LDR,NCOEF), &
                 SCPE(LDSCPE,NDEP), TOL, X(LDX,NCOL), XMAX(NCOEF), &
                 XMIN(NCOEF)
!
      DATA (X(1,J),J=1,NCOL)/7.0, 5.0, 6.0, 7.0, 1.0/
      DATA (X(2,J),J=1,NCOL)/2.0, -1.0, 6.0, -5.0, 4.0/
      DATA (X(3,J),J=1,NCOL)/7.0, 3.0, 5.0, 6.0, 10.0/
      DATA (X(4,J),J=1,NCOL)/-3.0, 1.0, 4.0, 5.0, 5.0/
      DATA (X(5,J),J=1,NCOL)/2.0, -1.0, 0.0, 5.0, -2.0/
      DATA (X(6,J),J=1,NCOL)/2.0, 1.0, 7.0, -2.0, 4.0/
      DATA (X(7,J),J=1,NCOL)/-3.0, -1.0, 3.0, 0.0, -6.0/
      DATA (X(8,J),J=1,NCOL)/2.0, 1.0, 1.0, 8.0, 2.0/
      DATA (X(9,J),J=1,NCOL)/2.0, 1.0, 4.0, 3.0, 0.0/
!
      NROW = 1
      IIND = -NIND
      IDEP = -NDEP
      DO 10  I=1, 9
         IF (I .EQ. 1) THEN
            IDO = 1
         ELSE IF (I .EQ. 9) THEN
            IDO = 3
         ELSE
            IDO = 2
         END IF
      CALL RGIVN (X(I:I, 1:NCOL), IIND, INDIND, IDEP, INDDEP, &
               B, IDO=IDO, R=R, D=D,IRANK=IRANK, DFE=DFE,&
               SCPE=SCPE,NRMISS=NRMISS, xmin=xmin, xmax=xmax)

   10 CONTINUE
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('R', R)
      CALL WRRRN ('SCPE', SCPE)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Regressor   XMIN     XMAX'
      DO 20  I=1, NCOEF
         WRITE (NOUT,'(1X,I5,2X,2F9.1)') I, XMIN(I), XMAX(I)
   20 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'DFE = ', DFE
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
             B
           1       2
1    7.733  -1.633
2   -0.200   0.400
3    2.333   0.167
4   -1.667   0.667

                R
           1       2       3       4
1    3.00    6.00    3.00   12.00
2    0.00   10.00    4.00    2.00
3    0.00    0.00    4.00    2.00
4    0.00    0.00    0.00    6.00

          SCPE
           1       2
1     4.0    20.0
2    20.0   110.0

Regressor    XMIN     XMAX
     1        1.0      1.0
     2       -3.0      7.0
     3       -1.0      5.0
     4        0.0      7.0

IRANK =   4
DFE =     5.00000
NRMISS =   0
```

Example 3

The data for the third example are taken from Maindonald (1984, pages 104-106). The constant regressor and the independent variables X_1, X_2, and X_3 are linearly dependent

$$\left(X_3 = \tfrac{1}{2} + X_1 - \tfrac{1}{2}X_2\right)$$

```
      USE RGIVN_INT
      USE WRRRN_INT
      USE UMACH_INT
      INTEGER    INTCEP, LDB, LDR, LDSCPE, LDX, NCOEF, NCOL, NDEP, &
                 NIND, NOBS
      PARAMETER  (INTCEP=1, NCOL=5, NDEP=1, NIND=4, NOBS=9, &
                 LDSCPE=NDEP, LDX=NOBS, NCOEF=INTCEP+NIND, LDB=NCOEF,&
                 LDR=NCOEF)
!
      INTEGER    I, IDEP, IIND, INDDEP(1), INDIND(1), &
                 IRANK, NOUT, NRMISS, NROW
      REAL       B(LDB,NDEP), D(NCOEF), DFE, R(LDR,NCOEF), &
                 SCPE(LDSCPE,NDEP), TOL, X(LDX,NCOL), XMAX(NCOEF), &
                 XMIN(NCOEF)
!
      DATA (X(1,J),J=1,NCOL)/-1.0, 0.0, -0.5, 1.0, 0.0/
      DATA (X(2,J),J=1,NCOL)/3.0, 0.0, 3.5, 1.0, 0.0/
```

```
      DATA (X(3,J),J=1,NCOL)/2.0, -2.0, 3.5, -2.0, -2.0/
      DATA (X(4,J),J=1,NCOL)/-2.0, -1.0, -1.0, 1.0, 1.0/
      DATA (X(5,J),J=1,NCOL)/-1.0, 1.0, -1.0, -1.0, -1.0/
      DATA (X(6,J),J=1,NCOL)/3.0, 3.0, 2.0, 1.0, 3.0/
      DATA (X(7,J),J=1,NCOL)/2.0, 2.0, 1.5, 2.0, 4.0/
      DATA (X(8,J),J=1,NCOL)/-2.0, -1.0, -1.0, -1.0, -2.0/
      DATA (X(9,J),J=1,NCOL)/2.0, 1.0, 2.0, 1.0, 3.0/
!
      NROW = NOBS
      IIND = -NIND
      IDEP = -NDEP
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, r=r, d=d, &
                  irank=irank, dfe=dfe, scpe=scpe, nrmiss=nrmiss, &
                  xmin=xmin, xmax=xmax)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('R', R)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Regressor  Minimum  Maximum'
      DO 10  I=1, NCOEF
         WRITE (NOUT,'(1X,I5,2X,2F9.1)') I, XMIN(I), XMAX(I)
   10 CONTINUE
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'DFE = ', DFE, '  SCPE(1,1) = ', SCPE(1,1)
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
      B
1   0.056
2   0.167
3   0.500
4   0.000
5   1.000

                      R
        1       2       3       4       5
1   3.000   2.000   1.000   3.000   1.000
2   0.000   6.000   2.000   5.000   1.000
3   0.000   0.000   4.000  -2.000   2.000
4   0.000   0.000   0.000   0.000   0.000
5   0.000   0.000   0.000   0.000   3.000

Regressor  Minimum  Maximum
      1        1.0      1.0
      2       -2.0      3.0
      3       -2.0      3.0
      4       -1.0      3.5
      5       -2.0      2.0

IRANK =   4
DFE =     5.00000  SCPE(1,1) =     6.00000
NMISS =   0
```

RGLM

Fits a multivariate general linear model.

Required Arguments

X — |NROW| by NCOL matrix containing the data. (Input)

INDCL — Index vector of length NCLVAR containing the column numbers of X that are the classification variables. (Input)

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input)

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(NEF). (Input)
The first NVEF(1) elements give the column numbers of X for each variable in the first effect; the next NVEF(2) elements give the column numbers for each variable in the second effect; and so on. The last NVEF(NEF) elements give the column numbers for each variable in the last effect.

IDEP — Dependent variable option. (Input)
The absolute value of IDEP is the number of dependent (response) variables. The sign of IDEP specifies the following options:

IDEP	Action
< 0	The last -IDEP columns of X contain the dependent (response) variables. That is, columns NCOL + IDEP + 1, NCOL + IDEP + 2, …, NCOL contain the dependent variables.
> 0	The data for the IDEP dependent variables are in the columns of X whose column numbers are given by the elements of INDDEP.
= 0	There are no dependent variables. (Generally, this option is not used. However, it is possible to get the *R* matrix from a *QR* decomposition of a matrix of regressors in this way.)

INDDEP — Index vector of length IDEP containing the column numbers of *X* that are the dependent (response) variables. (Input, if IDEP is positive)
If IDEP is nonpositive, INDDEP is not referenced and can be a vector of length one.

MAXCL — An upper bound on the sum of the number of distinct values taken on by each classification variable. (Input)

B — NCOEF by | IDEP| matrix containing on return from the final invocation of this routine a least-squares solution $\hat{B}$ for the regression coefficients. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
Here, NCOEF = IRBEF(NEF + 1) - 1 is the number of coefficients in the model. If INTCEP = 1, row 1 is for the intercept. Column *j* is for the *j*-th dependent variable.

IDO	Action
1 or 2	A current least-squares solution is given by a solution *x* to the equation *R* * *x* = B
0 or 3	A least-squares solution for the regression coefficients is returned in B. Elements of the appropriate row(s) of B are set to 0.0 if linear dependence of the regressors is declared.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RGLM for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to RGLM will be made. Initialization and updating for the data in X are performed.
2	This is an intermediate invocation of RGLM, and updating for the data in X is performed.
3	This is the final invocation of this routine. Updating for the data in X and wrap-up computation are performed.

NROW — The absolute value of NROW is the number of rows of data currently input in X. (Input)
NROW may be positive, zero, or negative. Negative NROW means that the -NROW rows of data are to be deleted from some aspects of the analysis, and this should be done only if IDO is 2 or 3 and the wrap-up computations have not been performed. When a negative value is input for NROW, it is assumed that each of the -NROW rows of X has been input (with positive NROW) in previous invocations of RGIVN. Use of negative values of NROW should be made with care and with the understanding that XMIN, XMAX, and CLVAL cannot be updated properly in this case. It is also possible that a constant variable in the remaining data will not be recognized as such.
Default: NROW = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

NCLVAR — Number of classification variables. (Input)
Default: NCLVAR = size (INDCL,1).

NEF — Number of effects (sources of variation) in the model excluding error. (Input)
Default: NEF = size (NVEF,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies. If X(I, IFRQ) = 0.0, none of the remaining elements of row I of X are referenced and updating of statistics is skipped for row I.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

IDUMMY — Dummy variable option. (Input)
Default: IDUMMY = 1.
Some indicator variables are defined for the I-th class variable as follows: Let J = NCLVAL(1) + NCLVAL(2) + … + NCLVAL(I - 1). NCLVAL(I) indicator variables are defined such that for K = 1, 2, …, NCLVAL(I) the K-th indicator variable for observation number IOBS takes the value 1.0 if X(IOBS, INDCL(I)) = CLVAL(J + K) and equals 0.0 otherwise. Dummy variables are generated from these indicator variables, and restrictions may be applied as given by the following:

IDUMMY	Description
0	The NCLVAL(I) indicator variables are the dummy variables. The usual balanced-data restrictions on the regression parameters are applied as part of the wrap-up computations regardless of whether the data are balanced.
1	The NCLVAL(I) indicator variables are the dummy variables.
2	1 indicator variables are used as the dummy variables. The indicator variable associated with the class value given in the first row of X on the first invocation is omitted.

ICEN — Data centering option. (Input)
If INTCEP = 0, ICEN must equal 0.
Default: ICEN = 1.

ICEN	Action
0	No centering. This option should be used when (1) the data are already centered, (2) there is no intercept in the model, or (3) the regressors for a large percentage of the data are zero, and sparsity of the problem needs to be preserved in order that the fast Givens transformations are performed quickly
1	Variables are centered using the method of provisional means for improved accuracy of the computations. The final estimate for the intercept along with the *R* matrix are given for the uncentered data. This option is generally recommended.

TOL — Tolerance used in determining linear dependence. (Input)
For RGLM, TOL = 100 * AMACH(4) is a common choice. See the documentation for IMSL routine AMACH in *Reference Material*.
Default: TOL = 1.e-5 for single precision and 2.d –14 for double precision.

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3) NCLVAL(I) is the number of distinct values for the I-th classification variable.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the values of the classification variables. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
Since in general the length of CLVAL will not be known in advance, MAXCL (an upper bound for this length) should be used for purposes of dimensioning CLVAL. The first NCLVAL(1) elements contain the values of the first classification variable; the next NCLVAL(2) elements contain the values of the second classification variable; and so on. The last NCLVAL(NCLVAR) elements contain the values of the last classification variable. If IDUMMY = 0 or 1, the values are in ascending order for each classification variable. If IDUMMY = 2, the last value for each classification variable is the value associated with the indicator variable omitted from the model. The remaining values for each classification variable are in ascending order.

IRBEF — Index vector of length NEF + 1. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
For I = 1, 2, ..., NEF, rows IRBEF(I), IRBEF(I) + 1, ..., IRBEF(I + 1) - 1 of B correspond to the I-th effect.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

R — NCOEF by NCOEF upper triangular matrix containing, on return from the final invocation of this routine, the *R* matrix from a *QR* decomposition of the matrix of regressors. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
Upon completion of the wrap-up computations, a zero row indicates a nonfull rank model. If IDUMMY = 0, a negative diagonal element of *R* indicates that the associated row corresponds to a summation restriction.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

D — Vector of length NCOEF. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

IDO	Action
1 or 2	D contains the current scale factors associated with the fast Givens transformations. The current matrix of uncorrected sums of squares and crossproducts for the regressors can be found as $R^T \cdot \text{diag(D)} \cdot R$ where diag(D) is the diagonal matrix whose diagonal elements are the elements of D.
0 or 3	Each element of D is set to 1.0.

IRANK — Rank of *R*. (Output, if IDO = 0 or 3)
IRANK less than NCOEF indicates linear dependence of the regressors was declared.

DFE — Degrees of freedom for error on return from the final invocation of this routine. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
Prior to the final invocation, DFE is the sum of the frequencies.

SCPE — |IDEP| by |IDEP| matrix containing error (residual) sums of squares and crossproducts. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
SCPE(M, N) is the current sum of crossproducts of residuals for the M-th and N -th dependent variables.

LDSCPE — Leading dimension of SCPE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCPE = size (SCPE,1).

NRMISS — Number of rows of data encountered in calls to **RGLM** containing NaN (not a number) for the independent, dependent, weight, and/or frequency variables. (Output, if **IDO** = 0 or 1, Input/Output, if **IDO** = 2 or 3)
If a row of data contains NaN for any of these variables, that row is excluded from the computations.

XMIN — Vector of length **NCOEF** containing the minimum values for each of the regressors. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)

XMAX — Vector of length **NCOEF** containing the maximum values for each of the regressors. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)

FORTRAN 90 Interface

Generic: **CALL RGLM** (**X**, **INDCL**, **NVEF**, **INDEF**, **IDEP**, **INDDEP**, **MAXCL**, **B** [, ...])
Specific: The specific interface names are **S_RGLM** and **D_RGLM**.

FORTRAN 77 Interface

Single: **CALL RGLM** (**IDO**, **NROW**, **NCOL**, **X**, **LDX**, **INTCEP**, **NCLVAR**, **INDCL**, **NEF**, **NVEF**, **INDEF**, **IDEP**, **INDDEP**, **IFRQ**, **IWT**, **IDUMMY**, **ICEN**, **TOL**, **MAXCL**, **NCLVAL**, **CLVAL**, **IRBEF**, **B**, **LDB**, **R**, **LDR**, **D**, **IRANK**, **DFE**, **SCPE**, **LDSCPE**, **NRMISS**, **XMIN**, **XMAX**)
Double: The double precision name is **DRGLM**.

Description

Routine **RGLM** fits a multivariate linear regression model. (See the chapter introduction for a description of the multivariate linear regression model.) The routine **RGLM** is designed so that multiple invocations can be made. In this case, zero, one, or several rows of the data set can be input for each invocation of **RGLM** (with **IDO** = 1, 2, 2, ..., 2, 3). Alternatively, one invocation of **RGLM** (with **IDO** = 0) can be made with the entire data set contained in **X**. Routines RSTAT and RCASE can be invoked after the wrap-up computations are performed by **RGLM** to compute and print summary statistics and case statistics related to the fitted regression.

The data matrix can contain classification variables as well as continuous variables. The specification of a general linear model through the arguments **INTCEP**, **NCLVAR**, **INDCL**, **NEF**, **NVEF**, **INDEF** is discussed in the chapter introduction.

Regressors for effects composed solely of continuous variables are generated as powers and crossproducts. Consider a data matrix containing continuous variables as columns 3 and 4. The effect (3, 3) generates a regressor whose *i*-th value (*i* = 1, 2, ..., *n*) is the square of the *i*-th value in column 3. The effect (3, 4) generates a regressor whose *i*-th value is the product of the *i*-th value in column 3 with the *i*-th value in column 4.

Regressors for an effect containing a single classification variable are generated using indicator variables. Let the classification variable *A* take on values $a_1, a_2, \ldots, a_n$ (stored in that order in **CLVAL**). From this classification variable, *n* indicator variables I_k are created. For $k = 1, 2, \ldots, n$ we have

$$I_k = \begin{cases} 1 & \text{if } A = a_k \\ 0 & \text{otherwise} \end{cases}$$

For each classification variable, another set of variables is created from the indicator variables. We call these new variables *dummy variables*. Dummy variables are generated from the indicator variables in one of two manners: (1) the dummies are the *n* indicator variables, or (2) the dummies are the first $n - 1$ indicator variables. In particular, for **IDUMMY** = 0 or **IDUMMY** = 1, the dummy variables are $A_k = I_k$ ($k = 1, 2, \ldots, n$). For **IDUMMY** = 2, the dummy variables are $A_k = I_k$ ($k = 1, 2, \ldots, n - 1$).

Let m_j be the number of dummies generated for the *j*-th classification variable. Suppose there are two classification variables *A* and *B* with dummies $A_1, A_2, \ldots, A_{m1}$ and $B_1, B_2, \ldots, B_{m2}$, respectively. The regressors generated for an effect composed of two classification variables *A* and *B* are

$$\begin{aligned} A \otimes B &= \left(A_1, A_2, \ldots, A_{m_1}\right) \otimes \left(B_1, B_2, \ldots, B_{m_2}\right) \\ &= \left(A_1B_1, A_1B_2, \ldots, A_1B_{m_2}, A_2B_1, A_2B_2, \ldots, A_2B_{m_2}, A_{m_1}B_1, A_{m_1}B_2, \ldots, A_{m_1}B_{m_2}\right) \end{aligned}$$

More generally, the regressors generated for an effect composed of several classification variables and several continuous variables are given by the Kronecker products of variables, where the order of the variables is specified in **INDEF**. Consider a data matrix containing classification variables in columns 1 and 2 and continuous variables in columns 3 and 4. Label these four columns *A*, *B*, X_1, and X_2, respectively. The regressors generated by the effect (1, 2, 3, 3, 4) are $A \otimes B \otimes X_1X_1X_2$.

Routine **RGLM** performs an orthogonal reduction of the matrix of regressors to upper triangular form. The reduction is based on fast Givens transformations. (See routines **SROTMG** and **SROTM**, Golub and Van Loan 1983, pages 156-162, Gentleman 1974.) This method has two main advantages: (1) the loss of accuracy resulting from forming the crossproduct matrix used in the normal equations is avoided, and (2) data can be conveniently added or deleted to take advantage of the previous computations performed.

With **ICEN** = 1, the current means of the regressors and dependent variables are used to center the data for improved accuracy. Let x_i be a column vector containing the *i*-th row of data for the regressors. Let $\bar{x}_i$ represent the mean vector for the regressors given the data for observations
1, 2, ..., *i*. The mean vector is defined to be

$$\bar{x}_i = \frac{\sum_{j=1}^{i} w_j f_j x_j}{\sum_{j=1}^{i} w_j f_j}$$

where the w_j's and f_j's are the weights and frequencies, respectively. The *i*-th row of data has $\bar{x}_i$ subtracted from it, and then, $w_i f_i$ is multiplied by the factor a_i/a_{i-1} where

$$a_i = \sum_{j=1}^{i} w_j f_j$$

Although a crossproduct matrix is not computed, the validity of this centering operation can be seen from the following formula for the sum of squares and crossproducts matrix:

$$\sum_{i=1}^{n} w_i f_i (x_i - \bar{x}_n)(x_i - \bar{x}_n)^T = \sum_{i=2}^{n} \frac{a_i}{a_{i-1}} w_i f_i (x_i - \bar{x}_i)(x_i - \bar{x}_i)^T$$

An orthogonal reduction on the centered matrix is computed. When wrap-up computations (**IDO** = 3 or **IDO** = 0) are performed, the first rows of *R* and *B* are updated so that they reflect the statistics for the original (uncentered) data. This means that the *R* matrix and the estimate of the intercept are for the uncentered data.

If the *i*-th regressor is a linear combination of the first *i* - 1 regressors, the *i*-th diagonal element of *R* will be close to zero (exactly zero if infinite precision arithmetic could be used) prior to the wrap-up computations. When performing the wrap-up computations, **RGLM** checks sequentially for linear dependent regressors. Linear dependence of the regressors is declared if any of the following three conditions is satisfied:

- A regressor equals zero, as determined from **XMIN** and **XMAX**.
- Two or more regressors are constant, as determined from **XMIN** and **XMAX**.
- $\sqrt{1 - R^2_{i \cdot 1,2,\ldots i-1}}$ is less than or equal to **TOL**. Here $R_{i \cdot 1, 2, \ldots, i-1}$ is the multiple correlation coefficient of the *i*-th regressor with the first *i* - 1 regressors. If no intercept is in the model (**INTCEP** = 0) the 'multiple correlation' coefficient is computed without adjusting for the mean.

When a dependence is declared, *R* is changed in the wrap-up computations so as to reflect the deletion of the *i*-th regressor from the model. On completion of the wrap-up computations, if the*i*-th regressor is declared to be dependent upon the previous *i* - 1 regressors, then the *R* and *B* matrices will have all elements in their *i*-th rows set to zero.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2LM/DR2LM**. The reference is:

```
CALL R2LM (IDO, NROW, NCOL, X, LDX, INTCEP, NCLVAR, INDCL, NEF, NVEF, INDEF,
    IDEP, INDEP, IFRQ, IWT, IDUMMY, ICEN, TOL, MAXCL, NCLVAL, VAL, IRBEF, B, LDB, R,
    LDR, D, IRANK, DFE, SCPE, LDSCPE, NRMISS, XMIN, XMAX, IWK, WK)
```

The additional arguments are as follows:

IWK — Work vector of length max(MAXB, NCLVAR).

WK — Work vector of length MAXB + |IDEP| + 2.

2. Informational errors

Type	Code	Description
4	1	Negative weight encountered.
4	2	Negative frequency encountered.
4	7	MAXCL is too small. Increase MAXCL and the dimension of CLVAL.
4	8	LDB or LDR is too small. One or more of the dimensions of B, R, D, XMIN, and XMAX must be increased.

3. Let the data matrix **X** = (A, B, X_1, Y) where A and B are classification variables, X_1 is a continuous independent variable, and Y is a response variable. The model containing an intercept and the effects A, B, AB, X_1, AX_1, BX_1, and ABX_1 is specified as follows: **INTCEP** = 1, **NCLVAR** = 2, **INDCL** = (1, 2), **NEF** = 7, **NVEF** = (1, 1, 2, 1, 2, 2, 3), **INDEF** = (1, 2, 1, 2, 3, 1, 3, 2, 3, 1, 2, 3), **IDEP** = 1, and **INDDEP** = (4).

   For this model suppose **NCLVAL**(1) = 2, **NCLVAL**(2) = 3, and **CLVAL** = (1.0, 2.0, 1.0, 2.0, 3.0). Let A_1, A_2, B_1, B_2, and B_3, be the associated indicator variables. For each **IDUMMY** option the regressors following the intercept in their order of appearance in the model are given as follows:

IDUMMY	Regressors
0 or 1	A_1, A_2, B_1, B_2, B_3, A_1B_1, A_1B_2, A_1B_3, A_2B_1, A_2B_2, A_2B_3, X_1, A_1X_1, $A_2X_1B_1X_1$, B_2X_1, B_3X_1, $A_1B_1X_1$, $A_1B_2X_1$, $A_1B_3X_1$, $A_2B_1X_1$, $A_2B_2X_1$, $A_2B_3X_1$
2	A_1, B_1, B_2, A_1B_1, A_1B_2, X_1, A_1X_1, B_1X_1, B_2X_1, $A_1B_1X_1$, $A_1B_2X_1$

   Within a group of regressors corresponding to an interaction effect, the indicator variables composing the regressors change most rapidly for the last classification variable, change next most rapidly for the next to last classification variable, etc.

4. If **NROW** is negative, no downdating of **XMIN**, **XMAX**, **NCLVAL**, and **CLVAL** can occur.

Examples

Example 1

A one-way analysis of covariance model is fitted to the turkey data discussed by Draper and Smith (1981, pages 243 - 249). The response variable is turkey weight *y* (in pounds). There are three groups of turkeys corresponding to the three states where they were reared. The age of a turkey (in weeks) is the covariate. The explanatory variables are group, age, and interaction. The model is

$$y_{ij} = \mu + \alpha_i + \beta x_{ij} + \beta_i x_{ij} + \varepsilon_{ij} \quad i = 1, 2, 3; j = 1, 2, \ldots, n_i$$

where $\alpha_3 = 0$ and $\beta_3 = 0$. Here, the **IDUMMY** = 2 option is used. The fitted model gives three separate lines, one for each state where the turkeys were reared.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
!                                SPECIFICATIONS FOR PARAMETERS
      INTEGER    IDEP, INTCEP, LDB, LDR, LDSCPE, LDX, MAXB, MAXCL, &
                 NCLVAR, NCOL, NEF, NROW
      PARAMETER  (IDEP=1, INTCEP=1, LDX=13, MAXB=6, MAXCL=3, NCLVAR=1, &
                 NCOL=3, NEF=3, NROW=13, LDB=MAXB, LDR=MAXB, &
                 LDSCPE=IDEP)
!
      INTEGER    I, IDUMMY, INDCL(NCLVAR), INDDEP(IDEP), &
                 INDEF(4), IRANK, IRBEF(NEF+1), J, &
                 NCLVAL(NCLVAR), NCOEF, NOUT, NRMISS, NVEF(NEF)
      REAL       B(LDB,IDEP), CLVAL(MAXCL), D(MAXB), DFE, &
                 R(LDR,MAXB), SCPE(LDSCPE,IDEP), TOL, X(LDX,NCOL), &
                 XMAX(MAXB), XMIN(MAXB)
      CHARACTER  CLABEL(7)*6, RLABEL(1)*4
!
      DATA (X(1,J),J=1,3)  /25, 13.8, 3/
      DATA (X(2,J),J=1,3)  /28, 13.3, 1/
      DATA (X(3,J),J=1,3)  /20,  8.9, 1/
      DATA (X(4,J),J=1,3)  /32, 15.1, 1/
      DATA (X(5,J),J=1,3)  /22, 10.4, 1/
      DATA (X(6,J),J=1,3)  /29, 13.1, 2/
      DATA (X(7,J),J=1,3)  /27, 12.4, 2/
      DATA (X(8,J),J=1,3)  /28, 13.2, 2/
      DATA (X(9,J),J=1,3)  /26, 11.8, 2/
      DATA (X(10,J),J=1,3) /21, 11.5, 3/
      DATA (X(11,J),J=1,3) /27, 14.2, 3/
      DATA (X(12,J),J=1,3) /29, 15.4, 3/
      DATA (X(13,J),J=1,3) /23, 13.1, 3/
      DATA INDCL/3/, NVEF/1, 1, 2/, INDEF/3, 1, 1, 3/, INDDEP/2/
      DATA CLABEL/' ', 'MU', 'ALPHA1', 'ALPHA2', 'BETA', 'BETA1', &
           'BETA2'/
      DATA RLABEL/'NONE'/
!
      IDUMMY = 2
      CALL RGLM (X, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, B, &
                 idummy=idummy, nclvar=nclvar, nclval=nclval, &
                 clval=clval, irbef=irbef, r=r, d=d, irank=irank, &
                 dfe=dfe, scpe=scpe, nrmiss=nrmiss, xmin=xmin, &
```

```
                  xmax=xmax)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  DFE = ', DFE, '  '// &
                     'SCPE(1,1) = ', SCPE(1,1)
      J = 0
      DO 10  I=1, NCLVAR
         CALL WRRRN ('Class values', CLVAL((J+1):), 1, NCLVAL(I), 1)
         J = J + NCLVAL(I)
   10 CONTINUE
      NCOEF = IRBEF(NEF+1) - 1
      CALL WRRRN ('XMIN', XMIN, 1, NCOEF, 1)
      CALL WRRRN ('XMAX', XMAX, 1, NCOEF, 1)
      CALL WRIRN ('IRBEF', IRBEF, 1, NEF+1, 1)
      CALL WRRRN ('R-MATRIX', R)
      CALL WRRRL ('B', B, RLABEL, CLABEL, 1, NCOEF, 1)
!
      END
```

Output

```
 NRMISS =   0
 IRANK =   6  DFE =     7.00000  SCPE(1,1) =    0.706176

       Class values
     1       2       3
 1.000   2.000   3.000

                    XMIN
    1       2       3       4       5       6
 1.00    0.00    0.00   20.00    0.00    0.00

                    XMAX
    1       2       3       4       5       6
 1.00    1.00    1.00   32.00   32.00   29.00

     IRBEF
 1   2   3   4
 2   4   5   7

                    R-MATRIX
         1       2       3       4       5       6
 1    3.61    1.11    1.11   93.47   28.29   30.51
 2            1.66   -0.74   -1.02   42.43  -20.34
 3                    1.49    3.73    0.00   40.99
 4                           11.66    7.80    0.43
 5                                    5.49   -0.61
 6                                            2.11

                                 B
    MU      ALPHA1      ALPHA2        BETA       BETA1       BETA2
 2.475      -3.454      -2.775       0.445     0.06104       0.025
```

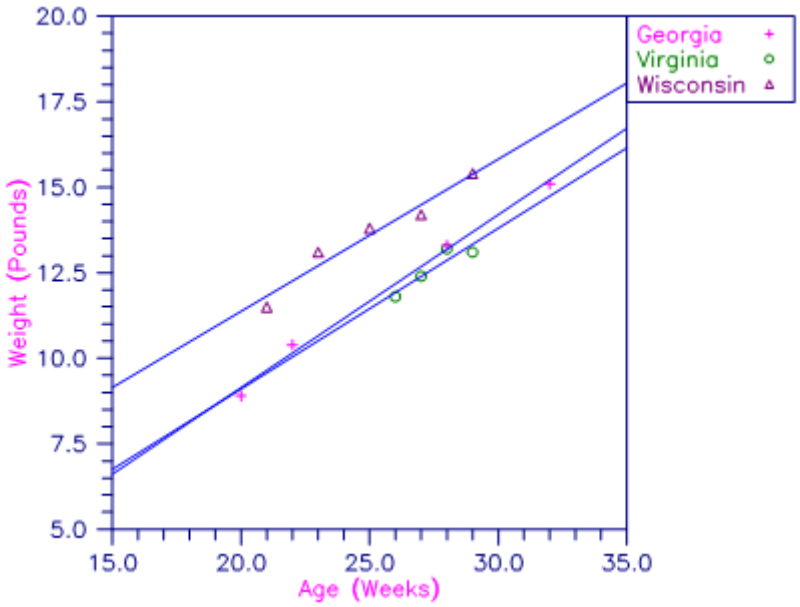


Figure 4, Plot of Turkey Weights and Fitted Lines by State

Example 2

A two-way analysis-of-variance model is fitted to balanced data discussed by Snedecor and Cochran (1967, Table 12.5.1, page 347). The responses are the weight gains (in grams) of rats fed diets varying in two components — level of protein and source of protein. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \quad i = 1, 2; j = 1, 2, 3; k = 1, 2, \ldots, 10$$

where

$$\sum_{i=1}^{2} \alpha_i = 0; \sum_{j=1}^{3} \beta_j = 0; \sum_{i=1}^{2} \gamma_{ij} = 0 \text{ for } j = 1, 2, 3; \text{ and } \sum_{j=1}^{3} \gamma_{ij} = 0 \text{ for } i = 1, 2$$

Here, the **IDUMMY** = 0 option is used.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    IDEP, LDB, LDR, LDSCPE, LDX, LINDEF, MAXB, MAXCL, &
                     NCLVAR, NCOL, NEF, NROW
          PARAMETER  (IDEP=1, LINDEF=4, MAXB=12, MAXCL=5, NCLVAR=2, &
                     NCOL=3, NEF=3, NROW=60, LDB=MAXB, LDR=MAXB, &
                     LDSCPE=IDEP, LDX=NROW)
    !
```

```
      INTEGER    I, IDUMMY, INDCL(NCLVAR), INDDEP(IDEP), &
                 INDEF(LINDEF), INTCEP, IRANK, IRBEF(NEF+1), J, &
      NCLVAL(NCLVAR), NCOEF, NOUT, NRMISS, NVEF(NEF)

      REAL       B(LDB,IDEP), CLVAL(MAXCL), D(MAXB), DFE, &
                 R(LDR,MAXB), SCPE(LDSCPE,IDEP), X(LDX,NCOL), &
                 XMAX(MAXB), XMIN(MAXB)
      CHARACTER  CLABEL(MAXB+1)*7, RLABEL(1)*4
!
      DATA X/73.0, 102.0, 118.0, 104.0, 81.0, 107.0, 100.0, 87.0, &
           117.0, 111.0, 98.0, 74.0, 56.0, 111.0, 95.0, 88.0, 82.0, &
           77.0, 86.0, 92.0, 94.0, 79.0, 96.0, 98.0, 102.0, 102.0, &
           108.0, 91.0, 120.0, 105.0, 90.0, 76.0, 90.0, 64.0, 86.0, &
           51.0, 72.0, 90.0, 95.0, 78.0, 107.0, 95.0, 97.0, 80.0, &
           98.0, 74.0, 74.0, 67.0, 89.0, 58.0, 49.0, 82.0, 73.0, 86.0, &
           81.0, 97.0, 106.0, 70.0, 61.0, 82.0, 30*1.0, 30*2.0, &
           10*1.0, 10*2.0, 10*3.0, 10*1.0, 10*2.0, 10*3.0/
      DATA INDCL/2, 3/, NVEF/1, 1, 2/, INDEF/2, 3, 2, 3/, INDDEP/1/
      DATA CLABEL/' ', 'MU', 'ALPHA1', 'ALPHA2', 'BETA1', 'BETA2', &
           'BETA3', 'GAMMA11', 'GAMMA12', 'GAMMA13', 'GAMMA21', &
           'GAMMA22', 'GAMMA23'/
      DATA RLABEL/'NONE'/
      !
      IDUMMY = 0
      CALL RGLM (X, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, B, &
                 idummy=idummy, nclvar=nclvar, nclval=nclval, &
                 clval=clval, irbef=irbef, r=r, d=d, irank=irank, &
                 dfe=dfe, scpe=scpe, nrmiss=nrmiss, xmin=xmin, &
                 xmax=xmax)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  DFE = ', DFE, '  '// &
                     'SCPE(1,1) = ', SCPE(1,1)
      J = 0
      DO 10  I=1, NCLVAR
         CALL WRRRN ('Class Values', CLVAL((J+1):), 1, NCLVAL(I), 1)
         J = J + NCLVAL(I)
   10 CONTINUE
      NCOEF = IRBEF(NEF+1) - 1
      CALL WRRRN ('XMIN', XMIN, 1, NCOEF, 1)
      CALL WRRRN ('XMAX', XMAX, 1, NCOEF, 1)
      CALL WRIRN ('IRBEF', IRBEF, 1, NEF+1, 1)
      CALL WRRRN ('R-MATRIX', R, NRA=NCOEF, NCA=NCOEF, ITRING=1)
      CALL WRRRL ('B', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(2W10.4)')
!
      END
```

Output

```
 NRMISS =   0
 IRANK =   12  DFE =     54.0000  SCPE(1,1) =     11586.0

 Class Values
     1       2
 1.000   2.000

     Class Values
     1       2       3
```

```
1.000   2.000   3.000

                                    XMIN
    1      2      3      4      5      6      7      8      9     10
1.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000

   11     12
0.000  0.000

                                    XMAX
    1      2      3      4      5      6      7      8      9     10
1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000  1.000

   11     12
1.000  1.000

     IRBEF
1    2    3    4
2    4    7   13

                                  R-MATRIX
        1      2      3      4      5      6      7      8      9
1   7.746  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
2         -1.000 -1.000  0.000  0.000  0.000  0.000  0.000  0.000
3                 7.746  0.000  0.000  0.000  0.000  0.000  0.000
4                       -1.000 -1.000 -1.000  0.000  0.000  0.000
5                              6.325  3.162  0.000  0.000  0.000
6                                     5.477  0.000  0.000  0.000
7                                            -1.000  0.000  0.000
8                                                   -1.000  0.000
9                                                           -1.000

       10     11     12
1   0.000  0.000  0.000
2   0.000  0.000  0.000
3   0.000  0.000  0.000
4   0.000  0.000  0.000
5   0.000  0.000  0.000
6   0.000  0.000  0.000
7  -1.000  0.000  0.000
8   0.000 -1.000  0.000
9   0.000  0.000 -1.000
10 -1.000 -1.000 -1.000
11         6.325  3.162
12                5.477

                                      B
        MU      ALPHA1      ALPHA2       BETA1       BETA2       BETA3
     87.87       7.267      -7.267       1.733      -2.967       1.233

   GAMMA11     GAMMA12     GAMMA13     GAMMA21     GAMMA22     GAMMA23
     3.133      -6.267       3.133      -3.133       6.267      -3.133
```

RLEQU

Fits a multivariate linear regression model with linear equality restrictions *H B* = *G* imposed on the regression parameters given results from routine RGIVN after IDO = 1 and IDO = 2 and prior to IDO = 3.

Required Arguments

H — NH by NCOEF matrix with the *i*-th row specifying a linear combination of the regression parameters for the *i*-th row in the restriction *H B* = *G*. (Input)

B — NCOEF by NDEP matrix containing on return from the final invocation of this routine a least-squares solution for the regression coefficients in the restricted model. (Input/Output)
Invocation of RLEQU with INVOKE = 0 and 1 requires as input the B matrix from RGIVN after RGIVN's invocation with IDO = 1 and IDO = 2 and prior to IDO = 3 with NROW = 0. After the wrap-up computations are computed by RLEQU, B contains a least-squares solution for the regression coefficients in the restricted model.

R — NCOEF by NCOEF upper triangular matrix containing, on return from the final invocation of this routine, the *R* matrix from the restricted regression fit. (Input/Output)
Invocation of RLEQU with INVOKE = 0 and 1 requires as input the *R* matrix from RGIVN after RGIVN's invocation with IDO = 1 and IDO = 2 and prior to IDO = 3 with NROW = 0. After the wrap-up computations are computed by RLEQU, *R* contains the *R* matrix from the restricted regression fit. Elements to the right of a diagonal element of *R* (that is zero) are also zero. A zero row in *R* indicates a nonfull rank model. Each row of *R* corresponding to a restriction has a corresponding diagonal element that is negative. Each remaining row of *R* has a corresponding diagonal element that is positive.

D — Vector of length NCOEF containing scale factors associated with the fast Givens transformations. (Input/Output)
Invocation of RLEQU with INVOKE = 0 and 1 requires as input the D from RGIVN after RGIVN's invocation with IDO = 1 and IDO = 2 and prior to IDO = 3 with NROW = 0. After the wrap-up computations are computed by RLEQU, D contains all its elements set to 1.0.

DFE — Degrees of freedom for error for the restricted model on return from the final invocation of this routine. (Input/Output)
Prior to the final invocation of this routine, DFE contains the sum of the frequencies. Invocation of RLEQU with INVOKE = 0 and 1 requires as input the DFE from RGIVN after RGIVN's invocation with IDO = 1 and IDO = 2 and prior to IDO = 3 with NROW = 0.

SCPE — NDEP by NDEP matrix containing error (residual) sums of squares and crossproducts for the restricted model. (Input/Output)
SCPE(M, N) is the current sum of crossproducts of residuals for the M-th and N-th dependent variables. Invocation of RLEQU with INVOKE = 0 and 1 requires as input the SCPE matrix from RGIVN after RGIVN's invocation with IDO = 1 and IDO = 2 and prior to IDO = 3 with NROW = 0.

Optional Arguments

INVOKE — Invocation option. (Input)
Default: INVOKE = 0.

INVOKE	Action
0	This is the only invocation of RLEQU. All the restrictions are input at once.
1	This is the first invocation, and additional calls to RLEQU will be made. Initialization and updating for the restrictions *H B* = *G* are performed.
2	This is an intermediate invocation of RLEQU, and updating for the restrictions *H B* = *G* is performed.
3	This is the final invocation of this routine. Updating for the restrictions *H B* = *G* is performed, and wrap-up computations are performed.

NH — Number of rows in the restriction *H B* = *G*. (Input)
Default: NH = size (H,1).

NCOEF — Number of coefficients in the regression equation for each dependent variable. (Input)
Default: NCOEF = size (H,2).

LDH — Leading dimension of H exactly as specified in the dimension statement of the calling program. (Input)
Default: LDH = size (H,1).

IG — Option for G matrix. (Input)
Default: IG = 0.

IG	Restrictions
0	*H B* = 0
1	*H B* = *G*

NDEP — Number of dependent (response) variables. (Input)
Default: NDEP = size (B,2).

G — NH by NDEP matrix containing the right-hand side of the restriction
H B = *G*. (Input, if IG = 1)
If IG = 0, G is not referenced and can be a 1 by 1 array.
Default: G is a 1 by 1 array.

LDG — Leading dimension of G exactly as specified in the dimension statement in the calling program. (Input)
Default: LDG = size(G, 1).

TOL — Tolerance used in determining linear dependence. (Input)
For RLEQU, TOL = 100.0 * AMACH(4) is a common choice. See the documentation for IMSL routines AMACH in *Reference Material*.
Default: TOL = 1.e-5 for single precision and 2.d-14 for double precision.

LDB — Leading dimension B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

IRANKR — Rank of matrix *R*. (Output, if INVOKE = 0 or 3)

LDSCPE — Leading dimension of SCPE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCPE = size (SCPE,1).

IRANKH — Rank of matrix *H*. (Output)

FORTRAN 90 Interface

Generic: CALL RLEQU (H, B, R, D, DFE, SCPE[, …])
Specific: The specific interface names are S_RLEQU and D_RLEQU.

FORTRAN 77 Interface

Single: CALL RLEQU (INVOKE, NH, NCOEF, H, LDH, IG, NDEP, G, LDG, TOL, B, LDB, R, LDR, D, IRANKR, DFE, SCPE, LDSCPE, IRANKH)
Double: The double precision name is DRLEQU.

Description

Routine **RLEQU** requires the output from routine **RGIVN** after **RGIVN** has been invoked with **IDO** = 1 and **IDO** = 2 and prior to **IDO** = 3 with **NROW** = 0. Similarly, **RLEQU** can use results from IMSL routine **RGLM**.

The routine **RLEQU** is designed so that you can partition a large number of restrictions, as might arise in classification models, into several groups of restrictions (each requiring less space) and make multiple calls to **RLEQU** (with **INVOKE** = 1, 2, 2, …, 3). Alternatively, one invocation of **RLEQU** (with **INVOKE** = 0) can be made with all the restrictions contained in **H** and **G**.

After the wrap-up computations are performed by **RLEQU**, routines **RSTAT** and **RCASE** can be used to compute and print summary statistics and case statistics related to the fitted regression.

Routine **RGIVN** (or **RGLM**) together with routine **RLEQU** compute estimates of the regression coefficients in a multivariate general linear model $Y = X B + \mathbf{E}$ subject to $H B = G$. Here, Y is the $n \times q$ matrix of responses, X is the $n \times p$ matrix of regressors, B is the $p \times q$ matrix of regression coefficients, and $\mathbf{E}$ is the $n \times q$ matrix of errors whose q-dimensional rows are identically and independently distributed multivariate normal with mean vector 0 and variance-covariance matrix $\mathbf{\Sigma}$. The restriction is specified by the $h \times p$ matrix H and the $h \times q$ matrix G.

Previously, algorithms for solving the restricted least-squares problem were based on solving the following equations (Rao, 1973, page 232):

$$X^T X\hat{B} + H^T \Lambda = X^T Y$$
$$H\hat{B} = G$$

Routine **RLEQU** is based on an orthogonal reduction of X to upper triangular form. Fast Givens transformations with modifications described by Stirling (1981) for incorporating restrictions are used. This method has two main advantages: (1) the loss of accuracy resulting from forming $X^T X$ and $X^T Y$ is avoided, and (2) restrictions can be conveniently added so as to take advantage of the previous computations performed.

The method conceptually treats restrictions as observations with zero error variance. Fast Givens transformations as described by Golub and Van Loan (1983, pages 156-162) are used. The modification to the matrix R from the unrestricted fit to form a modified

$$\tilde{R}$$

for the restricted fit is as follows:

1. If the leading nonzero element of the first restriction is small (as determined by **TOL** times a computed scale factor), the element is set to zero.

2. Let *i* be the index of the leading nonzero element in the modified first restriction. Replace row *i* of *R* by the restriction. Flag the *i*-th row as a restriction. Use the restriction to reduce the first nonzero element of the row that was removed from *R* to zero. Incorporate the row that has been reduced by the restriction into the remaining rows of *R* as if it were new data.

3. Add additional restrictions into *R* by using Gaussian elimination, with the rows in *R* corresponding to restrictions, to reduce the restriction to a form so that it can replace a row of *R* corresponding to data and preserve the upper triangular structure of *R*. While performing the Gaussian elimination, set small nonzero elements (as determined by **TOL** times a computed scale factor) of the reduced restriction to zero, so that errors from inexact computer arithmetic are not incorporated as a new restriction. Flag the row as a restriction. Use the restriction to reduce the first nonzero element of the row that was removed from *R* to zero. Incorporate the row that has been reduced by the restriction into the remaining rows of *R* as if it were new data.

4. After all the data and restrictions are incorporated, the *i*-th row of *R* (where *i* ranges over each row of *R* corresponding to a linearly independent constraint) is used to zero out elements of *R* in the *i*-th column of the previous rows of *R* that correspond to data. Although this step is not required to get a least-squares solution, Sallas (1988) recommends this step so that the rows and columns of

$$\tilde{R}$$

corresponding to data form the *R* matrix for the reduced model that arises from expressing some regression parameters, β_i, in terms of other regression parameters, $\beta_j (j > i)$.

Linear dependence of the regressors in the reduced model is then checked as part of the wrap-up computations, using the rows and columns of *R* corresponding to the reduced model. The check is complicated somewhat by the fact that a regressor could become zero in the reduced model, but because of the finite precision of computer arithmetic, the regressor is not exactly zero. Let d_i equal the *i*-th diagonal element of $X^T X$, and let

$$\tilde{d}_i$$

equal the corresponding diagonal from the crossproducts matrix for the reduced model. Linear dependence of regressors in the reduced model is declared if

$$\sqrt{1 - R^2_{i \cdot 1,2,\ldots i-1}}$$

is less than or equal to **TOL** or if

$$\sqrt{\left(1 - R^2_{i \cdot 1,2,\ldots i-1}\right)\tilde{d}_i / d_i}$$

is less than or equal to **TOL**. (The last check is designed to detect a zero regressor in the reduced model.) Here,

$$R^2_{i\cdot 1,2,\ldots i-1}$$

is the square of the "multiple correlation" coefficient of the *i*-th regressor in the reduced model with the first *i* - 1 regressors in the reduced model. The "multiple correlation" coefficient is computed using the regressors in the reduced model and adjusted for the mean only if the incorporated restrictions have that effect.

When a linear dependence is declared, *R* is changed so as to reflect the deletion of the *i*-th regressor from the model. On completion of the wrap-up computations, the rows of *R* can be partitioned into three classes according to the sign of the corresponding diagonal element:

1. A positive diagonal element means the row/column corresponds to data for regressors in the reduced model.
2. A negative diagonal element means the row corresponds to a linearly independent restriction imposed on the regression parameters by *H B* = *G*.
3. A zero diagonal element means a linear dependence in the reduced model was declared. The regression coefficients in the corresponding row of

   $$\hat{B}$$

   are set to zero. This represents an arbitrary restriction that is imposed to obtain a solution for the regression coefficients. The elements of the corresponding row of *R* are also set to zero.

Redundant restrictions on the regression parameters are frequently specified in general linear models. Routine RLEQU permits redundant restrictions and returns the rank of *H*. An informational error is issued if inconsistent restrictions are detected.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2EQU/DR2EQ. The reference is:

   ```
   CALL R2EQU (INVOKE, NH, NCOEF, H, LDH, IG, NDEP, G, LDG, TOL, B, LDB, R, LDR, D,
       IRANKR, DFE, SCPE, LDSCPE, IRANKH, WK)
   ```

   The additional argument is:

   WK — Work vector of length NCOEF + NDEP.

2. Informational error

Type	Code	Description
3	1	The restrictions are inconsistent.

3. The results of routine RGLM can be used as input to RLEQU in place of the results of routine RGIVN.

Examples

Example 1

A grafted polynomial (spline function) is fit to data discussed by Fuller (1976, pages 396-398). The data set contains the response variable y measuring the annual wheat yield (in bushels per acre) for the years 1908 through 1971. In order to fit the trend, Fuller fits a function that is constant for the first 25 years, increases at a quadratic rate until 1961, and is linear for the last 10 years. This trend is represented by the function $f(t)$ where

$$f(t) = \begin{cases} \beta_1 & \text{if } 1 \le t \le 25 \\ \beta_2 + \beta_3 t + \beta_4 t^2 & \text{if } 25 \le t \le 54 \\ \beta_5 + \beta_6 t & \text{if } 54 \le t \le 64 \end{cases}$$

where t = 1 for 1908.

In order to fit a smooth function to the data, we require both continuity and differentiability. This imposes four restrictions on the coefficients given as follows:

1. $\beta_1 - \beta_2 - 25\,\beta_3 - 25^2\,\beta_4 = 0$
2. $\beta_2 + 54\,\beta_3 + 54^2\,\beta_4 - \beta_5 - 54\,\beta_6 = 0$
3. $\beta_3 + 50\,\beta_4 = 0$
4. $\beta_3 + 108\,\beta_4 - \beta_6 = 0$

The example program first calls routine **RGIVN** with **IDO** = 1, which specifies that initialization and updating for the data are performed and wrap-up computations are not performed. This intermediate output from **RGIVN** along with the restrictions is the input to **RLEQU** .

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    IDEP, LDB, LDG, LDH, LDR, LDSCPE, LDX, NCOEF, NH, &
                     NOBS, NVAR, J
          PARAMETER  (IDEP=1, LDG=1, NCOEF=6, NH=4, NOBS=64, NVAR=7, &
                     LDB=NCOEF, LDH=NH, LDR=NCOEF, LDSCPE=IDEP, LDX=NOBS)
    !
          INTEGER    I, IDO, IG, INDDEP(IDEP), INDIND(NCOEF), INTCEP, &
                     IRANK, IRANKH, IRANKR, ICEN, NOUT, NRMISS
          REAL       B(LDB,IDEP), D(NCOEF), DFE, G(LDG,IDEP), &
                     H(LDH,NCOEF), R(LDR,NCOEF), SCPE(LDSCPE,IDEP), &
                     X(LDX,NVAR), XMAX(NCOEF), XMIN(NCOEF)
          CHARACTER*4 RLABEL(1), CLABEL(1)
    !
          DATA INDIND/1, 2, 3, 4, 5, 6/, INDDEP/7/
          DATA X/384*0.0, 14.3, 15.5, 13.7, 12.4, 15.1, 14.4, 16.1, 16.7, &
                 11.9, 13.2, 14.8, 12.9, 13.5, 12.7, 13.8, 13.3, 16.0, 12.8, &
                14.7, 14.7, 15.4, 13.0, 14.2, 16.3, 13.1, 11.2, 12.1, 12.2, &
```

```
            12.8, 13.6, 13.3, 14.1, 15.3, 16.8, 19.5, 16.4, 17.7, 17.0, &
             17.2, 18.2, 17.9, 14.5, 16.5, 16.0, 18.4, 17.3, 18.1, 19.8, &
             20.2, 21.8, 27.5, 21.6, 26.1, 23.9, 25.0, 25.2, 25.8, 26.5, &
             26.3, 25.9, 28.4, 30.6, 31.0, 33.9/
      DATA (H(1,J),J=1,NCOEF)/1, -1, -25, -625, 0, 0/
      DATA (H(2,J),J=1,NCOEF)/0, 1, 54, 2916, -1, -54/
      DATA (H(3,J),J=1,NCOEF)/0, 0, 1, 50, 0, 0/
      DATA (H(4,J),J=1,NCOEF)/0, 0, 1, 108, 0, -1/
!
      DATA RLABEL/'NONE'/,CLABEL/'NONE'/
!
      DO 10  I=1, NOBS
         IF (I .LE. 25) THEN
!                                Constant function.
            X(I,1) = 1.0
         ELSE IF (I.GT.25 .AND. I.LE.54) THEN
!                                Quadratic function.
            X(I,2) = 1.0
            X(I,3) = I
            X(I,4) = I**2
         ELSE IF (I .GT. 54) THEN
!                                Linear function.
            X(I,5) = 1.0
            X(I,6) = I
         END IF
   10 CONTINUE
      IDO    = 1
      INTCEP = 0
      ICEN = 0
      CALL RGIVN (X, NCOEF, INDIND, IDEP, INDDEP, B, IDO=IDO, &
                  INTCEP=INTCEP, ICEN=ICEN, R=R, D=D, DFE=DFE, SCPE=SCPE)
      CALL RLEQU (H, B,  r, d, DFE, SCPE, irankr=irankr, Irankh=irankh)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IRANKR = ', IRANKR, '  IRANKH = ', IRANKH
      WRITE (NOUT,*) 'DFE = ', DFE, '  SCPE(1,1) = ', SCPE(1,1)
      CALL WRRRL ('%/B', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(2W10.4)')
      CALL WRRRL ('%/R', R, RLABEL, CLABEL, ITRING=1, FMT='(2W10.4)')
      END
```

Output

```
 IRANKR =   6  IRANKH =   4
 DFE =     62.0000  SCPE(1,1) =     172.559

                              B
 13.99       21.58     -0.6068     0.01214      -13.81      0.7039

                              R
    -1          1          25         625          0.        0.0
                -1         -54       -2916          1.       54.0
                           -1         -50          0.        0.0
                                      -58          0.        1.0
                                                   8.       359.4
                                                             59.4
```

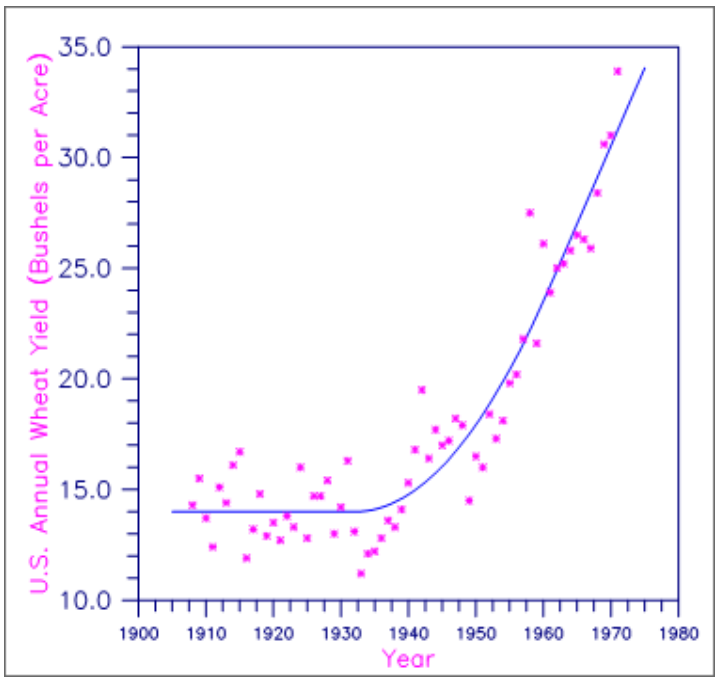


Figure 5, Annual U.S. Wheat Yield and a Grafted Polynomial Fit

Example 2

A fit to unbalanced data for a two-way classification model is computed. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \qquad i = 1, 2; j = 1, 2; k = 1, 2, \ldots, n_{ij}$$

where the α_i's and β_j's are the row and column effects, respectively, and γ_{ij}'s are the interaction effects. The responses y_{ijk} are given in the cells of the following 2 × 2 table:

17, 14, 11	13, 12
12, 14, 15, 14, 12	13, 14

The following restrictions can be imposed on the regression parameters in order to compute a cell-means fit to the responses:

1. $5\ \alpha_1 + 7\ \alpha_2 = 0$
2. $8\ \beta_1 + 4\ \beta_2 = 0$
3. $3\ \alpha_1 + 5\ \alpha_2 + 3\ \gamma_{11} + 5\ \gamma_{21} = 0$
4. $2\ \alpha_1 + 2\ \alpha_2 + 2\ \gamma_{12} + 2\ \gamma_{22} = 0$

5. $3\beta_1 + 2\beta_2 + 3\gamma_{11} + 2\gamma_{21} = 0$
6. $5\beta_1 + 2\beta_2 + 5\gamma_{12} + 2\gamma_{22} = 0$

The example program first calls IMSL routine **RGLM** with **IDO** = 1, which specifies that initialization and updating for the data are performed and wrap-up computations are not performed. This intermediate output from **RGLM** along with the restrictions is the input to **RLEQU**.

A cell-means fit to the data could also be obtained without using **RLEQU** and using **IDO** = 0 in the call to **RGLM** in this example. Although the fitted y_{ijk} would be the same, the coefficient estimates and their interpretations would be different.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    IDEP, INTCEP, LDB, LDG, LDH, LDR, LDSCPE, LDX, MAXCL, &
                     NCLVAR, NCOEF, NEF, NH, NOBS, NVAR, J
          PARAMETER  (IDEP=1, INTCEP=1, LDG=1, LDH=6, MAXCL=4, NCLVAR=2, &
                     NCOEF=9, NEF=3, NH=6, NOBS=12, NVAR=3, LDB=NCOEF, &
                     LDR=NCOEF, LDSCPE=IDEP, LDX=NOBS)
    !
          INTEGER    IDO, INDCL(NCLVAR), INDDEP(1), INDEF(4),&
                     IRANK, IRANKH, IRANKR, IRBEF(NEF+1), ICEN, &
                     NCLVAL(NCLVAR), NOUT, NRMISS, NVEF(NEF)
          REAL       B(LDB,IDEP), CLVAL(MAXCL), D(NCOEF), DFE, &
                     G(LDG,IDEP), H(LDH,NCOEF), R(LDR,NCOEF), &
                     SCPE(LDSCPE,IDEP), X(LDX,NVAR), XMAX(NCOEF), &
                     XMIN(NCOEF)
          CHARACTER  CLABEL(10)*7, RLABEL(1)*4
    !
          DATA INDCL/1, 2/, NVEF/1, 1, 2/, INDEF/1, 2, 1, 2/, INDDEP/3/
          DATA CLABEL/' ', 'MU', 'ALPHA1', 'ALPHA2', 'BETA1', 'BETA2', &
               'GAMMA11', 'GAMMA12', 'GAMMA21', 'GAMMA22'/
          DATA (X(1,J),J=1,NVAR)  /1, 1, 17/
          DATA (X(2,J),J=1,NVAR)  /1, 1, 14/
          DATA (X(3,J),J=1,NVAR)  /1, 1, 11/
          DATA (X(4,J),J=1,NVAR)  /1, 2, 13/
          DATA (X(5,J),J=1,NVAR)  /1, 2, 12/
          DATA (X(6,J),J=1,NVAR)  /2, 1, 12/
          DATA (X(7,J),J=1,NVAR)  /2, 1, 14/
          DATA (X(8,J),J=1,NVAR)  /2, 1, 15/
          DATA (X(9,J),J=1,NVAR)  /2, 1, 14/
          DATA (X(10,J),J=1,NVAR) /2, 1, 12/
          DATA (X(11,J),J=1,NVAR) /2, 2, 13/
          DATA (X(12,J),J=1,NVAR) /2, 2, 14/
          DATA (H(1,J),J=1,NCOEF) /0, 5, 7, 0, 0, 0, 0, 0, 0/
          DATA (H(2,J),J=1,NCOEF) /0, 0, 0, 8, 4, 0, 0, 0, 0/
          DATA (H(3,J),J=1,NCOEF) /0, 3, 5, 0, 0, 3, 0, 5, 0/
          DATA (H(4,J),J=1,NCOEF) /0, 2, 2, 0, 0, 0, 2, 0, 2/
          DATA (H(5,J),J=1,NCOEF) /0, 0, 0, 3, 2, 3, 2, 0, 0/
          DATA (H(6,J),J=1,NCOEF) /0, 0, 0, 5, 2, 0, 0, 5, 2/
    !
          IDO   = 1
          ICEN  = 0
          CALL RGLM (IDO=IDO, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, B, &
                     ICEN=ICEN, R=R, D=D, DFE=DFE, SCPE=SCPE)
```

```
      CALL RLEQU (H, B, r, d, DFE, SCPE, irankr=irankr,  &
                  irankh=irankh)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IRANKR = ', IRANKR, '  IRANKH = ', IRANKH
      WRITE (NOUT,*) 'DFE = ', DFE, '  SCPE(1,1) = ', SCPE(1,1)
      RLABEL(1) = 'NONE'
      CALL WRRRL ('B', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(F7.2)')
      CALL WRRRN ('R', R, ITRING=1)
      END
```

Output

```
 IRANKR =   9  IRANKH =   5
 DFE =     8.00000  SCPE(1,1) =     26.2000

                                    B
       MU   ALPHA1   ALPHA2    BETA1    BETA2  GAMMA11  GAMMA12  GAMMA21
    13.42    -0.02     0.01     0.21    -0.42     0.39    -0.48    -0.24

 GAMMA22
    0.49

                                    R
        1       2       3       4       5       6       7       8       9
1    3.46    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
2           -5.00   -7.00    0.00    0.00    0.00    0.00    0.00    0.00
3                   -0.80    0.00    0.00   -3.00    0.00   -5.00    0.00
4                           -8.00   -4.00    0.00    0.00    0.00    0.00
5                                   -0.50   -3.00   -2.00    0.00    0.00
6                                           -3.00   -2.00   -5.00   -2.00
7                                                   10.41    3.20   11.37
8                                                            24.56    9.65
9                                                                     2.45
```

RSTAT

Computes statistics related to a regression fit given the coefficient estimates $\hat{\beta}$ and the *R* matrix.

Required Arguments

IRBEF — Index vector of length |**IEF**| + 1. (Input, if **IEF** is positive.)
For *i* = 1, 2, ..., |**IEF**|, element numbers **IRBEF**(*i*), **IRBEF**(*i*) + 1, ..., **IRBEF**(*i* + 1) **-** 1, of **B** correspond to the *i*-th effect.

B — Vector of length **NCOEF** containing a least-squares solution $\hat{\beta}$ for the regression coefficients. (Input)
Here, if **IEF** > 0, then **NCOEF** = **IRBEF**(**IEF** + 1) **-** 1; and if **IEF** ≤ 0, then **NCOEF** = **INTCEP - IEF**. If **INTCEP** = 1, then **B**(1) must be the estimated intercept.

R — **NCOEF** by **NCOEF** upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements. Only the upper triangle of *R* is referenced.

DFE — Degrees of freedom for error. (Input)

SSE — Sum of squares for error. (Input)

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	**AOV(I)**
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic

10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

If INTCEP = 1, the regression and total are corrected for the mean. If INTCEP = 0, the regression and total are not corrected for the mean, and AOV(14) and AOV(15) are set to NaN (not a number).

SQSS — |IEF| by 4 matrix containing in columns 1 through 4 the sequential degrees of freedom, sum of squares, *F*-statistic, and *p*-value. (Output)
Each row corresponds to an effect. If IEF = 0, SQSS is not referenced and can be a vector of length one.

COEF — NCOEF by 5 matrix containing statistics relating to the regression coefficients. (Output)
Each row corresponds to a coefficient in the model. Row INTCEP + I corresponds to the coefficient for the I-th independent variable. If INTCEP = 1, the first row corresponds to the intercept. The statistics in the columns are

Col.	Description
1	Coefficient estimate.
2	Estimated standard error of the coefficient estimate.
3	*t*-statistic for the test that the coefficient is zero.
4	*p*-value for the two-sided *t* test.
5	Variance inflation factors. The square of the multiple correlation coefficient for the I-th regressor after all others can be obtained from COEF(I, 5) by the formula 1.0 - 1.0/COEF(I, 5). If INTCEP = 0 or INTCEP = 1 and I = 1, the "multiple correlation coefficient" is not adjusted for the mean.

COVB — NCOEF by NCOEF matrix that is the estimated variance-covariance matrix of the estimated regression coefficients when *R* is nonsingular and is from an unrestricted regression fit. (Output)
See Comments for an explanation of COVB when *R* is singular or *R* is from a restricted regression fit. If *R* is not needed, COVB and *R* can share the same storage locations.

Optional Arguments

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IEF — Effect option. (Input)
Default: IEF = 0.
The absolute value of IEF is the number of effects (sources of variation) in the model excluding the error. The sign of IEF specifies the following options:

IEF	Meaning
< 0	Each effect corresponds to a single regressor (coefficient) in the model.
> 0	Each effect corresponds to one or more regressors. The association between the effects and the regressors is given by elements of IRBEF.
0	There are no effects in the model. INTCEP must equal 1.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

PRINT — Printing option. (Input)
Default: PRINT = 'N'.
PRINT is a character string indicating what is to be printed. The PRINT string is composed of one character print codes to control printing. These print codes are given as follows:

PRINT(I : I)	Printing that occurs
'A'	All
'N'	None
'1'	AOV
'2'	SQSS
'3'	COEF
'4'	COVB

The concatenated print codes 'A', 'N', '1', …, '4' that comprise the PRINT string give the combination of statistics to be printed. Here are a few examples.

PRINT	Printing that occurs
'A'	All
'N'	None
'13'	AOV and COEF
'124'	AOV, SQSS, and COVB

LDSQSS — Leading dimension of SQSS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSQSS = size (SQSS,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOVB — Leading dimension of COVB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVB = size (COVB,1).

FORTRAN 90 Interface

Generic: CALL RSTAT (IRBEF, B, R, DFE, SSE, AOV, SQSS, COEF, COVB [, ...])
Specific: The specific interface names are S_RSTAT and D_RSTAT.

FORTRAN 77 Interface

Single: CALL RSTAT (INTCEP, IEF, IRBEF, B, R, LDR, DFE, SSE, PRINT, AOV, SQSS, LDSQSS, COEF, LDCOEF, COVB, LDCOVB)
Double: The double precision name is DRSTAT.

Description

Routine RSTAT computes summary statistics from a fitted general linear model. The model is $y = X\beta + \varepsilon$ where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors, β is the $p \times 1$ vector of regression coefficients, and ε is the $n \times 1$ vector of errors whose elements are each independently distributed with mean 0 and variance σ^2. Routine RGIVN or routine RGLM can be used to compute the fit of the model. Next, RSTAT uses the results of this fit to compute summary statistics, including analysis of variance, sequential sum of squares, t tests, and estimated variance-covariance matrix of the estimated regression coefficients.

Some generalizations of the general linear model are allowed. If the i-th element of ε has variance σ^2/w_i and the weights w_i are used in the fit of the model, RSTAT produces summary statistics from the weighted least-squares fit. More generally, if the variance-covariance matrix of ε is $\sigma^2 V$, RSTAT can be used to produce summary statistics from the generalized least-squares fit. (Routine RGIVN can be used to perform a generalized least-squares fit, by regressing y^* on X^* where $y^* = (T^{-1})^T y$, $X^* = (T^{-1})^T X$ and T satisfies $T^T T = V$. Routines for computing y^* and X^* can be found in the IMSL MATH/LIBRARY.)

If the general linear model has the restriction $H\beta = g$ on the regression parameters, and this restriction is used in the fit of the model by routine **RLEQU**, **RSTAT** produces summary statistics from this restricted least-squares fit.

The sequential sum of squares for the i-th regression parameter is given by

$$\left(R\hat{\beta}\right)_i^2$$

The regression sum of squares is given by the sum of the sequential sums of squares. If an intercept is in the model, the regression sum of squares is adjusted for the mean, i.e.,

$$\left(R\hat{\beta}\right)_1^2$$

is not included in the sum.

The estimate of σ^2 is s^2 (stored in **A0V**(8)) that is computed as **SSE/DFE**.

If R is nonsingular, the estimated variance-covariance matrix of $\hat{\beta}$ (stored in **C0VB**) is computed by $s^2R^{-1}(R^{-1})^T$.

If R is singular, corresponding to rank $(X) < p$, a generalized inverse is used. For a matrix G to be a g_i(i = 1, 2, 3, or 4) inverse of a matrix A, G must satisfy conditions j (for $j \leq i$) for the Moore-Penrose inverse but generally must fail conditions k (for $k > i$). The four conditions for G to be a Moore-Penrose inverse of A are as follows:

1. $AGA = A$
2. $GAG = G$
3. AG is symmetric
4. GA is symmetric

In the case where R is singular, the method for obtaining **C0VB** follows the discussion of Maindonald (1984, pages 101 - 103). Let Z be the diagonal matrix with diagonal elements defined by

$$z_{ii} = \begin{cases} 1 & \text{if } r_{ii} \neq 0 \\ 0 & \text{if } r_{ii} = 0 \end{cases}$$

Let G be the solution to $RG = Z$ obtained by setting the i-th ($\{i : r_{ii} = 0\}$) row of G to zero. **C0VB** is set to s^2GG^T. (G is a g_3 inverse of R. For any g_3 inverse of R, represented by

$$R^{g_3}$$

the result

$$R^{g_3} R^{g_3 T}$$

is a symmetric g_2 inverse of $R^T R = X^T X$. See Sallas and Lionti [1988].)

Note that **COVB** can only be used to get variances and covariances of estimable functions of the regression coefficients, i.e., nonestimable functions (linear combinations of the regression coefficients not in the space spanned by the nonzero rows of R) must not be used. See, for example, Maindonald (1984, pages 166 - 168) for a discussion of estimable functions.

The estimated standard errors of the estimated regression coefficients (stored in column 2 of **COEF**) are computed as square roots of the corresponding diagonal entries in **COVB**.

For the case where an intercept is in the model, put

$$\overline{R}$$

equal to the matrix R with the first row and column deleted. Generally, the variance inflation factor (VIF) for the i-th regression coefficient is computed as the product of the i-th diagonal element of $R^T R$ and the i-th diagonal element of its computed inverse. If an intercept is in the model, the VIF for those coefficients not corresponding to the intercept uses the diagonal elements of

$$\overline{R}^T \overline{R}$$

(see Maindonald 1984, page 40).

The preceding discussion can be modified to include the restricted least-squares problem. The modification is based on the work of Stirling (1981). Let the matrix $D = \text{diag}(d_1, d_2, \ldots, d_p)$ be a diagonal matrix with elements $d_i = 0$ if the i-th row of R corresponds to restriction. In the unrestricted case, D is simply the $p \times p$ identity matrix. The formula for **COVB** is $s^2 GDG^T$. The formula for the sequential sum of squares for the i-th ($\{i : r_{ii} > 0\}$) regression parameter is given by

$$\left(DR\hat{\beta} \right)_i^2$$

Sequential sums of squares for $\{i : r_{ii} \leq 0\}$ are set to zero.

For the restricted least-squares problem, the sequential and regression sums of squares correspond to those from a fitted reduced model obtained by first substituting the restriction $H\,\beta = g$ into the model. In general, the reduced model is not unique. Care must be taken to interpret the sequential sums of squares in the context of the particular reduced model indicated by the R matrix. If $g = 0$, any of the reduced models that could be computed from the restrictions will produce the same regression sum of squares. However, if $g \neq 0$, different reduced models resulting from the same restricted model can have different regressands, and hence, different total and regression sums of squares.

Comments

When R is nonsingular and comes from an unrestricted regression fit, **COVB** is the estimated variance-covariance matrix of the estimated regression coefficients, and **COVB** = (**SSE**/**DFE**) * $(R^TR)^{-1}$. Otherwise, variances and covariances of estimable functions of the regression coefficients can be obtained using **COVB**, and **COVB** = (**SSE**/**DFE**) * GDG^T. Here, D is the diagonal matrix with diagonal elements equal to 0 if the corresponding rows of R are restrictions and with diagonal elements equal to one otherwise. Also, G is a particular generalized inverse of R. See the Description section.

Examples

Example 1

This example uses a data set discussed by Draper and Smith (1981, pages 629-630). This data set is put into the matrix **X** by routine **GDATA** (see Chapter 19, "Utilities"). There are four independent variables and one dependent variable. Routine **RGIVN** is invoked to fit the regression model and **RSTAT** is invoked to compute summary statistics.

```
      USE RSTAT_INT
      USE GDATA_INT
      USE RGIVN_INT

      IMPLICIT   NONE
!                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    INTCEP, LDB, LDCOEF, LDCOVB, LDR, LDSCPE, LDSQSS, &
                 LDX, NCOEF, NDEP, NDX, NIND
      PARAMETER  (INTCEP=1, LDX=13, NDEP=1, NDX=5, NIND=4, &
                 LDSCPE=NDEP, LDSQSS=NIND, NCOEF=INTCEP+NIND, &
                 LDB=NCOEF, LDCOEF=NCOEF, LDCOVB=NCOEF, LDR=NCOEF)
!
      INTEGER    IDEP, IDO, IEF, IFRQ, IIND, INDDEP(1), INDIND(1), &
                 IRANK, IRBEF(1), IWT, NCOL, NRMISS, NROW
      REAL       AOV(15), B(LDB,NDEP), COEF(LDCOEF,5), &
                 COVB(LDCOVB,5), D(NCOEF), DFE, R(LDR,NCOEF), &
                 SCPE(LDSCPE,NDEP), SQSS(LDSQSS,4), SSE, TOL, &
                 X(LDX,NDX), XMAX(NCOEF), XMIN(NCOEF)
      CHARACTER  PRINT*5
!
      CALL GDATA (5, X, NROW, NCOL)
      IIND = -NIND
      IDEP = -NDEP

      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, &
                  SCPE=SCPE)
      PRINT = 'A'
      IEF   = -NIND
      SSE   = SCPE(1,1)
!
      CALL RSTAT (IRBEF, B(:, 1), R, DFE, SSE, AOV, SQSS, COEF, COVB,  &
                  IEF=IEF, PRINT=PRINT)
```

```
!
      END
```

Output

```
R-squared   Adjusted  Est. Std. Dev.              Coefficient of
(percent)  R-squared  of Model Error        Mean  Var. (percent)
   98.238     97.356           2.446       95.42           2.563

               * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
Source               DF     Squares      Square  Overall F  Larger F
Regression            4      2667.9       667.0    111.479    0.0000
Residual              8        47.9         6.0
Corrected Total      12      2715.8
         * * * Sequential Statistics * * *
Indep.    Degrees of     Sum of                Prob. of
Variable     Freedom     Squares  F-statistic   Larger F
       1           1      1450.1      242.368     0.0000
       2           1      1207.8      201.870     0.0000
       3           1         9.8        1.637     0.2366
       4           1         0.2        0.041     0.8441

                * * * Inference on Coefficients * * *
                     Standard                Prob. of    Variance
Coef.    Estimate       Error  t-statistic  Larger |t|   Inflation
    1       62.41       70.07        0.891      0.3991     10668.5
    2        1.55        0.74        2.083      0.0708        38.5
    3        0.51        0.72        0.705      0.5009       254.4
    4        0.10        0.75        0.135      0.8959        46.9
    5       -0.14        0.71       -0.203      0.8441       282.5

  * * * Variance-Covariance Matrix for the Coefficient Estimates * * *
              1            2            3            4            5
1       4909.95       -50.51       -50.60       -51.66       -49.60
2                       0.55         0.51         0.55         0.51
3                                    0.52         0.53         0.51
4                                                 0.57         0.52
5                                                              0.50
```

Example 2

A one-way analysis of covariance model is fitted to the turkey data discussed by Draper and Smith (1981, pages 243-249). The response variable is turkey weight *y* (in pounds). Three groups of turkeys corresponding to the three states where they were reared are used. The age of a turkey (in weeks) is the covariate. The explanatory variables are age, group, and interaction. The model is

$$y_{ij} = \mu + \beta x_{ij} + \alpha_i + \beta_i x_{ij} + \varepsilon_{ij} \qquad i = 1, 2, 3; j = 1, 2, \ldots, n_i$$

where $\alpha_3 = 0$ and $\beta_3 = 0$. Routine **RGLM** is used to fit the model with the option **IDUMMY** = 2. Then, **RSTAT** is used to compute summary statistics. The fitted model gives three separate lines with slopes 0.506, 0.470, and 0.445. The *F* test for interaction (the last effect) suggests omitting the interaction from the model and using a model with identical slopes for each group.

```
      USE RSTAT_INT
      USE RGLM_INT

      IMPLICIT   NONE
!                                  SPECIFICATIONS FOR PARAMETERS
      INTEGER    IDEP, IEF, INTCEP, LDB, LDCOEF, LDCOVB, LDR, LDSCPE, &
                 LDSQSS, LDX, MAXB, MAXCL, NCLVAR, NCOL, NROW, J
      PARAMETER  (IDEP=1, IEF=3, INTCEP=1, LDX=13, MAXB=6, MAXCL=3, &
                 NCLVAR=1, NCOL=3, NROW=13, LDB=MAXB, LDCOEF=MAXB, &
                 LDCOVB=MAXB, LDR=MAXB, LDSCPE=IDEP, LDSQSS=IEF)
!
      INTEGER    IDO, IDUMMY, IFRQ, INDCL(NCLVAR), INDDEP(IDEP), &
                 INDEF(4), IRANK, IRBEF(IEF+1), IWT, NCLVAL(NCLVAR), &
                 NRMISS, NVEF(IEF)
      REAL       AOV(15), B(LDB,IDEP), CLVAL(MAXCL), &
                 COEF(LDCOEF,5), COVB(LDCOVB,MAXB), D(MAXB), DFE, &
                 R(LDR,MAXB), SCPE(LDSCPE,IDEP), SQSS(LDSQSS,4), SSE, &
                 TOL, X(LDX,NCOL), XMAX(MAXB), XMIN(MAXB)
      CHARACTER  PRINT*1
!
      DATA (X(1,J),J=1,3)/25, 13.8, 3/
      DATA (X(2,J),J=1,3)/28, 13.3, 1/
      DATA (X(3,J),J=1,3)/20, 8.9, 1/
      DATA (X(4,J),J=1,3)/32, 15.1, 1/
      DATA (X(5,J),J=1,3)/22, 10.4, 1/
      DATA (X(6,J),J=1,3)/29, 13.1, 2/
      DATA (X(7,J),J=1,3)/27, 12.4, 2/
      DATA (X(8,J),J=1,3)/28, 13.2, 2/
      DATA (X(9,J),J=1,3)/26, 11.8, 2/
      DATA (X(10,J),J=1,3)/21, 11.5, 3/
      DATA (X(11,J),J=1,3)/27, 14.2, 3/
      DATA (X(12,J),J=1,3)/29, 15.4, 3/
      DATA (X(13,J),J=1,3)/23, 13.1, 3/
      DATA INDCL/3/, NVEF/1, 1, 2/, INDEF/1, 3, 1, 3/, INDDEP/2/
!
      IDUMMY = 2
      CALL RGLM (X, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, &
                 B, IDUMMY=IDUMMY, IRBEF=IRBEF, R=R, DFE=DFE, SCPE=SCPE)
!
      SSE   = SCPE(1,1)
      PRINT = 'A'
      CALL RSTAT (IRBEF, B(:,1), R, DFE, SSE, AOV, SQSS, COEF, COVB, &
                 ief=ief,PRINT=PRINT)
!
      END
```

Output

```
R-squared   Adjusted  Est. Std. Dev.              Coefficient of
(percent)  R-squared  of Model Error        Mean  Var. (percent)
   98.208     96.929          0.3176       12.78           2.484

                   * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
Source               DF     Squares      Square  Overall F  Larger F
Regression            5       38.71       7.742     76.744    0.0000
Residual              7        0.71       0.101
Corrected Total      12       39.42
```

```
              * * * Sequential Statistics * * *
            Degrees of     Sum of                Prob. of
Effect       Freedom     Squares  F-statistic   Larger F
     1             1       26.20      259.728     0.0000
     2             2       12.40       61.477     0.0000
     3             2        0.11        0.520     0.6156

                  * * * Inference on Coefficients * * *
                        Standard                 Prob. of    Variance
Coef.     Estimate       Error  t-statistic  Larger |t|   Inflation
    1        2.475       1.264        1.959      0.0910       205.7
    2        0.445       0.050        8.861      0.0000         3.8
    3       -3.454       1.531       -2.257      0.0586        64.3
    4       -2.775       4.109       -0.675      0.5211       463.4
    5        0.061       0.060        1.013      0.3447        68.1
    6        0.025       0.151        0.166      0.8729       472.3

       * * * Variance-Covariance Matrix for the Coefficient Estimates * * *
                1             2             3             4             5
1          1.5965       -0.0631       -1.5965       -1.5965        0.0631
2                        0.0025        0.0631        0.0631       -0.0025
3                                      2.3425        1.5965       -0.0913
4                                                   16.8801       -0.0631
5                                                                  0.0036

                6
1          0.0631
2         -0.0025
3         -0.0631
4         -0.6179
5          0.0025
6          0.0227
```

Example 3

A two-way analysis-of-variance model is fitted to balanced data discussed by Snedecor and Cochran (1967, Table 12.5.1, page 347). The responses are the weight gains (in grams) of rats fed diets varying in two components—level of protein and source of protein. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \qquad i = 1, 2; j = 1, 2, 3; k = 1, 2, \ldots, 10$$

where

$$\sum_{i=1}^{2} \alpha_i = 0; \sum_{j=1}^{3} \beta_j = 0; \sum_{i=1}^{2} \gamma_{ij} = 0 \text{ for } j = 1, 2, 3; \text{ and } \sum_{j=1}^{3} \gamma_{ij} = 0 \text{ for } i = 1, 2$$

Routine **RGLM** is used to fit the model with the **IDUMMY** = 0 option. Then, **RSTAT** is used to compute summary statistics.

```
        USE RSTAT_INT
        USE RGLM_INT

        IMPLICIT   NONE
```

```
      INTEGER    IDEP, IEF, LDB, LDCOEF, LDCOVB, LDR, LDSCPE, LDSQSS, &
                 LDX, LINDEF, MAXB, MAXCL, NCLVAR, NCOL, NEF, NROW
      PARAMETER  (IDEP=1, LINDEF=4, MAXB=12, MAXCL=5, NCLVAR=2, &
                 NCOL=3, NEF=3, NROW=60, IEF=NEF, LDB=MAXB, &
                 LDCOEF=MAXB, LDCOVB=MAXB, LDR=MAXB, LDSCPE=IDEP, &
                 LDSQSS=NEF, LDX=NROW)
!
      INTEGER    IDO, IDUMMY, IFRQ, INDCL(NCLVAR), INDDEP(IDEP),&
                 INDEF(LINDEF), INTCEP, IRANK, IRBEF(NEF+1), IWT, &
                 NCLVAL(NCLVAR), NRMISS, NVEF(NEF)
      REAL       AOV(15), B(LDB,IDEP), CLVAL(MAXCL), &
                 COEF(LDCOEF,5), COVB(LDCOVB,MAXB), D(MAXB), DFE, &
                 R(LDR,MAXB), SCPE(LDSCPE,IDEP), SQSS(LDSQSS,4), SSE, &
                 TOL, X(LDX,NCOL), XMAX(MAXB), XMIN(MAXB)
      CHARACTER  PRINT*1
!
      DATA X/73.0, 102.0, 118.0, 104.0, 81.0, 107.0, 100.0, 87.0, &
           117.0, 111.0, 98.0, 74.0, 56.0, 111.0, 95.0, 88.0, 82.0, &
           77.0, 86.0, 92.0, 94.0, 79.0, 96.0, 98.0, 102.0, 102.0, &
           108.0, 91.0, 120.0, 105.0, 90.0, 76.0, 90.0, 64.0, 86.0, &
           51.0, 72.0, 90.0, 95.0, 78.0, 107.0, 95.0, 97.0, 80.0, &
           98.0, 74.0, 74.0, 67.0, 89.0, 58.0, 49.0, 82.0, 73.0, 86.0, &
           81.0, 97.0, 106.0, 70.0, 61.0, 82.0, 30*1.0, 30*2.0, &
           10*1.0, 10*2.0, 10*3.0, 10*1.0, 10*2.0, 10*3.0/
      DATA INDCL/2, 3/, NVEF/1, 1, 2/, INDEF/2, 3, 2, 3/, INDDEP/1/
!
      IDUMMY = 0
      CALL RGLM (X, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, B, &
                IDUMMY=IDUMMY, IRBEF=IRBEF, R=R, DFE=DFE, SCPE=SCPE)
!
      SSE   = SCPE(1,1)
      PRINT = 'A'
      CALL RSTAT (IRBEF, B(:,1), R, DFE, SSE, AOV, SQSS, COEF, &
                 COVB, IEF=IEF, PRINT=PRINT)
!
      END
```

Output

```
 R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 (percent)  R-squared  of Model Error        Mean  Var. (percent)
    28.477     21.854           14.65       87.87           16.67

                     * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
 Source                   DF    Squares      Square  Overall F  Larger F
 Regression                5     4612.9       922.6      4.300    0.0023
 Residual                 54    11586.0       214.6
 Reduced Model Total      59    16198.9

         * * * Sequential Statistics * * *
          Degrees of     Sum of               Prob. of
 Effect     Freedom     Squares  F-statistic   Larger F
      1           1      3168.3       14.767     0.0003
      2           2       266.5        0.621     0.5411
      3           2      1178.1        2.746     0.0732

                * * * Inference on Coefficients * * *
                      Standard                 Prob. of    Variance
```

Coef.	Estimate	Error	t-statistic	Larger |t|	Inflation
1	87.87	1.891	46.47	0.0000	1.000
2	7.27	1.891	3.84	0.0003	NaN
3	-7.27	1.891	-3.84	0.0003	1.000
4	1.73	2.674	0.65	0.5196	NaN
5	-2.97	2.674	-1.11	0.2722	1.333
6	1.23	2.674	0.46	0.6465	1.333
7	3.13	2.674	1.17	0.2465	NaN
8	-6.27	2.674	-2.34	0.0228	NaN
9	3.13	2.674	1.17	0.2465	NaN
10	-3.13	2.674	-1.17	0.2465	NaN
11	6.27	2.674	2.34	0.0228	1.333
12	-3.13	2.674	-1.17	0.2465	1.333

* * * Variance-Covariance Matrix for the Coefficient Estimates * * *

	1	2	3	4	5
1	3.57593	0.00000	0.00000	0.00000	0.00000
2		3.57593	-3.57593	0.00000	0.00000
3			3.57593	0.00000	0.00000
4				7.15185	-3.57592
5					7.15185

	6	7	8	9	10
1	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000	0.00000	0.00000
4	-3.57593	0.00000	0.00000	0.00000	0.00000
5	-3.57593	0.00000	0.00000	0.00000	0.00000
6	7.15185	0.00000	0.00000	0.00000	0.00000
7		7.15185	-3.57592	-3.57593	-7.15185
8			7.15185	-3.57593	3.57592
9				7.15185	3.57593
10					7.15185

	11	12
1	0.00000	0.00000
2	0.00000	0.00000
3	0.00000	0.00000
4	0.00000	0.00000
5	0.00000	0.00000
6	0.00000	0.00000
7	3.57592	3.57593
8	-7.15185	3.57593
9	3.57593	-7.15185
10	-3.57592	-3.57593
11	7.15185	-3.57593
12		7.15185

RCOVB

Computes the estimated variance-covariance matrix of the estimated regression coefficients given the *R* matrix.

Required Arguments

R — NCOEF by NCOEF upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements. Only the upper triangle of *R* is referenced.

S2 — s^2, the estimated variance of the error in the regression model. (Input)
s^2 is the error mean square from the regression fit.

COVB — NCOEF by NCOEF matrix that is the estimated variance-covariance matrix of the estimated regression coefficients when *R* is nonsingular and is from an unrestricted regression fit. (Output)
See `Comments` for an explanation of COVB when *R* is singular or *R* is from a restricted regression fit. If *R* is not needed, COVB and R can share the same storage locations.

Optional Arguments

NCOEF — Number of regression coefficients in the model. (Input)
Default: NCOEF = size (R,1).

LDR — Leading dimension of *R* exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

LDCOVB — Leading dimension of COVB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVB = size (COVB,1).

FORTRAN 90 Interface

Generic: CALL RCOVB (R, S2, COVB [, ...])

Specific: The specific interface names are **S_RCOVB** and **D_RCOVB**.

FORTRAN 77 Interface

Single: **CALL RCOVB** (**NCOEF**, **R**, **LDR**, **S2**, **COVB**, **LDCOVB**)

Double: The double precision name is **DRCOVB**.

Description

Routine **RCOVB** computes an estimated variance-covariance matrix of estimated regression parameters from the R matrix in several models. In the simplest situation, the model is a general linear model given by $y = X\beta + \varepsilon$ where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors, β is the $p \times 1$ vector of regression coefficients, and ε is the $n \times 1$ vector of errors whose elements are each independently distributed with mean 0 and variance σ^2. Routine **RGIVN** can be used to get the fit of the model and the R matrix.

If the i-th element of ε has variance σ^2/w_i and the weights w_i are used in the fit of the model, **RCOVB** produces the estimated variance-covariance matrix from the R matrix in the weighted least squares fit. More generally, if the variance-covariance matrix of ε is $\sigma^2 V$, **RCOVB** can be used to produce the estimated variance-covariance matrix from the generalized least-squares fit. (Routine **RGIVN** can be used to perform a generalized least-squares fit, by regressing y^* on X^* where $y^* = (T^{-1})^T y$, $X^* = (T^{-1})^T X$ and T satisfies $T^T T = V$.)

If the general linear model has the restriction $H\beta = g$ on the regression parameters and this restriction is used in the fit of the model by routine **RLEQU**, **RCOVB** produces the estimated variance-covariance from the R matrix in the restricted least squares fit.

Routine **RCOVB** computes an estimated variance-covariance matrix for the estimated regression coefficients,

$$\hat{B}$$

in a fitted multivariate general linear model. The model is $Y = XB + E$ where Y is the $n \times q$ matrix of responses, X is the $n \times p$ matrix of regressors, B is the $p \times q$ matrix of regression coefficients, and E is the $n \times q$ matrix of errors whose rows are each independently distributed as a q-dimensional multivariate normal each with mean vector 0 and variance-covariance matrix Σ. Let

$$\hat{B} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_q \right)$$

The estimated covariance matrix

$$Cov\left(\hat{\beta}_i,\hat{\beta}_j\right) = s_{ij}\left(X^T X\right)^{-1}$$

Here, s_{ij} (input in **S2**) is the estimate of the *ij*-th element of **Σ**.

If a nonlinear regression model is fit using routine **RNLIN**, **RCOVB** produces the asymptotic estimated variance-covariance matrix from the *R* matrix in that fit.

If *R* is singular, corresponding to rank(*R*) < *p*, a generalized inverse is used to compute **COVB**. For a matrix *G* to be a g_i(*i* = 1, 2, 3, or 4) inverse of a matrix *A*, *G* must satisfy conditions *j*(for $j \le i$) for the Moore-Penrose inverse but, generally, must fail conditions *k* (for *k* > *i*). The four conditions for *G* to be a Moore-Penrose inverse of *A* are as follows:

1. *AGA* = *A*
2. *GAG* = *G*
3. *AG* is symmetric
4. *GA* is symmetric

In the case that *R* is singular, the method for obtaining **COVB** follows the discussion of Maindonald (1984, pages 101-103). Let *Z* be the diagonal matrix with diagonal elements defined by

$$z_{ii} = \begin{cases} 1 & \text{if } r_{ii} \neq 0 \\ 0 & \text{if } r_{ii} = 0 \end{cases}$$

Let *G* be the solution to *RG* = *Z* obtained by setting the *i*-th ({*i* : r_{ii} = 0}) row of *G* to zero. **COVB** is set to s^2GG^T. (*G* is a g_3 inverse of *R*. For any g_3 inverse of *R*, represented by

$$R^{g_3}$$

the result

$$R^{g_3}R^{g_3 T}$$

is a symmetric g_2 inverse of $R^TR = X^TX$. See Sallas and Lionti [1988].)

Note that **COVB** can only be used to get variances and covariances of estimable functions of the regression coefficients, i.e., nonestimable functions (linear combinations of the regression coefficients not in the space spanned by the nonzero rows of *R*) must not be used. See, for example, Maindonald (1984, pages 166-168) for a discussion of estimable functions.

The preceding discussion can be modified to include the restricted least-squares problem. The modification is based on the work of Stirling (1981). Let the matrix $D = \text{diag}(d_1, d_2, \ldots, d_p)$ be a diagonal matrix with elements $d_{ii} = 0$ if the i-th row of R corresponds to a restriction and 1 otherwise. In the unrestricted case, D is simply the $p \times p$ identity matrix. The formula for **COVB** is s^2GDG^T.

Comments

When R is nonsingular and comes from an unrestricted regression fit, **COVB** is the estimated variance-covariance matrix of the estimated regression coefficients, and **COVB** = $s^2(R^TR)^{-1}$. Otherwise, variances and covariances of estimable functions of the regression coefficients can be obtained using **COVB**, and **COVB** = s^2GDG^T. Here, D is the diagonal matrix with diagonal elements equal to 0 if the corresponding rows of R are restrictions and with diagonal elements equal to one otherwise. Also, G is a particular generalized inverse of R. See the Description section.

Examples

Example 1

This example uses a data set discussed by Draper and Smith (1981, pages 629-630). This data set is put into the matrix X by routine **GDATA** (see Chapter 19, "Utilities"). There are 4 independent variables and 1 dependent variable. Routine **RGIVN** is invoked to fit the regression model, and **RCOVB** is invoked to compute summary statistics.

```
          USE RCOVB_INT
          USE GDATA_INT
          USE RGIVN_INT
          USE WRRRL_INT

          IMPLICIT   NONE
          INTEGER    INTCEP, LDB, LDCOEF, LDCOVB, LDR, LDSCPE, LDX, NCOEF, &
                     NDEP, NDX, NIND
          PARAMETER  (INTCEP=1, LDX=13, NDEP=1, NDX=5, NIND=4, &
                     LDSCPE=NDEP, NCOEF=INTCEP+NIND, LDB=NCOEF, &
                     LDCOEF=NCOEF, LDCOVB=NCOEF, LDR=NCOEF)
    !
          INTEGER    IDEP, IDO, IFRQ, IIND, INDDEP(1), INDIND(1), IRANK, &
                     ICEN, IWT, NCOL, NRMISS, NROW
          REAL       B(LDB,NDEP), COVB(LDCOVB,5), DFE, R(LDR,NCOEF), &
                     S2, SCPE(LDSCPE,NDEP), X(LDX,NDX)
          CHARACTER  CLABEL(6)*10, RLABEL(5)*10
    !
          DATA RLABEL/'Intercept', 'X1', 'X2', 'X3', 'X4'/
          DATA CLABEL/' ', 'Intercept', 'X1', 'X2', 'X3', 'X4'/
    !
          CALL GDATA (5, X, NROW, NCOL)
          IIND = -NIND
          IDEP = -NDEP
```

```
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, &
                  SCPE=SCPE)
      S2 = SCPE(1,1)/DFE
!
      CALL RCOVB (R, S2, COVB)
      CALL WRRRL ('COVB', COVB, RLABEL, CLABEL, FMT='(2W10.4)')
!
      END
```

Output

```
                                  COVB
             Intercept          X1          X2          X3          X4
Intercept       4910.0      -50.51      -50.60      -51.66      -49.60
X1               -50.5        0.55        0.51        0.55        0.51
X2               -50.6        0.51        0.52        0.53        0.51
X3               -51.7        0.55        0.53        0.57        0.52
X4               -49.6        0.51        0.51        0.52        0.50
```

Example 2

In this example, routine **RNLIN** is first invoked to fit the following nonlinear regression model discussed by Neter, Wasserman, and Kutner (1983, pages 475-478):

$$y_i = \theta_1 e^{\theta_2 x_i} + \varepsilon_i i = 1, 2, \ \ldots, 15$$

Then, **RCOVB** is used to compute the estimated asymptotic variance-covariance matrix of the estimated nonlinear regression parameters. Finally, the diagonal elements of the output matrix from **RCOVB** are used together with routine **TIN** (see Chapter 17, "Probability Distribution Functions and Inverses") to compute 95% confidence intervals on the regression parameters.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDR, NOBS, NPARM
      PARAMETER  (NOBS=15, NPARM=2, LDR=NPARM)
!
      INTEGER    I, IDERIV, IRANK, ISETNG, NOUT
      REAL       A, DFE, R(LDR,NPARM), SQRT, SSE, THETA(NPARM)
      INTRINSIC  SQRT
      EXTERNAL   EXAMPL
!
      DATA THETA/60.0, -0.03/
!
      CALL UMACH (2, NOUT)
!
      IDERIV = 1
      CALL RNLIN (EXAMPL, THETA, IDERIV=IDERIV, R=R, DFE=DFE, SSE=SSE)
!
      CALL RCOVB (R, SSE/DFE, R)
!                                 Print
      ISETNG=2
      CALL WROPT (-6, ISETNG, 0)
```

```
      CALL WRRRN ('Estimated Asymptotic Variance-Covariance Matrix', &
                  R)
!                                Compute and print 95 percent
!                                confidence intervals.
      WRITE (NOUT,*)
      WRITE (NOUT,*) '        95% Confidence Intervals     '
      WRITE (NOUT,*) '   Estimate  Lower Limit  Upper Limit'
      DO 10  I=1, NPARM
         A = TIN(0.975,DFE)*SQRT(R(I,I))
         WRITE (NOUT,'(1X, F10.3, 2F13.3)') THETA(I), THETA(I) - A, &
                                    THETA(I) + A
   10 CONTINUE
      END
!
      SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, &
                         IEND)
      INTEGER    NPARM, IOPT, IOBS, IEND
      REAL       THETA(NPARM), FRQ, WT, E, DE(NPARM)
!
      INTEGER    NOBS
      PARAMETER  (NOBS=15)
!
      REAL       EXP, XDATA(NOBS), YDATA(NOBS)
      INTRINSIC  EXP
!
      DATA YDATA/54.0, 50.0, 45.0, 37.0, 35.0, 25.0, 20.0, 16.0, 18.0, &
           13.0, 8.0, 11.0, 8.0, 4.0, 6.0/
      DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
           38.0, 45.0, 52.0, 53.0, 60.0, 65.0/
!
      IF (IOBS .LE. NOBS) THEN
         WT   = 1.0E0
         FRQ  = 1.0E0
         IEND = 0
         IF (IOPT .EQ. 0) THEN
            E = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
         ELSE
            DE(1) = -EXP(THETA(2)*XDATA(IOBS))
            DE(2) = -THETA(1)*XDATA(IOBS)*EXP(THETA(2)*XDATA(IOBS))
         END IF
      ELSE
         IEND = 1
      END IF
      RETURN
      END
```

Output

```
 Estimated Asymptotic Variance-Covariance Matrix
                1             2
1   2.16701E+00  -1.78121E-03
2  -1.78121E-03   2.92786E-06

     95% Confidence Intervals
Estimate  Lower Limit  Upper Limit
58.603       55.423       61.784
-0.040       -0.043       -0.036
```

CESTI

more...

Constructs an equivalent completely testable multivariate general linear hypothesis $H\,BU = G$ from a partially testable hypothesis $H_pBU = G_p$.

Required Arguments

HP — NHP by NCOEF matrix H_p with each row corresponding to a row in the hypothesis and containing the constants that specify a linear combination of the regression coefficients. (Input)

NDEP — Number of dependent (response) variables. (Input)

NU — U matrix option. (Input)
For positive NU, NU is the number of linear combinations of the dependent variables to be considered. If NU = 0, the hypothesis is $H_pB = G_p$, and U is automatically taken to be the identity. NU must be less than or equal to NDEP .

GP — Matrix G_p containing the null hypothesis values. (Input)
If NU = 0, then GP is NHP by NDEP; otherwise, GP is NHP by NU.

R — NCOEF by NCOEF upper triangular matrix containing the R matrix. (Input)
The R matrix can come from a regression fit based on a QR decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of R that is zero must also be zero. A zero row indicates a nonfull rank model. For an R matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of R corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of R must have positive diagonal elements. Only the upper triangle of R is referenced.

IRANKP — Rank of H_p. (Output)

NH — Number of rows in the completely testable hypothesis (also, the degrees of freedom for the hypothesis). (Output)
The degrees of freedom for the hypothesis (**NH**) classify the hypothesis H_p *BU* = G_p as nontestable (**NH** = 0), partially testable (0 < **NH** < **IRANKP**), or completely testable (0 < **NH** = **IRANKP**).

H — **NH** by **NCOEF** matrix *H* with each row corresponding to a row in the completely testable hypothesis and containing the constants that specify an estimable linear combination of the regression coefficients. (Output)
If **HP** is not needed, **H** and **HP** can occupy the same storage locations.

G — Matrix *G* containing the null hypothesis values for the completely testable hypothesis. (Output)
If **NU** = 0, then *G* is **NH** by **NDEP**, otherwise, *G* is **NH** by **NU**. If **GP** is not needed, **G** and **GP** can occupy the same storage locations.

Optional Arguments

NHP — Number of rows in the hypothesis. (Input)
Default: **NHP** = size (**HP**,1).

NCOEF — Number of regression coefficients in the model. (Input)
Default: **NCOEF** = size (**HP**,2).

LDHP — Leading dimension of **HP** exactly as specified in the dimension statement of the calling program. (Input)
Default: **LDHP** = size (**HP**,1).

LDGP — Leading dimension of **GP** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDGP** = size (**GP**,1).

LDR — Leading dimension of *R* exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDR** = size (**R**,1).

LDH — Leading dimension of **H** exactly as specified in the dimension statement of the calling program. (Input)
Default: **LDH** = size (**H**,1).

LDG — Leading dimension of **G** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDG** = size (**G**,1).

FORTRAN 90 Interface

Generic: CALL CESTI (HP, NDEP, NU, GP, R, IRANKP, NH, H, G [, ...])

Specific: The specific interface names are S_CESTI and D_CESTI.

FORTRAN 77 Interface

Single: CALL CESTI (NHP, NCOEF, HP, LDHP, NDEP, NU, GP, LDGP, R, LDR, IRANKP, NH, H, LDH, G, LDG)

Double: The double precision name is DCESTI.

Description

Once a general linear model $y = X\beta + \varepsilon$ is fitted, particular hypothesis tests are frequently of interest. If the matrix of regressors X is not full rank (as evidenced by the fact that some diagonal elements of the R matrix output from the fit are equal to zero), methods that use the results of the fitted model to compute the hypothesis sum of squares (see routine RHPSS) require one to specify in the hypothesis only linear combinations of the regression parameters that are estimable. A linear combination of regression parameters $c^T\beta$ is *estimable* means that there exists some vector a such that $c^T = a^TX$, i.e., c^T is in the space spanned by the rows of X. For a further discussion of estimable functions, see Maindonald (1984, pages 166-168) and Searle (1971, pages 180 - 188). Routine CESTI is only useful in the case of nonfull rank regression models, i.e., when the problem of estimability arises.

Peixoto (1986) noted that the customary definition of testable hypothesis in the context of a general linear hypothesis test $H\beta = g$ is overly restrictive. He extended the notion of a testable hypothesis (a hypothesis composed of estimable functions of the regression parameters) to include partially testable and completely testable hypotheses. A hypothesis $H\beta = g$ is *partially testable* means that the intersection of the row space of H (denoted by $R(H)$) and the row space of X($R(X)$) is not essentially empty and is a proper subset of $R(H)$, i.e., $\{0\} \subset R(H) \cap R(X) \subset R(H)$. A hypothesis $H\beta = g$ is *completely testable* means that $\{0\} \subset R(H) \subseteq R(X)$. Peixoto also demonstrated a method for converting a partially testable hypothesis to one that is completely testable so that the usual method for obtaining the sum of squares for the hypothesis from the results of the fitted model can be used. The method replaces H_p in the partially testable hypothesis $H_p\beta = g_p$ by a matrix H whose rows are a basis for the intersection of the row space of H_p and the row space of X. A corresponding conversion of the null hypothesis values from g_p to g is also made. A sum of squares for the completely testable hypothesis can then be computed (see routine RHPSS). The sum of squares that is computed for the hypothesis $H\beta = g$ equals the difference in the error sums of squares from two fitted models the restricted model with the partially testable hypothesis $H_p\beta = g_p$ adjoined to the model as linear equality restrictions (see routine RLEQU) and the unrestricted model.

Routines **RGLM**, **RGIVN**, **RLEQU**, and **RCOV** can be used to compute the fit of the general linear model prior to invoking **CESTI**. The *R* matrix is required for input to **CESTI**. After converting a partially testable hypothesis to a completely testable hypothesis, **RHPSS** can be invoked to compute the sum of squares for the hypothesis.

For the general case of the Multivariate General Linear Model $Y = XB + E$ with possible linear equality restrictions on the regression parameters, **CESTI** converts the partially testable hypothesis $H_p\ BU = G_p$ to a completely testable hypothesis $H\ BU = G$. For the case of the linear model with linear equality restrictions, the definitions of estimable functions, nontestable hypotheses, partially testable hypotheses, and completely testable hypothesis are similar to those previously given for the unrestricted model with the exception that $R(X)$ is replaced by $R(R)$ where R is the upper triangular matrix output from **RLEQU**. The nonzero rows of R form a basis for the rowspace of the matrix $(X^T, A^T)^T$. The rows of H form an orthonormal basis for the intersection of two subspaces: the subspace spanned by the rows of H_p and the subspace spanned by the rows of R. The algorithm used by **CESTI** for computing the intersection of these two subspaces is based on an algorithm for computing angles between linear subspaces due to to Bjorck and Golub (1973). (See also Golub and Van Loan 1983, pages 429-430). The method is closely related to a canonical correlation analysis discussed by Kennedy and Gentle (1980, 56-565). The algorithm is as follows:

1. Compute a QR factorization of

$$H_p^T$$

with column permutations so that

$$H_p^T = Q_1 R_1 P_1^T$$

Here, P_1 is the associated permutation matrix that is also an orthogonal matrix. Determine the rank of H_p as the number of nonzero diagonal elements of R_1, say n_1. Partition $Q_1 = (Q_{11}, Q_{12})$ so that $Q_{11}$is the first $n_1$columns of Q_1. Set IRANKP = n_1.

2. Compute a *QR* factorization of the transpose of the *R* matrix input to **CESTI** with column permutations so that

$$R^T = Q_2 R_2 P_2^T$$

Determine the rank of *R* from the number of nonzero diagonal elements of *R*, say n_2. Partition $Q_2 = (Q_{21}, Q_{22})$ so that Q_{21} is the first n_2 columns of Q_2.

3. Form

$$A = Q_{11}^T Q_{21}$$

4. Compute the singular values of *A*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(n_1,n_2)}$$

and the left singular vectors W of the singular value decomposition of A so that

$$W^T AV = diag\left(\sigma_1, \ \ldots, \ \sigma_{\min(n_1,n_2)}\right)$$

If $\sigma_1 < 1$, then the dimension of the intersection of the two subspaces is $s = 0$. Otherwise, take the dimension of the intersection to be s if $\sigma_s = 1 > \sigma_{s+1}$. Set NH = s.

5. Let W_1 be the first s columns of W. Set $H = (Q_1 W_1)^T$.

6. Take R_{11} to be a **NHP** by **NHP** matrix related to R_1 as follows. If **NHP** ≤ **NCOEF**, R_{11} equals the first **NHP** rows of R_1. Otherwise, R_{11} contains R_1 in its first **NCOEF** rows and zeros in the remaining rows. Compute a solution Z to the linear system

$$R_{11}^T Z = P_1^T G_p$$

using routine GIRTS (IMSL MATH/LIBRARY). If this linear system is declared inconsistent, an error message with error code equal to 2 is issued.

7. Partition

$$Z^T = \left(Z_1^T, Z_2^T\right)$$

so that Z_1 is the first n_1 rows of Z. Set

$$G = W_1^T Z_1$$

The degrees of freedom (**NH**) classify the hypothesis H_p $BU = G_p$ as nontestable (**NH** = 0), partially testable (0 < **NH** < **IRANKP**), or completely testable (0 < **NH** = **IRANKP**).

For further details concerning the algorithm, see Sallas and Lionti (1988).

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2STI/DC2STI**. The reference is:

   CALL C2STI (NCOEF, NHP, HP, LDHP, NDEP, NU, GP, LDGP, R, LDR, IRANKP, NH, H, LDH, G, LDG, IWK, WK)

   The additional arguments are as follows:

   IWK — Work vector of length max{NHP, NCOEF}.

WK — Work vector of length NCOEF * m + NCOEF^2 + NHP^2 + n * r + n^2 + m + max{2 * m, n + r + max(n, r) - 1}.

2. Informational errors

Type	Code	Description
4	1	There is inadequate space to store the completely testable hypothesis. Increase LDH or LDG so that it is greater than or equal to NH.
3	2	The hypothesis H_p BU = G_p is inconsistent.

Example

A one-way analysis-of-variance model discussed by Peixoto (1986) is fitted to some data. The model is

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij} \qquad (i, j) = (1, 1), (2, 1), (2, 2)$$

The model is fitted using routine **RGLM**. Next, the partially testable hypothesis

$$H_0: \begin{matrix} \alpha_1 = 5 \\ \alpha_2 = 3 \end{matrix}$$

is converted to a completely testable hypothesis using CESTI. Sum of squares associated with the hypothesis are computed using routine **RHPSS**. Finally, the F statistic is computed along with the associated p-value using routine FDF (see Chapter 17, "Probability Distribution Functions and Inverses").

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDB, LDG, LDGP, LDH, LDHP, LDR, LDSCPE, LDSCPH, &
                 LDX, LINDEF, MAXB, NCOL, NDEP, NEF,NHP, NROW, MAXCL, &
                 NCLVAR, J, NU
      PARAMETER  (LINDEF=1, MAXB=3, MAXCL=2, NCLVAR=1, NCOL=2, &
                 NDEP=1, NEF=1, NHP=2, NROW=3, LDB=MAXB, LDG=NHP, &
                 LDGP=NHP, LDH=NHP, LDHP=NHP, LDR=MAXB, LDSCPE=NDEP, &
                 LDSCPH=NDEP, LDX=NROW)
!
      INTEGER    INDCL(NCLVAR), INDDEP(NDEP),INDEF(LINDEF), INTCEP, &
      IRANK, IRANKP, IRBEF(NEF+1),NCOEF, NH, NOUT, NVEF(NEF)
      REAL       B(LDB,NDEP), DFE, DFH, F, G(LDG,NDEP), GP(LDGP,NDEP),&
                   H(LDH,MAXB), HP(LDHP,MAXB), PVALUE, R(LDR,MAXB), &
                   SCPE(LDSCPE,NDEP), SCPH(LDSCPH,NDEP),X(LDX,NCOL)
!
      DATA X/1.0, 2.0, 2.0, 17.3, 24.1, 26.3/
      DATA INDCL/1/, NVEF/1/, INDEF/1/, INDDEP/2/
      DATA (HP(1,J),J=1,MAXB)/0.0, 1.0, 0.0/
      DATA (HP(2,J),J=1,MAXB)/0.0, 0.0, 1.0/
      DATA GP/5.0, 3.0/
!
      CALL RGLM (X, INDCL, NVEF, INDEF, NDEP, INDDEP, MAXCL, B, &
      IRBEF=IRBEF, R=R, DFE=DFE, SCPE=SCPE)
      NCOEF = IRBEF(NEF+1) - 1
```

```
!
      NU = 0
      CALL CESTI (HP, NDEP, NU, GP, R, IRANKP, NH, H, G, NCOEF=NCOEF)
!
      CALL UMACH (2, NOUT)
      IF (NH .EQ. 0) THEN
         WRITE (NOUT,*) 'Nontestable hypothesis'
      ELSE IF (NH .LT. IRANKP) THEN
         WRITE (NOUT,*) 'Partially testable hypothesis'
      ELSE
         WRITE (NOUT,*) 'Completely testable hypothesis'
      END IF
      CALL WRRRN ('H', H, NH, NCOEF, LDH)
      CALL WRRRN ('G', G, NH, NDEP, LDG)
      CALL RHPSS (H, B, G, R, SCPH, DFH=DFH)
!
      F      = (SCPH(1,1)/DFH)/(SCPE(1,1)/DFE)
      PVALUE = 1.0 - FDF(F,DFH,DFE)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'Degrees of    Sum of                 Prob. of'
      WRITE (NOUT,*) '   Freedom   Squares   F-statistic   Larger F'
      WRITE (NOUT,99999) DFH, SCPH(1,1), F, PVALUE
99999 FORMAT (F8.1, 3X, 1F10.3, F11.3, 2X, F10.4)
      END
```

Output

```
Partially testable hypothesis

            H
     1        2        3
0.0000   0.7071  -0.7071

  G
1.414

Degrees of    Sum of                Prob. of
   Freedom   Squares   F-statistic   Larger F
     1.0       65.340     27.000      0.1210
```

RHPSS

more...

Computes the matrix of sums of squares and crossproducts for the multivariate general linear hypothesis *H BU* = *G* given the coefficient estimates

$$\hat{B}$$

and the *R* matrix.

Required Arguments

H — **NH** by **NCOEF** matrix *H* with each row corresponding to a row in the hypothesis and containing the constants that specify an estimable linear combination of the regression coefficients. (Input)

B — **NCOEF** by **NDEP** matrix

$$\hat{B}$$

containing a least-squares solution for the regression coefficients. (Input)

G — Matrix containing the null hypothesis values. (Input)
If **NU** = 0, then *G* is **NH** by **NDEP**; otherwise, *G* is **NH** by **NU**.

R — **NCOEF** by **NCOEF** upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements. Only the upper triangle of *R* is referenced.

SCPH — Matrix containing sums of squares and crossproducts attributable to the hypothesis. (Output)
If **NU** = 0, **SCPH** is a **NDEP** by **NDEP** matrix, otherwise, **SCPH** is a **NU** by **NU** matrix.

Optional Arguments

NH — Number of rows in the hypothesis. (Input)
Default: **NH** = size (**H**,1).

NCOEF — Number of regression coefficients in the model. (Input)
Default: **NCOEF** = size (**H**,2).

LDH — Leading dimension of *H* exactly as specified in the dimension statement of the calling program. (Input)
Default: **LDH** = size (**H**,1).

NDEP — Number of dependent (response) variables. (Input)
Default: **NDEP** = size (**B**,2).

LDB — Leading dimension of *B* exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDB** = size (**B**,1).

NU — *U* matrix option. (Input)
For positive **NU**, **NU** is the number of linear combinations of the dependent variables to be considered. If **NU** = 0, the hypothesis is *HB* = *G*, i.e., *U* is automatically taken to be the identity. **NU** must be less than or equal to **NDEP**.
Default: **NU** = 0.

U — **NDEP** by **NU** matrix *U* in test *H BU* = *G*. (Input, if **NU** is positive)
If **NU** = 0, **U** is not referenced and can be a 1 x 1 array.
Default: **U** is a 1x 1 array.

LDU — Leading dimension of **U** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDU** = size (**U**, 1,).

LDG — Leading dimension of **G** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDG** = size (**G**,1).

LDR — Leading dimension of **R** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDR** = size (**R**,1).

DFH — Degrees of freedom for **SCPH**. (Output)
DFH equals the rank of *H*.

LDSCPH — Leading dimension of **SCPH** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDSCPH** = size (**SCPH**,1).

FORTRAN 90 Interface

Generic: **CALL RHPSS** (**H**, **B**, **G**, **R**, **SCPH** [, ...])
Specific: The specific interface names are **S_RHPSS** and **D_RHPSS**.

FORTRAN 77 Interface

Single: **CALL RHPSS** (**NH**, **NCOEF**, **H**, **LDH**, **NDEP**, **B**, **LDB**, **NU**, **U**, **LDU**, **G**, **LDG**, **R**, **LDR**, **DFH**, **SCPH**, **LDSCPH**)
Double: The double precision name is **DRHPSS**.

Description

Routine **RHPSS** computes the matrix of sums of squares and crossproducts for the general linear hypothesis $HBU = G$ for the multivariate general linear model $Y = XB + E$ with possible linear equality restrictions $AB = Z$. (See the chapter introduction for a description of the Multivariate General Linear Model.) Routines **RGLM**, **RGIVN**, **RLEQU**, and **RCOV** can be used to compute the fit of the general linear model prior to invoking **RHPSS**. The R matrix and $\hat{B}$ from any of those routines are required for input to **RHPSS**.

The rows of H must be linear combinations of the rows of R, i.e., $HB = G$ must be completely testable. If the hypothesis is not completely testable, Routine **CESTI** can be used to construct an equivalent completely testable hypothesis.

Computations are based on an algorithm discussed by Kennedy and Gentle (1980, page 317) that is extended by Sallas and Lionti (1988) for multivariate nonfull rank models with possible linear equality restrictions. The algorithm is as follows:

1. Form

$$W = H\hat{B}U - G$$

2. Find C as the solution of $R^TC = H^T$ using routine **GIRTS** (IMSL MATH/LIBRARY). If the equations are declared inconsistent within a computed tolerance, an error message with code 1 is issued that the hypothesis is not completely testable.

3. For all rows of R corresponding to restrictions, i.e., containing negative diagonal elements from a restricted least-squares fit using **RLEQU**, zero out the corresponding rows of C, i.e., form DC.

4. Decompose *DC* using Householder transformations and column pivoting to yield a square, upper triangular matrix *T* with diagonal elements of nonincreasing magnitude and permutation matrix *P* such that

$$DCP = Q\begin{bmatrix} T \\ 0 \end{bmatrix}$$

where *Q* is an orthogonal matrix.

5. Determine the rank of *T*, say *r*. If $t_{11} = 0$, then $r = 0$. Otherwise, the rank of *T* is *r* if

$$|t_{rr}| > |t_{11}|\varepsilon \geq |t_{r+1,\ r+1}|$$

where **ε** = 10.0 * AMACH(4). Then, zero out all rows of *T* below row *r*. Set the degrees of freedom for the hypothesis, output in DFH, to *r*.

6. Find *V* as a solution to $T^{\mathsf{T}}V = P^{\mathsf{T}}W$ using routine **GIRTS**. If the equations are inconsistent, an error message with code 2 is issued that the hypothesis is inconsistent within a computed tolerance, i.e., the linear system

$$HBU = G$$
$$AB = Z$$

does not have a solution for *B*.

7. Form $V^{\mathsf{T}}V$, which is the required matrix of sum of squares and crossproducts output in **SCPH**.

   In general, the two errors with code 1 and 2 are serious user errors that require the user to correct the hypothesis before any meaningful sums of squares from this routine can be computed. However, in some cases, the user may know the hypothesis is consistent and completely testable, but the checks in **RHPSS** are too tight. For this reason, **RHPSS** continues with the computations.

   Routine **RHPSS** gives a matrix of sums of squares and crossproducts that could also be obtained from separate fittings of the two models

$$Y^* = XB^* + E^*$$
$$AB^* = Z^* \qquad (1)$$
$$HB^* = G$$

and

$$Y^* = XB^* + E^*$$
$$AB^* = Z^* \qquad (2)$$

where *Y** = *YU*, *B** = *BU*, *E** = *EU*, and *Z** = *ZU*. The error sum of squares and crossproduct matrix for (1) minus that for (2) is the matrix of sum of squares and crossproducts output in SCPH. Note that this approach avoids entirely the question of testability.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2PSS/DR2PSS. The reference is:

   CALL R2PSS (NCOEF, NH, H, LDH, NDEP, B, LDB, NU, U, LDU, G, LDG, R, LDR, DFH, SCPH, LDSCPH, IWK, WK)

   The additional arguments are as follows:

   IWK — Work vector of length NH.

   WK — Work vector of length NH * (NDEP + NCOEF + max(NCOEF, NH) + 3) + NU * NDEP - 1.

2. Informational errors

Type	Code	Description
3	1	The hypothesis is not completely testable. Each row of *H* must be in the space spanned by the rows of *R*.
3	2	The hypothesis is inconsistent. The linear system *HB U* = *G* combined with any restrictions from a regression fit with linear equality restrictions must have a solution for *B*.

3. $SCPH = \left(H\hat{B}U - G\right)^T \left(C^T DC\right)^- \left(H\hat{B}U - G\right)$
   where $(C^T DC)^-$ is a generalized inverse of $C^T DC$, *C* is a solution to $R^T C = H^T$, and *D* is a diagonal matrix with

$$d_{ii} = \begin{cases} 1 & \text{if } r_{ii} > 0 \\ 0 & \text{if } r_{ii} \le 0 \end{cases}$$

Examples

Example 1

A two-way analysis-of-variance model is fitted to balanced data discussed by Snedecor and Cochran (1967, Table 12.5.1, page 347). The responses are the weight gains (in grams) of rats fed diets varying in two components-level of protein and source of protein. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \qquad i = 1, 2; j = 1, 2, 3; k = 1, 2, \ldots, 10$$

where

$$\sum_{i=1}^{2}\alpha_i = 0;\ \sum_{j=1}^{3}\beta_j = 0;\ \sum_{i=1}^{2}\gamma_{ij} = 0 \text{ for } j = 1,2,3; \text{ and } \sum_{j=1}^{3}\gamma_{ij} = 0 \text{ for } i = 1,\ 2$$

The model is fitted using routine **RGLM**. Next, the sum of squares for interaction

$$H_0:\ \begin{matrix}\gamma_{11} - \gamma_{12} - \gamma_{21} + \gamma_{22} = 0\\ \gamma_{11} - \gamma_{13} - \gamma_{21} + \gamma_{23} = 0\end{matrix}$$

is computed using **RHPSS**. Finally, the *F* statistic is computed along with the associated *p*-value using routine **FDF** (see Chapter 17, "Probability Distribution Functions and Inverses").

```
      USE RHPSS_INT
      USE RGLM_INT
      USE UMACH_INT
      USE FDF_INT

      IMPLICIT   NONE
      INTEGER    LDB, LDG, LDH, LDR, LDSCPE, LDSCPH, LDU, LDX, LINDEF, &
                 MAXB, NCOL, NDEP, NEF, NH, NROW, MAXCL, NCLVAR, J
      PARAMETER  (NDEP=1, LINDEF=4, MAXB=12, MAXCL=5, NCLVAR=2, NCOL=3, &
                  NEF=3, NH=2, NROW=60, LDB=MAXB, LDG=NH, LDH=NH, &
                  LDR=MAXB, LDSCPE=NDEP, LDSCPH=NDEP, LDX=NROW)
!
      INTEGER   INDCL(NCLVAR), INDDEP(NDEP), INDEF(LINDEF), INTCEP,&
                 IRANK, IRBEF(NEF+1), NCOEF, NOUT, NVEF(NEF)
      REAL       B(LDB,NDEP), DFE, DFH, F, G(LDG,NDEP), H(LDH,MAXB), &
                 PVALUE, R(LDR,MAXB), SCPE(LDSCPE,NDEP), &
                 SCPH(LDSCPH,NDEP),X(LDX,NCOL), XMAX(MAXB), &
                 XMIN(MAXB)
!
      DATA X/73.0, 102.0, 118.0, 104.0, 81.0, 107.0, 100.0, 87.0, &
           117.0, 111.0, 98.0, 74.0, 56.0, 111.0, 95.0, 88.0, 82.0, &
           77.0, 86.0, 92.0, 94.0, 79.0, 96.0, 98.0, 102.0, 102.0, &
           108.0, 91.0, 120.0, 105.0, 90.0, 76.0, 90.0, 64.0, 86.0, &
           51.0, 72.0, 90.0, 95.0, 78.0, 107.0, 95.0, 97.0, 80.0, &
           98.0, 74.0, 74.0, 67.0, 89.0, 58.0, 49.0, 82.0, 73.0, 86.0, &
           81.0, 97.0, 106.0, 70.0, 61.0, 82.0, 30*1.0, 30*2.0, &
          10*1.0, 10*2.0, 10*3.0, 10*1.0, 10*2.0, 10*3.0/
      DATA INDCL/2, 3/, NVEF/1, 1, 2/, INDEF/2, 3, 2, 3/, INDDEP/1/
      DATA (H(1,J),J=1,MAXB)/6*0.0, 1.0, -1.0, 0.0, -1.0, 1.0, 0.0/
      DATA (H(2,J),J=1,MAXB)/6*0.0, 1.0, 0.0, -1.0, -1.0, 0.0, 1.0/
      DATA G/2*0.0/
!
      CALL RGLM (X, INDCL, NVEF, INDEF, NDEP, INDDEP, MAXCL, B, &
                 IRBEF=IRBEF, R=R, DFE=DFE, SCPE=SCPE)
!
      NCOEF = IRBEF(NEF+1) - 1
      CALL RHPSS (H, B, G, R, SCPH, DFH=DFH)
!
      F      = (SCPH(1,1)/DFH)/(SCPE(1,1)/DFE)
      PVALUE = 1.0 - FDF(F,DFH,DFE)
      CALL UMACH (2, NOUT)
```

```
      WRITE (NOUT,*) 'Degrees of    Sum of                 Prob. of'
      WRITE (NOUT,*) '   Freedom   Squares   F-statistic   Larger F'
      WRITE (NOUT,99999) DFH, SCPH(1,1), F, PVALUE
99999 FORMAT (F8.1, 3X, 1F10.3, F11.3, 2X, F10.4)
      END
```

Output

```
 Degrees of    Sum of                 Prob. of
    Freedom   Squares   F-statistic   Larger F
      2.0     1178.135      2.746      0.0732
```

Example 2

The data for the second example are taken from Maindonald (1984, pages 203-204). The data are saved in the matrix X. A multivariate regression model containing two dependent variables and three independent variables is fit using routine RGIVN. The sum of squares and crossproducts matrix is computed for the third independent variable in the model.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    INTCEP, LDB, LDG, LDH, LDR, LDSCPE, LDSCPH, LDX, &
                 NCOEF, NCOL, NDEP, NH, NIND, NROW, J, LDU
      PARAMETER  (INTCEP=1, LDU=1, NCOL=5, NDEP=2, NH=1, NIND=3, &
                 NROW=9, LDG=NH, LDH=NH, LDSCPE=NDEP, LDSCPH=NDEP, &
                 LDX=NROW, NCOEF=INTCEP+NIND, LDB=NCOEF, LDR=NCOEF)
!
      INTEGER    IDEP, IIND, INDDEP(1), INDIND(1),&
                 NOUT, NRMISS
      REAL       B(LDB,NDEP), D(NCOEF), DFE, DFH, G(LDG,NDEP), &
                 H(LDH,NCOEF), R(LDR,NCOEF), SCPE(LDSCPE,NDEP), &
                 SCPH(LDSCPH,NDEP), X(LDX,NCOL)
!
      DATA (X(1,J),J=1,NCOL)/7.0, 5.0, 6.0, 7.0, 1.0/
      DATA (X(2,J),J=1,NCOL)/2.0, -1.0, 6.0, -5.0, 4.0/
      DATA (X(3,J),J=1,NCOL)/7.0, 3.0, 5.0, 6.0, 10.0/
      DATA (X(4,J),J=1,NCOL)/-3.0, 1.0, 4.0, 5.0, 5.0/
      DATA (X(5,J),J=1,NCOL)/2.0, -1.0, 0.0, 5.0, -2.0/
      DATA (X(6,J),J=1,NCOL)/2.0, 1.0, 7.0, -2.0, 4.0/
      DATA (X(7,J),J=1,NCOL)/-3.0, -1.0, 3.0, 0.0, -6.0/
      DATA (X(8,J),J=1,NCOL)/2.0, 1.0, 1.0, 8.0, 2.0/
      DATA (X(9,J),J=1,NCOL)/2.0, 1.0, 4.0, 3.0, 0.0/
      DATA H/3*0.0, 1.0/, G/0.0, 0.0/
!
      IIND = -NIND
      IDEP = -NDEP
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R)
      CALL RHPSS (H, B, G, R, SCPH, DFH=DFH)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'DFH = ', DFH
      CALL WRRRN ('SCPH', SCPH)
      END
```

Output

```
     DFH =     1.00000

          SCPH
             1       2
     1   100.0   -40.0
     2   -40.0    16.0
```

RHPTE

more...

Performs tests for a multivariate general linear hypothesis $H\,BU = G$ given the hypothesis sums of squares and crossproducts matrix S_H and the error sums of squares and crossproducts matrix S_E.

Required Arguments

DFE — Degrees of freedom for error matrix SCPE. (Input)

SCPE — NDEP by NDEP matrix S_E containing sums of squares and crossproducts for error. (Input)

DFH — Degrees of freedom for hypothesis matrix S_H. (Input)

SCPH — Matrix S_H containing sums of squares and crossproducts attributable to the hypothesis. (Input)
If NU = 0, S_H is a NDEP by NDEP matrix; otherwise, S_H is a NU by NU matrix.

TEST — Vector of length 8 containing test statistics and *p*-values for the hypothesis $H\,BU = G$. (Output)

Elem	Description
1, 5	Wilks' lambda and *p*-value
2, 6	Roy's maximum root criterion and *p*-value
3, 7	Hotelling's trace and *p*-value
4, 8	Pillai's trace and *p*-value

Optional Arguments

NDEP — Number of dependent variables. (Input)
Default: NDEP = size (SCPE,2).

LDSCPE — Leading dimension of SCPE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCPE = size (SCPE,1).

NU — *U* matrix option. (Input)
For positive **NU**, **NU** is the number of linear combinations of the dependent variables to be considered. If **NU** = 0, the hypothesis is *HB* = *G*, i.e., *U* is automatically taken to be the identity.
Default: **NU** = 0.

U — **NDEP** by **NU** matrix used to test *H BU* = *G*. (Input, if **NU** is positive)
The rank of the matrix *U* must equal the number of columns. If **NU** = 0, *U* is not referenced and can be a 1 × 1 array.
Default: **U** is a 1 × 1 array.

LDU — Leading dimension of **U** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDU** = size(**U**, 1).

LDSCPH — Leading dimension of **SCPH** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDSCPH** = size (**SCPH**,1).

FORTRAN 90 Interface

Generic: CALL RHPTE (DFE, SCPE, DFH, SCPH, TEST [, ...])
Specific: The specific interface names are S_RHPTE and D_RHPTE.

FORTRAN 77 Interface

Single: CALL RHPTE (DFE, NDEP, SCPE, LDSCPE, NU, U, LDU, DFH, SCPH, LDSCPH, TEST)
Double: The double precision name is DRHPTE.

Description

Routine **RHPTE** computes test statistics and *p*-values for the general linear hypothesis *H BU* = *G* for the multivariate general linear model. See the section "Multivariate General Linear Model" in the chapter introduction.

Routines RGLM, RGIVN, RLEQU, and RCOV can be used to compute the fit of the general linear model prior to invoking **RHPTE**. The error sum of squares and crossproducts matrix (**SCPE**) is required for input to **RHPTE**. In addition, the hypothesis sum of squares and crossproducts matrix (**SCPH**), which can be computed using routine RHPSS, is required for input to **RHPTE**.

The hypothesis sum of squares and crossproducts matrix input in **SCPH** is

$$S_H = \left(H\hat{B}U - G\right)^T \left(C^T DC\right)^- \left(H\hat{B}U - G\right)$$

where C is a solution to $R^T C = H$ and where D is a diagonal matrix with diagonal elements

$$d_{ii} = \begin{cases} 1 & \text{if } r_{ii} > 0 \\ 0 & \text{otherwise} \end{cases}$$

See the section "Linear Dependence and the R Matrix" in the chapter introduction.

The error sum of squares and crossproducts matrix for the model $Y = XB + E$ is

$$\left(Y - X\hat{B}\right)^T \left(Y - X\hat{B}\right)$$

which is input in **SCPE**. The error sum of squares and crossproducts matrix for the hypothesis $H\,BU = G$ computed by **RHPTE** is

$$S_{\mathrm{E}} = U^{\mathrm{T}} \left(Y - X\hat{B}\right)^{\mathrm{T}} \left(Y - X\hat{B}\right) U$$

Let p equal the order of the matrices S_{E} and S_{H}, i.e.,

$$p = \begin{cases} \mathtt{NU} & \text{if } \mathtt{NU} > 0 \\ \mathtt{NDEP} & \text{otherwise} \end{cases}$$

Let q (stored in **DFH**) be the degrees of freedom for the hypothesis. Let v (stored in **DFE**) be the degrees of freedom for error. Routine **RHTPE** computes three test statistics based on eigenvalues λ_i (i = 1, 2, …, p) of the generalized eigenvalue problem $S_H x = \lambda S_E x$. These test statistics are as follows:

Wilks' lambda

$$\Lambda = \frac{\det(S_{\mathrm{E}})}{\det(S_H + S_E)} = \prod_{i=1}^{p} \frac{1}{1+\lambda_i}$$

Λ is output in **TEST**(1). The p-value output in **TEST**(5) is based on an approximation discussed by Rao (1973, page 556). The statistic

$$F = \frac{ms - pq/2 + 1}{pq} \quad \frac{1 - \Lambda^{1/s}}{\Lambda^{1/s}}$$

has an approximate *F* distribution with *pq* and *ms* - *pq*/2 + 1 numerator and denominator degrees of freedom, respectively, where

$$s = \begin{cases} 1 & \text{if } p = 1 \text{ or } q = 1 \\ \sqrt{\frac{p^2q^2 - 4}{p^2 + q^2 - 5}} & \text{otherwise} \end{cases}$$

and

$$m = v - (p - q + 1)/2$$

The *F* test is exact if min(*p*, *q*) ≤ 2 (Kshirsagar 1972, Theorem 4, pages 299-300).

Roy's maximum root

$$c = \max_i \lambda_i$$

c is output in **TEST**(2). The *p*-value output in **TEST**(6) is based on the approximation

$$F = \frac{v + q - s}{s} c$$

where *s* = max(*p*, *q*) has an approximate *F* distribution with *s* and *v* + *q* - *s* numerator and denominator degrees of freedom, respectively. The *F* test is exact if *s* = 1, and then the *p*-value output in **TEST**(7) is exact. In general, the value output in **TEST**(7) is a lower bound on the actual *p*-value.

Hotelling's trace

$$U = tr\left(HE^{-1}\right) = \sum_{i=1}^{p} \lambda_i$$

U is output in **TEST**(3). The *p*-value output in **TEST**(7) is based on the approximation of McKeon (1974) that supersedes the approximation of Hughes and Saw (1972). McKeon's approximation is also discussed by Seber (1984, page 39). For

$$b = 4 + \frac{pq + 2}{\frac{(v+q-p-1)(v-1)}{(v-p-3)(v-p)} - 1}$$

the *p*-value output in **TEST**(7) is based on the result that

$$F = \frac{b(v-p-1)}{(b-2)pq}U$$

has an approximate *F* distribution with *pq* and *b* degrees of freedom. The test is exact if min(*p*, *q*) = 1. For $v \le p + 1$, the approximation is not valid, and **TEST**(7) is set to NaN (not a number).

These three test statistics are valid when S_E is positive definite. A necessary condition for S_E to be positive definite is $v \ge p$. If S_E is not positive definite, a warning error message with error code 1 is issued, and the entries in **TEST** corresponding to the computed test statistics and *p*-values are set to NaN (not a number).

Because the requirement $v \ge p$ can be a serious drawback, **RHTPE** computes a fourth test statistic based on eigenvalues $\theta_i (i = 1, 2, \ldots, p)$ of the generalized eigenvalue problem $S_H w = \theta (S_H + S_E) w$. This test statistic requires a less restrictive assumption—$S_H + S_E$ is positive definite. A necessary condition for $S_H + S_E$ to be positive definite is $v + q \ge p$. If S_E is positive definite, **RHPTE** avoids the computation of this generalized eigenvalue problem from scratch. In this case, the eigenvalues θ_i are obtained from λ_i by

$$\theta_i = \frac{\lambda_i}{1 + \lambda_i}$$

The fourth test statistic is as follows:

Pillai's trace

$$V = \mathrm{tr}\left[S_H (S_H + S_E)^{-1} \right]$$
$$= \sum_{i=1}^{p} \theta_i$$

V is output in **TEST**(4). The *p*-value output in **TEST**(8) is based on an approximation discussed by Pillai (1985). The statistic

$$F = \frac{2n + s + 1}{2m + s + 1} \quad \frac{V}{s - V}$$

has an approximate *F* distribution with *s*(2*m* + *s* + 1) and *s*(2*n* + *s* + 1) numerator and denominator degrees of freedom, respectively, where

$$s = \min(p, q)$$
$$m = \frac{1}{2}\left(|p - q| - 1 \right)$$
$$n = \frac{1}{2}(v - p - 1)$$

The *F* test is exact if min(*p*, *q*) = 1.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2PTE/DR2PTE**. The reference is:

   CALL R2PTE (**DFE**, **NDEP**, **SCPE**, **LDSCPE**, **NU**, **U**, **LDU**, **DFH**, **SCPH**, **LDSCPH**, **TEST**, **WK**)

   The additional argument is:

   WK — Work vector of length 2 * p^2 + 2 * p + NDEP + 2 * NU^2

2. Informational errors

Type	Code	Description
3	1	$U^T S_E U$ is singular. Only the Pillai trace statistic can be computed. Other statistics are set to NaN.
4	2	$U^T S_E U + S_H$ is singular. No tests can be computed.
4	3	Iterations for eigenvalues for the generalized eigenvalue problem $S_H x = \lambda(S_H + U^T S_E U)x$ failed to converge. Statistics cannot be computed.

Example

The data for the example are taken from Maindonald (1984, pages 203-204). The data are stored in the matrix **X**. A multivariate regression model containing two dependent variables and three independent variables is fit using routine **RGIVN**. The sum of squares and crossproducts matrix is computed for the third independent variable in the model using **RHPSS**. Routine **RHPTE** is used to test whether the third independent variable should be included in the regression.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    LDB, LDG, LDH, LDR, LDSCPE, LDSCPH, LDU, LDX, &
                     NCOEF, NCOL, NDEP, NH, NIND, NROW, J, INTCEP, IIND
          PARAMETER  (INTCEP=1, LDU=1, NCOL=5, NDEP=2, NH=1, NIND=3, &
                     NROW=9, LDG=NH, LDH=NH, LDSCPE=NDEP, LDSCPH=NDEP, &
                     LDX=NROW, NCOEF=INTCEP+NIND, LDB=NCOEF, LDR=NCOEF)
    !
          INTEGER    IDEP, IND, INDDEP(1), INDIND(1)
          REAL       B(LDB,NDEP), DFE, DFH, G(LDG,NDEP), &
                     H(LDH,NCOEF), R(LDR,NCOEF), SCPE(LDSCPE,NDEP), &
```

```
                    SCPH(LDSCPH,NDEP), TEST(8), X(LDX,NCOL)
          CHARACTER  CLABEL(3)*14, RLABEL(4)*9
    !
          DATA (X(1,J),J=1,NCOL)/7.0, 5.0, 6.0, 7.0, 1.0/
          DATA (X(2,J),J=1,NCOL)/2.0, -1.0, 6.0, -5.0, 4.0/
          DATA (X(3,J),J=1,NCOL)/7.0, 3.0, 5.0, 6.0, 10.0/
          DATA (X(4,J),J=1,NCOL)/-3.0, 1.0, 4.0, 5.0, 5.0/
          DATA (X(5,J),J=1,NCOL)/2.0, -1.0, 0.0, 5.0, -2.0/
          DATA (X(6,J),J=1,NCOL)/2.0, 1.0, 7.0, -2.0, 4.0/
          DATA (X(7,J),J=1,NCOL)/-3.0, -1.0, 3.0, 0.0, -6.0/
          DATA (X(8,J),J=1,NCOL)/2.0, 1.0, 1.0, 8.0, 2.0/
          DATA (X(9,J),J=1,NCOL)/2.0, 1.0, 4.0, 3.0, 0.0/
          DATA H/3*0.0, 1.0/, G/0.0, 0.0/
          DATA RLABEL/'Wilks', 'Roy', 'Hotelling', 'Pillai'/
          DATA CLABEL/' ', 'Test statistic', 'p-value'/
    !
          IIND = -NIND
          IDEP = -NDEP
          CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, SCPE=SCPE)
          CALL RHPSS (H, B, G, R, SCPH, DFH=DFH)
          CALL RHPTE (DFE, SCPE, DFH, SCPH, TEST)
          CALL WRRRL (' ', TEST, RLABEL, CLABEL, 4, 2, 4, FMT= '(F14.3, F9.6)')
          END
```

Output

```

             Test statistic    p-value
Wilks                 0.003   0.000010
Roy                 316.601   0.000010
Hotelling           316.601   0.000010
Pillai                0.997   0.000010
```

RLOFE

Computes a lack of fit test based on exact replicates for a fitted regression model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IREP — Variable option. (Input)

IREP	Meaning
< 0	The first -IREP columns of X contain the variables used to determine exact replicates.
> 0	The IREP variables used to determine exact replicates are specified by the column numbers in INDREP.
0	The exact replicates are specified in IGROUP.

INDREP — Index vector of length IREP containing the column numbers of X that are the variables used to determine replication. (Input, if IREP is positive) If IREP is less than or equal to 0, INDREP is not referenced and can be a vector of length one.

IRSP — Column number IRSP of X contains data for the response (dependent) variable. (Input)

DFE — Degrees of freedom for error from the fitted regression. (Input)

SSE — Sum of squares for error from the fitted regression. (Input)

IGROUP — Vector of length NOBS specifying group numbers. (Output, if IREP is nonzero; input, if IREP = 0)
On output, IGROUP(I) = J means row I of X is in the J-th group of replicates (J = 0, 1, 2, …, NGROUP). Here, J = 0 indicates the group of observations not used in the analysis because NaN (not a number) was input for one of more of the values of the response, replication, frequency, or weight variables. On input, IGROUP(I) = IGROUP(K), K ≠ I, indicates that row I and row K of X are in the same group. IGROUP(I) must equal 0 if row I of X has NaN as one or more of the values of the response, replication, frequency, or weight variables.

NGROUP — Number of groups in the lack of fit test. (Output)

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the model. (Output)

Elem	Description
1	Degrees of freedom for lack of fit
2	Degrees of freedom for pure error
3	Degrees of freedom for error (TESTLF(1)+ TESTLF(2))
4	Sum of squares for lack of fit
5	Sum of squares for pure error
6	Sum of squares for error
7	Mean square for lack of fit
8	Mean square for pure error
9	*F* statistic
10	*p*-value

If there are no replicates in the data set, a test for lack of fit cannot be performed. In this case, elements 8, 9, and 10 of TESTLF are set to NaN (not a number).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

FORTRAN 90 Interface

Generic: CALL RLOFE (X, IREP, INDREP, IRSP, DFE, SSE, IGROUP, NGROUP, TESTLF [, ...])

Specific: The specific interface names are S_RLOFE and D_RLOFE.

FORTRAN 77 Interface

Single: CALL RLOFE (NOBS, NCOL, X, LDX, IREP, INDREP, IRSP, IFRQ, IWT, DFE, SSE, IGROUP, NGROUP, TESTLF)

Double: The double precision name is DRLOFE.

Description

Routine RLOFE computes a lack of fit test based on exact replicates for a fitted regression model. The data need not be sorted prior to invoking RLOFE. The column indices of X for determining exact replicates can be input in INDREP. If the groups of exact replicates are known prior to invoking RLOFE, the option IREP = 0 allows RLOFE to bypass the computation of the groups. This option is particularly useful for computing a second lack of fit for a different dependent variable that uses the same columns of X for determining exact replicates as the first test.

If IREP is nonzero, routine SROWR (see Chapter 19, "Utilities") is used to compute a permutation vector that specifies the sorted X along with the n_i's, the number of rows of X in each group. If IREP is zero, the permutation vector and the n_i's are computed from IGROUP.

Let n_i be the number of rows of X in the i-th group of replicates (i = 1, 2, ..., k). Let y_{ij} be the response for the j-th row within the i-th group. Let w_{ij} and f_{ij} be the associated weight and frequency, respectively. The pure error (within group) sum of squares is

$$\text{SSPE} = \sum_{i=1}^{k}\sum_{j=1}^{n_i} w_{ij} f_{ij} \left(y_{ij} - \bar{y}_{i\bullet} \right)^2$$

The associated degrees of freedom are

$$\text{DFPE} = \left(\sum_{i=j}^{k}\sum_{j=1}^{n_i} f_{ij} \right) - k$$

The lack of fit sum of squares is SSE - SSPE and the lack of fit degrees of freedom are DFE - DFPE.

The *F* statistic for the test of the null hypothesis of no lack of fit is

$$F = \frac{(\mathrm{SSE} - \mathrm{SSPE}) / (\mathrm{DFE} - \mathrm{DFPE})}{\mathrm{SSPE} / \mathrm{DFPE}}$$

Under the hypothesis of no lack of fit, the computed *F* has an *F* distribution with numerator and denominator degrees of freedom DFE - DFPE and DFPE, respectively. The *p*-value for the test is computed as the probability that a random variable with this distribution is greater than or equal to the computed *F* statistic.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2OFE/DR2OFE. The reference is:

   CALL R2OFE (NOBS, NCOL, X, LDX, IREP, INDREP, IRSP, IFRQ, IWT, DFE, SSE, IGROUP, NGROUP, TESTLF, IWK, WK)

   The additional arguments are as follows:

   IWK — Work vector. If IREP = 0, the length of IWK is 3 * NOBS; otherwise, the length of IWK is |IREP| + *m* + 2.8854 * ln(*m*) + 3 * NOBS + 5.

   WK — Work vector. If IREP= 0, WK is not referenced and can be a vector of length 1; otherwise, WK is of length 2 * *m*.

2. Informational errors

Type	Code	Description
3	1	DFE is less than the degrees of freedom for pure error. The degrees of freedom for lack of fit is set to zero.
3	2	SSE is less than the sum of squares for pure error. The sum of squares for lack of fit is set to zero.
4	3	An invalid weight or frequency is encountered. Weights and frequencies must be nonnegative.
4	4	An element in X contains NaN (not a number), but the corresponding element in IGROUP is not zero. When IREP = 0, missing values in a row of X are indicated by setting the corresponding row of IGROUP to zero.

Examples

Example 1

This example uses data from Draper and Smith (1981, page 374), which is input in X. A multiple linear regression of column 6 of X on an intercept and columns 1, 3, and 4 has already been computed. The fit gave a residual sum of squares SSE = 163.93 with DFE = 16 degrees of freedom. A test for lack of fit is computed using routine RLOFE.

```
      USE RLOFE_INT
      USE UMACH_INT
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    LDX, NCOL, NOBS, NREP, J
      PARAMETER  (NCOL=6, NOBS=20, NREP=3, LDX=NOBS)
!
      INTEGER    IGROUP(NOBS), INDREP(NREP), IREP, IRSP,  &
                 NGROUP, NOUT
      REAL       DFE, SSE, TESTLF(10), X(LDX,NCOL)
!
      DATA (X(1,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 246.0/
      DATA (X(2,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(3,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 253.0/
      DATA (X(4,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 164.0/
      DATA (X(5,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 203.0/
      DATA (X(6,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 173.0/
      DATA (X(7,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 210.0/
      DATA (X(8,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(9,J),J=1,6)/0.0, 1.0, 0.0, 1.0, 0.0, 120.0/
      DATA (X(10,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 171.0/
      DATA (X(11,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 167.0/
      DATA (X(12,J),J=1,6)/0.0, 0.0, 1.0, 1.0, 0.0, 172.0/
      DATA (X(13,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(14,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(15,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 248.0/
      DATA (X(16,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 169.0/
      DATA (X(17,J),J=1,6)/0.0, 1.0, 0.0, 0.0, 0.0, 104.0/
      DATA (X(18,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 166.0/
      DATA (X(19,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 168.0/
      DATA (X(20,J),J=1,6)/0.0, 1.0, 1.0, 0.0, 0.0, 148.0/
      DATA INDREP/1, 3, 4/
!
      IREP = NREP
      IRSP = 6
      DFE  = 16.0
      SSE  = 163.93
      CALL RLOFE (X, IREP, INDREP, IRSP, DFE, SSE, IGROUP, NGROUP, TESTLF)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' NGROUP = ', NGROUP
      CALL WRIRN ('IGROUP', IGROUP, 1, NOBS, 1)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,99999) '                    Test for Lack of '// &
                         'Fit'
      WRITE (NOUT,99999) '                         Sum of    Mean  '// &
                         '      Prob. of'
      WRITE (NOUT,99999) ' Source of Error   DF  Squares  Square  '// &
                         '    F  Larger F'
      WRITE (NOUT,99999) ' Lack of Fit    ', TESTLF(1), TESTLF(4), &
                         TESTLF(7), TESTLF(9), TESTLF(10)
      WRITE (NOUT,99999) ' Expanded model ', TESTLF(2), TESTLF(5), &
                         TESTLF(8)
      WRITE (NOUT,99999) ' Original model ', TESTLF(3), TESTLF(6)
99999 FORMAT (A, F5.1, F9.1, F8.2, F7.3, F10.3)
      END
```

Output

```
 NGROUP =   6

                                        IGROUP
 1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 6  6  6   4   5   4   5   6   2   4   4   4   6   6   6   4   1   4   4   3

                     Test for Lack of Fit
                          Sum of    Mean         Prob. of
 Source of Error   DF  Squares  Square      F  Larger F
 Lack of Fit      2.0     20.5   10.25   1.001  0.393
 Expanded model  14.0    143.4   10.24
 Original model  16.0    163.9
```

Example 2

This example uses the same data as in Example 1. Here, the option **IREP** = 0 is used because **IGROUP** is known before invoking routine **RLOFE**. Routine **SROWR** (see Chapter 19, "Utilities") is used to compute the group numbers contained in **IGROUP**.

```
      USE RLOFE_INT
      USE SROWR_INT
      USE UMACH_INT
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    LDX, NCOL, NKEY, NOBS, J
      PARAMETER  (NCOL=6, NKEY=3, NOBS=20, LDX=NOBS)
!
      INTEGER    I, IGROUP(NOBS), INDKEY(NKEY), &
                 INDREP(1), IPERM(NOBS), IREP, IRET, IRSP, &
                 K, NGROUP, NI(NOBS), NOUT, NRMISS
      REAL       DFE, SSE, TESTLF(10), X(LDX,NCOL)
!
      DATA (X(1,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 246.0/
      DATA (X(2,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(3,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 253.0/
      DATA (X(4,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 164.0/
      DATA (X(5,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 203.0/
      DATA (X(6,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 173.0/
      DATA (X(7,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 210.0/
      DATA (X(8,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(9,J),J=1,6)/0.0, 1.0, 0.0, 1.0, 0.0, 120.0/
      DATA (X(10,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 171.0/
      DATA (X(11,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 167.0/
      DATA (X(12,J),J=1,6)/0.0, 0.0, 1.0, 1.0, 0.0, 172.0/
      DATA (X(13,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(14,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(15,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 248.0/
      DATA (X(16,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 169.0/
      DATA (X(17,J),J=1,6)/0.0, 1.0, 0.0, 0.0, 0.0, 104.0/
      DATA (X(18,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 166.0/
      DATA (X(19,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 168.0/
      DATA (X(20,J),J=1,6)/0.0, 1.0, 1.0, 0.0, 0.0, 148.0/
      DATA INDKEY/1, 3, 4/
```

```
!
      IRET  = 1
      CALL SROWR (X, INDKEY, IPERM, NGROUP, NI, IRET=IRET)
      K = 1
      DO 20  I=1, NGROUP
         DO 10  J=1, NI(I)
            IGROUP(IPERM(K)) = I
            K = K + 1
   10    CONTINUE
   20 CONTINUE
      IREP = 0
      IRSP = 6
      DFE  = 16.0
      SSE  = 163.93
      CALL RLOFE (X, IREP, INDREP, IRSP, DFE, SSE, IGROUP, NGROUP, TESTLF)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' NGROUP = ', NGROUP
      CALL WRIRN ('IGROUP', IGROUP, 1, NOBS, 1)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,99999) '                    Test for Lack of '// &
                         'Fit'
      WRITE (NOUT,99999) '                         Sum of    Mean  '// &
                         '      Prob. of'
      WRITE (NOUT,99999) ' Source of Error   DF  Squares  Square  '// &
                         '    F  Larger F'
      WRITE (NOUT,99999) ' Lack of Fit    ', TESTLF(1), TESTLF(4), &
                         TESTLF(7), TESTLF(9), TESTLF(10)
      WRITE (NOUT,99999) ' Expanded model ', TESTLF(2), TESTLF(5), &
                         TESTLF(8)
      WRITE (NOUT,99999) ' Original model ', TESTLF(3), TESTLF(6)
99999 FORMAT (A, F5.1, F9.1, F8.2, F7.3, F10.3)
      END
```

Output

```
 NGROUP =   6

                                    IGROUP
 1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 6  6  6   4   5   4   5   6   2   4   4   4   6   6   6   4   1   4   4   3

                    Test for Lack of Fit
                         Sum of    Mean         Prob. of
 Source of Error   DF  Squares  Square      F  Larger F
 Lack of Fit      2.0     20.5   10.25  1.001     0.393
 Expanded model  14.0    143.4   10.24
 Original model  16.0    163.9
```

RLOFN

more...

Computes a lack of fit test based on near replicates for a fitted regression model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)

IIND	Meaning
< 0	The first -IIND columns of X contain the independent (explanatory) variables.
> 0	The IIND independent variables are specified by the column numbers in INDIND.
= 0	There are no independent variables.

There are NCOEF = INTCEP + |IIND| regressors—the intercept (if INTCEP = 1) and the indepen-dent variables.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent variables. (Input, if IIND is positive)
If IIND is nonnegative, INDIND is not referenced and can be a vector of length one.

IRSP — Column number IRSP of X contains data for the response (dependent) variable. (Input)

B — Vector of length NCOEF containing a least-squares solution

$$\hat{\beta}$$

for the regression coefficients. (Input)

R — NCOEF by NCOEF upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row

indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements. Only the upper triangle of R is referenced.

DFE — Degrees of freedom for error from the fitted regression. (Input)

SSE — Sum of squares for error from the fitted regression. (Input)

NGROUP — Number of groups. (Input)
A cluster analysis based on NGROUP groups is performed. A good choice for NGROUP is the number of groups of near replicates in the data set.

IGROUP — Vector of length NOBS specifying group numbers. (Input, if ICLUST = 0; Output, if ICLUST ≥ 1)
IGROUP(I) = J means row I of X is in the J-th group of near replicates (J = 0, 1, 2, …, NGROUP). Here, J = 0 indicates the group of observations not used in the analysis because NaN (not a number) was input for one or more of the values of the response, independent, frequency, or weight variables.

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the model. (Output)

Elem	Description
1	Degrees of freedom for lack of fit
2	Degrees of freedom for error from the expanded model (one-way analysis of covariance model using clusters of near replicates as the groups).
3	Degrees of freedom for error (DFE = TESTLF(1) + TESTLF(2)).
4	Sum of squares for lack of fit.
5	Sum of squares for error from the expanded model.
6	Sum of squares for error (SSE = TESTLF(4) + TESTLF(5)).
7	Mean square for lack of fit.
8	Mean square for error from the expanded model.
9	*F* statistic
10	*p*-value

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

ICLUST — Clustering option. (Input)
Default: ICLUST = 1.

ICLUST	Meaning
0	Cluster groups are input in IGROUP.
1	Cluster groups are obtained using Euclidean distance.
2	Cluster groups are obtained using Mahalanobis distance.

MAXIT — Maximum number of iterations for the cluster analysis to determine near replicates. (Input, if ICLUST is positive, otherwise, MAXIT is not referenced)
MAXIT = 30 is usually sufficient for convergence.
Default: MAXIT = 30.

TOL — Tolerance used in determining linear dependence for the one-way analysis of covariance model using clusters as the groups. (Input)
TOL = **EPS**$^{2/3}$ is a good choice. For **RLOFN**, **EPS** = **AMACH**(4). See documentation for **AMACH** in *Reference Material*.
Default: **TOL** = 2.4e-5 for single precision and 3.6d – 11 for double precision.

FORTRAN 90 Interface

Generic: **CALL RLOFN** (**X**, **IIND**, **INDIND**, **IRSP**, **B**, **R**, **DFE**, **SSE**, **NGROUP**, **IGROUP**, **TESTLF** [, ...])

Specific: The specific interface names are **S_RLOFN** and **D_RLOFN**.

FORTRAN 77 Interface

Single: **CALL RLOFN** (**NOBS**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IRSP**, **IFRQ**, **IWT**, **B**, **R**, **LDR**, **DFE**, **SSE**, **ICLUST**, **MAXIT**, **TOL**, **NGROUP**, **IGROUP**, **TESTLF**)

Double: The double precision name is **DRLOFN**.

Description

Routine **RLOFN** computes a lack of fit test based on near replicates for a fitted regression model. The data need not be sorted prior to invoking **RLOFN**. The column indices of **X** for determining near replicates must correspond to the independent variables in the original fitted model. If the groups of near replicates are known prior to invoking **RLOFN**, the option **ICLUST** = 0 allows **RLOFN** to bypass the computation of the groups.

The data can contain missing values indicated by NaN. (NaN is **AMACH**(6). Routine **AMACH** is described in the section "Machine-Dependent Constants" in the Reference Material.) For **ICLUST** equal to 1 or 2, any row of **X** containing NaN as a value for the response, weight, frequency, or independent variables is omitted from the analysis. For **ICLUST** equal to 0, if the *i*-th row of **X** contains NaN for one of the variables in the analysis, the *i*-th element of **IGROUP** must be 0 on input.

Routine **KMEAN** (see Chapter 11, "Cluster Analysis") is used to compute *k* clusters or groups of near replicates. Prior to invoking **KMEAN**, a detached sort of the independent variables in the regression model is performed using routine **SROWR** (See Chapter 19, "Utilities".) If there are fewer than **NGROUP** distinct observations, a warning message is issued and *k* is set equal to the number of distinct observations. Otherwise, *k* equals **NGROUP**. For purposes of the cluster analysis, **ICLUST** = 1 specifies Euclidean distance and **ICLUST** = 2 specifies Mahalanobis distance. For Mahalanobis distance, the data are transformed before invoking **KMEAN** so that the Euclidean metric applied by **KMEAN** for the transformed data is equivalent to the sample Mahalanobis distance for the original (untransformed) data.

Let X be the $n \times p$ matrix of regressors, and let R be the upper triangular matrix computed from the fitted regression model. The matrix R can be computed by routines **RGLM**, **RGIVN**, or **RLEQU** for fitting the regression model. A linear equality restriction on the regression parameters corresponds to a row of R with a negative diagonal element. Let D be a $p \times p$ diagonal matrix with diagonal elements

$$d_{ii} = \begin{cases} 1 & \text{if } r_{ii} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let

$$x_i^T$$

be the i-th row of X, and let $t_i = Ds_i$ where s_i satisfies

$$R^T s_i = x_i$$

Then, the Mahalanobis distance from x_i to x_j equals the Euclidean distance from t_i to t_j because

$$\begin{aligned} \sigma^{-2}\mathrm{Var}\left[\left(x_i - x_j\right)^T\hat{\beta}\right] &= \sigma^{-2}\mathrm{Var}\left[\left(s_i - s_j\right)^T R\hat{\beta}\right] \\ &= \sigma^{-2}\left(s_i - s_j\right)^T \mathrm{Var}\left(R\hat{\beta}\right)\left(s_i - s_j\right) \\ &= \left(s_i - s_j\right)^T D\left(s_i - s_j\right) \\ &= \left(t_i - t_j\right)^T\left(t_i - t_j\right) \end{aligned}$$

Once the clusters are identified by **KMEAN** an expanded regression model—a one-way analysis of covariance model–is fitted to the original (untransformed) data. Denote the original model by $y = X\beta + \varepsilon$ and the expanded model by $y = X\beta + Z\gamma + \varepsilon$. The added regressors that are contained in the $n \times k$ matrix Z in the expanded model are indicator variables specifying cluster membership. The lack of fit test that is computed is an exact test of the hypothesis that $\gamma = 0$ in the expanded model. This test was proposed as a lack of fit test by Christensen (1989).

Let SSE(X, Z) be the error sum of squares from the fit of the expanded model and let SSE(X) be the error sum of squares from the fit of the original model. The lack of fit sum of squares is SSE(X) - SSE(X, Z) and the lack of fit degrees of freedom are DFE(X) - DFE(X, Z). The F statistic for the test of the null hypothesis of no lack of fit is

$$F = \frac{(\mathrm{SSE}(X) - \mathrm{SSE}(X,Z))/(\mathrm{DFE}(X) - \mathrm{DFE}(X,Z))}{\mathrm{SSE}(X,Z)/\mathrm{DFE}(X,Z)}$$

Under the hypothesis of no lack of fit, the computed F has an F distribution with numerator and denominator degrees of freedom DFE(X) - DFE(X, Z) and DFE(X, Z), respectively. The p-value for the test is computed as the probability that a random variable with this distribution is greater than or equal to the computed F statistic.

The error degrees of freedom and error sum of squares from the fit of the expanded model are computed as the error degrees of freedom and sum of squares from the reduced model where *Z* and *y* have been adjusted for *X*. Routine RCOV is used to fit the reduced model. Let *e* be the vector of residuals from the original fitted model, let *W* be the diagonal matrix whose *i*-th diagonal element is the product of the weight and frequency for the *i*-th observation. The sum of squares and crossproducts matrix for the adjusted *Z* and *y* in the reduced model, which is input into RCOV, is

$$\begin{bmatrix} Z^T WZ - A^T A & Z^T We \\ & e^T We \end{bmatrix}$$

where *A* is a solution of $R^T A = DX^T W Z$.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2OFN/DR2OFN. The reference is:

   ```
   CALL R2OFN (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND, IRSP, FRQ, IWT, B, R, LDR,
       DFE, SSE, ICLUST, MAXIT, TOL, NGROUP, IGROUP, TESTLF, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Work array of length 3 * NOBS + |IIND| + NGROUP + 3 + max{*m* + 2.8854 * ln(*m*) + 2, 3 * NGROUP, NCOEF}, if ICLUST is positive. If ICLUST = 0, IWK can be an array of length 1.

   WK — Work array of length LWK.

2. Informational errors

Type	Code	Description
3	1	Convergence did not occur in the cluster analysis for the lack of fit test within MAXIT iterations. Better results may be obtained by increasing MAXIT.
4	2	An invalid weight or frequency is encountered. Weights and frequencies must be nonnegative.
3	3	The matrix of sum of squares and crossproducts computed for the within cluster model for testing lack of fit is not nonnegative definite within the tolerance defined by TOL.
4	4	At least one element in the columns containing the independent variables, IRSP, IFRQ, or IWT of X contains NaN (not a number), but the corresponding element in IGROUP is not zero. When ICLUST = 0, missing values in a row of X are indicated by setting the corresponding row of IGROUP to zero.

Examples

Example 1

This example uses data from Draper and Smith (1981, page 374), which is input in **X**. A multiple linear regression of column 6 of **X** on an intercept and columns 1, 3, and 4 is computed using routine **RGIVN**. Tests for lack of fit are computed for choices of **NGROUP** equal to 4 and 6 using routine **RLOFN**. Note that for **NGROUP** equal to 6 the results are exactly the same as for routine **RLOFE**. (If there are exact replicates in the data and the number of clusters used by **RLOFN** equals the number of distinct cases of the independent variables, then **RLOFN** and **RLOFE** produce the same output.)

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDB, LDR, LDSCPE, LDX, NCOEF, NCOL, NDEP, &
                 NIND, NOBS, J, INTCEP
      PARAMETER  (INTCEP=1, NCOL=6, NDEP=1, NIND=3, NOBS=20, &
                 LDSCPE=NDEP, LDX=NOBS, NCOEF=INTCEP+NIND, LDB=NCOEF, &
                 LDR=NCOEF)
!
      INTEGER    ICLUST, IDEP, IGROUP(NOBS), IIND, INDDEP(NDEP), &
      INDIND(NIND), IRSP, NGROUP, NOUT,  NRMISS, NROW
      REAL       B(LDB,NDEP), DFE, R(LDR,NCOEF), SCPE(LDSCPE,NDEP), &
                 SSE, TESTLF(10), X(LDX,NCOL)
!
      DATA (X(1,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 246.0/
      DATA (X(2,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(3,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 253.0/
      DATA (X(4,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 164.0/
      DATA (X(5,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 203.0/
      DATA (X(6,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 173.0/
      DATA (X(7,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 210.0/
      DATA (X(8,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(9,J),J=1,6)/0.0, 1.0, 0.0, 1.0, 0.0, 120.0/
      DATA (X(10,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 171.0/
      DATA (X(11,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 167.0/
      DATA (X(12,J),J=1,6)/0.0, 0.0, 1.0, 1.0, 0.0, 172.0/
      DATA (X(13,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(14,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(15,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 248.0/
      DATA (X(16,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 169.0/
      DATA (X(17,J),J=1,6)/0.0, 1.0, 0.0, 0.0, 0.0, 104.0/
      DATA (X(18,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 166.0/
      DATA (X(19,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 168.0/
      DATA (X(20,J),J=1,6)/0.0, 1.0, 1.0, 0.0, 0.0, 148.0/
      DATA INDIND/1, 3, 4/, INDDEP/6/
!
      NROW = NOBS
      IIND = NIND
      IDEP = NDEP
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, SCPE=SCPE)
      SSE    = SCPE(1,1)
      IRSP   = 6
      ICLUST = 2
```

```
      DO 10  NGROUP=4, 6, 2
         CALL RLOFN (X, IIND, INDIND, IRSP, B(1:, 1), R, DFE, SSE, NGROUP, &
                     IGROUP, TESTLF, ICLUST=ICLUST)
         CALL UMACH (2, NOUT)
         WRITE (NOUT,*) ' '
         WRITE (NOUT,*) 'NGROUP = ', NGROUP
         CALL WRIRN ('IGROUP', IGROUP, 1, NOBS, 1)
         WRITE (NOUT,*) ' '
         WRITE (NOUT,99999) '                    Test for Lack of '// &
                            'Fit'
         WRITE (NOUT,99999) '                         Sum of    Mean  '// &
                            '       Prob. of'
         WRITE (NOUT,99999) ' Source of Error   DF  Squares  Square  '// &
                            '    F  Larger F'
         WRITE (NOUT,99999) ' Lack of Fit    ', TESTLF(1), TESTLF(4), &
                            TESTLF(7), TESTLF(9), TESTLF(10)
         WRITE (NOUT,99999) ' Expanded model ', TESTLF(2), TESTLF(5), &
                            TESTLF(8)
         WRITE (NOUT,99999) ' Original model ', TESTLF(3), TESTLF(6)
   10 CONTINUE
99999 FORMAT (A, F5.1, F9.1, F8.2, F7.3, F10.3)
      END
```

Output

```
 NGROUP =   4

                                  IGROUP
 1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 4  4  4   4   2   4   2   4   2   4   4   4   4   4   4   4   1   4   4   3

                Test for Lack of Fit
                       Sum of    Mean         Prob. of
 Source of Error   DF  Squares  Square      F  Larger F
 Lack of Fit      1.0      0.4    0.38  0.035     0.855
 Expanded model  15.0    163.6   10.90
 Original model  16.0    163.9

 NGROUP =   6

                                     IGROUP
 1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 6  6  6   4   5   4   5   6   2   4   4   4   6   6   6   4   1   4   4   3

                   Test for Lack of Fit
                      Sum of    Mean         Prob. of
 Source of Error   DF  Squares  Square      F  Larger F
 Lack of Fit      2.0     20.5   10.25  1.001     0.393
 Expanded model  14.0    143.4   10.24
 Original model  16.0    163.9
```

Example 2

This example uses the same data and model from Example 1. Here, the option `ICLUST` = 0 is input so that the group numbers for performing the lack of fit test are input.

```
      USE IMSL_LIBRARIES
```

```
      IMPLICIT   NONE
      INTEGER    LDB, LDR, LDSCPE, LDX, NCOEF, NCOL, NDEP, &
                 NIND, NOBS, J, INTCEP
      PARAMETER  (INTCEP=1, NCOL=6, NDEP=1, NIND=3, NOBS=20, &
                 LDSCPE=NDEP, LDX=NOBS, NCOEF=INTCEP+NIND, LDB=NCOEF, &
                 LDR=NCOEF)
!
      INTEGER    ICLUST, IDEP, IGROUP(NOBS), IIND, &
                 INDDEP(NDEP), INDIND(NIND), IRSP, &
                 NGROUP, NOUT
      REAL       B(LDB,NDEP), DFE, R(LDR,NCOEF), SCPE(LDSCPE,NDEP), &
                 SSE, TESTLF(10), TOL, X(LDX,NCOL), &
                 XMAX(NCOEF), XMIN(NCOEF)
!
      DATA (X(1,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 246.0/
      DATA (X(2,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(3,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 253.0/
      DATA (X(4,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 164.0/
      DATA (X(5,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 203.0/
      DATA (X(6,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 173.0/
      DATA (X(7,J),J=1,6)/1.0, 1.0, 0.0, 0.0, 1.0, 210.0/
      DATA (X(8,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(9,J),J=1,6)/0.0, 1.0, 0.0, 1.0, 0.0, 120.0/
      DATA (X(10,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 171.0/
      DATA (X(11,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 167.0/
      DATA (X(12,J),J=1,6)/0.0, 0.0, 1.0, 1.0, 0.0, 172.0/
      DATA (X(13,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 247.0/
      DATA (X(14,J),J=1,6)/1.0, 1.0, 1.0, 0.0, 1.0, 252.0/
      DATA (X(15,J),J=1,6)/1.0, 0.0, 1.0, 0.0, 1.0, 248.0/
      DATA (X(16,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 169.0/
      DATA (X(17,J),J=1,6)/0.0, 1.0, 0.0, 0.0, 0.0, 104.0/
      DATA (X(18,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 166.0/
      DATA (X(19,J),J=1,6)/0.0, 1.0, 1.0, 1.0, 0.0, 168.0/
      DATA (X(20,J),J=1,6)/0.0, 1.0, 1.0, 0.0, 0.0, 148.0/
      DATA INDIND/1, 3, 4/, INDDEP/6/
      DATA IGROUP/4*4, 2, 4, 2, 4, 2, 7*4, 1, 2*4, 3/
!
      IIND = NIND
      IDEP = NDEP
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, SCPE=SCPE)
      SSE    = SCPE(1,1)
      IRSP   = 6
      ICLUST = 0
      NGROUP = 4
      CALL RLOFN (X, IIND, INDIND, IRSP, B(1:, 1), R, DFE, SSE, NGROUP,  &
                  IGROUP, TESTLF, iclust=iclust)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'NGROUP = ', NGROUP
      CALL WRIRN ('IGROUP', IGROUP, 1, NOBS, 1)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,99999) '                    Test for Lack of '// &
                         'Fit'
      WRITE (NOUT,99999) '                         Sum of    Mean  '// &
                         '       Prob. of'
      WRITE (NOUT,99999) ' Source of Error   DF  Squares  Square  '// &
                         '    F  Larger F'
      WRITE (NOUT,99999) ' Lack of Fit    ', TESTLF(1), TESTLF(4),&
                        TESTLF(7), TESTLF(9), TESTLF(10)
```

```
      WRITE (NOUT,99999) ' Expanded model ', TESTLF(2), TESTLF(5),&
                         TESTLF(8)
      WRITE (NOUT,99999) ' Original model ', TESTLF(3), TESTLF(6)
99999 FORMAT (A, F5.1, F9.1, F8.2, F7.3, F10.3)
      END
```

Output

```
 NGROUP =   4

                                        IGROUP
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
4  4  4   4   2   4   2   4   2   4   4   4   4   4   4   4   1   4   4   3

                    Test for Lack of Fit
                       Sum of    Mean          Prob. of
Source of Error   DF  Squares  Square      F  Larger F
Lack of Fit      1.0      0.4    0.38  0.035     0.855
Expanded model  15.0    163.6   10.90
Original model  16.0    163.9
```

RCASE

Computes case statistics and diagnostics given data points, coefficient estimates

$$\hat{\beta}$$

and the *R* matrix for a fitted general linear model.

Required Arguments

X — NRX by NCOL matrix containing the data. (Input)

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

B — Vector of length NCOEF containing a least-squares solution

$$\hat{\beta}$$

for the regression coefficients. (Input)

R — NCOEF by NCOEF upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements. Only the upper triangle of *R* is referenced.

DFE — Degrees of freedom for error. (Input)

SSE — Sum of squares for error. (Input)

CASE — NRX by 12 matrix containing the case statistics. (Output)
Columns 1 through 12 contain the following:

Col.	Description
1	Observed response
2	Predicted response
3	Residual
4	Leverage

5	Standardized residual
6	Jackknife residual
7	Cook's distance
8	DFFITS
9, 10	Confidence interval on the mean
11, 12	Prediction interval

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RCASE for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to RCASE will be made. Case statistics are computed for the data in X.
2	This is an intermediate or final invocation of RCASE. Case statistics are computed for the data in X.

NRX — Number of rows in X. (Input)
Default: NRX = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IEF — Effect option. (Input)
Default: IEF = -1.
The absolute value of IEF is the number of effects (sources of variation) due to the model. The sign of IEF specifies the following options.

IEF	Meaning
< 0	Each effect corresponds to a single regressor (coefficient) in the model. In this case, arguments NCLVAR, INDCL, NCLVAL, CLVAL, NVEF, INDEF, and IDUMMY are not referenced.
> 0	Each effect corresponds to one or more regressors. A general linear model is specified through the arguments NCLVAR, INDCL, NCLVAL, CLVAL, NVEF, INDEF, and IDUMMY.
0	There are no effects in the model. INTCEP must equal 1.

NCLVAR — Number of classification variables. (Input, if IEF is positive)

INDCL — Index vector of length NCLVAR containing the column numbers of X that are the classification variables. (Input, if IEF is positive)

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Input, if IEF is positive)
NCLVAL(I) is the number of distinct values for the I-th classification variable.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the values of the classification variables. (Input, if IEF is positive)
The first NCLVAL(1) variables contain the values of the first classification variable; the next NCLVAL(2) variables contain the values of the second classification variable; and so on. The last NCLVAL(NCLVAR) variables contain the values of the last classification variable.

NVEF — Vector of length IEF containing the number of variables associated with each effect in the model. (Input, if IEF is positive)

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(IEF). (Input, if IEF is positive)
The first NVEF(1) elements give the column numbers of X for each variable in the first effect; the next NVEF(2) elements give the column numbers for each variable in the second effect; and so on. The last NVEF(NEF) elements give the column numbers for each variable in the last effect.

IDUMMY — Dummy variable option. (Input, if IEF is positive)
Some indicator variables are defined for the I-th class variable as follows: Let J = NCLVAL(1) + NCLVAL(2) + … + NCLVAL(I - 1). NCLVAL(I) indicator variables are defined such that for K = 1, 2, …, NCLVAL(I) the K-th indicator variable for row M of X takes the value 1.0 if X(M, INDCL(I)) = CLVAL(J + K) and equals 0.0 otherwise. Dummy variables are generated from these indicator variables in one of the three following ways:

IDUMMY	Method
0, 1	The NCLVAL(I) indicator variables are the dummy variables (In RCASE, the computations for IDUMMY = 0 and IDUMMY = 1 are the same. The two values 0 and 1 are provided so that RCASE can be called after routine RGLM with no change in IDUMMY.)
2	The first NCLVAL(I) - 1 indicator variables are the dummy variables. The last indicator variable is omitted.
3	The K-th indicator variable minus the NCLVAL (I)-th indicator variable is the K-th dummy variable (K = 1, 2, …, NCLVAL(I) - 1).

IWT — Weighting option. (Input)

IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights, and the computed prediction interval uses SSE/(DFE * X(I, IWT)) for the estimated variance of a future response.

Default: IWT = 0.

IPRED — Prediction interval option. (Input)

IPRED = 0 means that prediction intervals are desired for a single future response. For positive IPRED, column number IPRED of X contains the number of future responses for which a prediction interval is desired on the average of the future responses.

Default: IPRED = 0.

CONPCM — Confidence level for two-sided interval estimates on the mean, in percent. (Input)

CONPCM percent confidence intervals are computed, hence, CONPCM must be greater than or equal to 0.0 and less than 100.0. CONPCM often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, where ONECL is greater than or equal to 50.0 and less than 100.0, set CONPCM = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPCM = 95.0.

CONPCP — Confidence level for two-sided prediction intervals, in percent. (Input)

CONPCP percent prediction intervals are computed, hence, CONPCP must be greater than or equal to 0.0 and less than 100.0. CONPCP often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, where ONECL is greater than or equal to 50.0 and less than 100.0, set CONPCP = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPCP = 95.0.

PRINT — Printing option. (Input)

Default: PRINT = 'N'.

PRINT is a character string indicating what is to be printed. The PRINT string is composed of one-character print codes to control printing. These print codes are given as follows:

PRINT(I:I)	Printing that Occurs
'A'	All
'N'	None
'1'	Observed response
'2'	Predicted response
'3'	Residual
'4'	Leverage
'5'	Standardized residual
'6'	Jackknife residual
'7'	Cook's distance
'8'	DFFITS
'M'	Confidence interval on the mean
'P'	Prediction interval
'X'	Influential cases (unusual "*x*-value")
'Y'	Outlier cases (unusual "*y*-value")

The concatenated print codes 'A', 'N', '1', ..., 'P' that comprise the PRINT string give the combination of statistics to be printed. Concatenation of these codes with print codes 'X' or 'Y' restricts printing to cases determined to be influential or outliers. Here are a few examples.

PRINT	Printing Action
'A'	All.
'N'	None.
'46'	Leverage and jackknife residual for all cases.
'AXY'	All statistics are printed for cases that are highly influential or are outliers.
'46XY'	Leverage and jackknife residual are printed for cases that are highly influential or are outliers.

IOBS — Number of the observation corresponding to the first row of X. (Input)
This observation number is used only for printing the row labels for the individual case statistics.
Default: IOBS = size (X,1).

NCOEF — Number of regression coefficients in the model. (Input)
Default: NCOEF = size (B,1).

LDR — Leading dimension of *R* exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of CASE containing NaN (not a number). (Output)
If any row of data contains NaN as a value of a variable other than the response variable, columns 3 through 12 of the corresponding row in CASE are set to NaN. If the response is missing, columns 1, 3, and 5 through 8 are set to NaN.

FORTRAN 90 Interface

Generic: CALL RCASE (X, IRSP, B, R, DFE, SSE, CASE [, …])
Specific: The specific interface names are S_RCASE and D_RCASE.

FORTRAN 77 Interface

Single: CALL RCASE (IDO, NRX, NCOL, X, LDX, INTCEP, IEF, NCLVAR, INDCL, NCLVAL, CLVAL, NVEF, INDEF, IDUMMY, IRSP, IWT, IPRED, CONPCM, CONPCP, PRINT, IOBS, NCOEF, B, R, LDR, DFE, SSE, CASE, LDCASE, NRMISS)
Double: The double precision name is DRCASE.

Description

The general linear model used by routine RCASE is

$$y = X\beta + \varepsilon$$

where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors, β is the $p \times 1$ vector of regression coefficients, and ε is the $n \times 1$ vector of errors whose elements are independently normally distributed with mean 0 and variance σ^2/w_i. The model used by RCASE also permits linear equality restrictions on β. From a general linear model fitted using the w_i's as the weights, routine RCASE computes confidence intervals and statistics for the individual cases that constitute the data set. Let x_i be a column vector containing elements of the i-th row of X. Let $W = \text{diag}(w_1, w_2, \ldots, w_n)$. The leverage is defined as

$$h_i = x_i^T \left(X^T W X \right)^{-} x_i w_i$$

(In the case of linear equality restrictions on β, the leverage is defined in terms of the reduced model.) Put $D = \text{diag}(d_1, d_2, \ldots, d_p)$ with $d_j = 1$ if the j-th diagonal element of R is positive and 0 otherwise. The leverage is computed as $h_i = (a^T D a) w_i$ where a is a solution to $R^T a = x_i$. The estimated variance of

$$\hat{y}_i = x_i^T \hat{\beta}$$

is given by $h_i s^2 / w_i$, where s^2 = **SSE**/**DFE**. The computation of the remainder of the case statistics follows easily from their definitions. See the Diagnostics for Individual Cases section in the chapter introduction for definitions of the case diagnostics.

Often predicted values and confidence intervals are desired for combinations of settings of the effect variables not used in computing the regression fit. This can be accomplished using a single data matrix by including these settings of the variables as part of the data matrix and by setting the response equal to NaN (not a number). NaN can be retrieved by the invocation of the function **AMACH**(6). The regression routine performing the fit will omit the case, and **RCASE** will compute a predicted value and confidence interval for the missing response from the given settings of the effect variables.

The type 3 informational errors can occur if the input variables *X*, *R*, *B* and SSE are not consistent with each other or if excessive rounding has occurred in their computation. The type 3 error message with error code 2 arises when *X* contains a row not in the space spanned by the rows of *R*. An examination of the model that was fitted and the *X* for which diagnostics are to be computed is required in order to insure that only linear combinations of the regression coefficients that can be estimated from the fitted model are specified in *X*. For further details, see the discussion of estimable functions given by Maindonald (1984, pages 166-168) and Searle (1971, pages 180-188).

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2ASE/DR2ASE**. The reference is:

   ```
   CALL R2ASE (IDO, NRX, NCOL, X, LDX, INTCEP, IEF, NCLVAR, INDCL, NCLVAL, CLVAL,
       NVEF, INDEF, IDUMMY, IRSP, IWT, IPRED, CONPCM, CONPCP, PRINT, IOBS, NCOEF, B,
       R, LDR, DFE, SSE, CASE, LDCASE, NRMISS, WK)
   ```

   The additional argument is:

   WK — Work vector of length **NCOEF** + 1.

2. Informational errors

Type	Code	Description
4	1	A weight is negative. Weights must be nonnegative.
3	2	The linear combination of the regression coefficients specified is not estimable within the preset tolerance.
3	3	A leverage much greater than 1.0 was computed. It is set to 1.0.
3	4	A deleted residual mean square much less than 0.0 was computed. It is set to 0.0.
4	5	A number of future observations for the prediction interval is nonpositive. It must be positive.

Examples

Example 1

A multiple linear regression model is fitted and case statistics computed for data discussed by Cook and Weisberg (1982, page 103). The fitted model is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

Some of the statistics in row 6 of the output matrix CASE are undefined (0.0/0.0) and are set to NaN (not a number). Some statistics in row 4 of CASE are set to Inf (positive machine infinity). The values of NaN and positive machine infinity can be retrieved by routine AMACH (or DMACH when using double precision), which is documented in the section "Machine-Dependent Constants" in Reference Material.

```
          USE RCASE_INT
          USE RGIVN_INT

          IMPLICIT   NONE
          INTEGER    INTCEP, LDB, LDCASE, LDR, LDSCPE, LDX, NCOEF, NCOL, &
                     NDEP, NIND, NROW, J, NRMISS
          PARAMETER  (INTCEP=1, NCOL=3, NDEP=1, NIND=2, NROW=7, &
                     LDCASE=NROW, LDSCPE=NDEP, LDX=NROW, &
                     NCOEF=INTCEP+NIND, LDB=NCOEF, LDR=NCOEF)
     !
          INTEGER    IDEP, IEF, IIND, INDDEP(1), INDIND(1), IOBS, IRSP
          REAL       B(LDB,NDEP), CASE(LDCASE,12), DFE, R(LDR,NCOEF), &
                     SCPE(LDSCPE,NDEP), SSE, X(LDX,NCOL)
          CHARACTER  PRINT*1
     !
          DATA (X(1,J),J=1,NIND+NDEP) /1.0, 1.0, 3.0/
          DATA (X(2,J),J=1,NIND+NDEP) /1.0, 2.0, 4.0/
          DATA (X(3,J),J=1,NIND+NDEP) /1.0, 3.0, 5.0/
          DATA (X(4,J),J=1,NIND+NDEP) /1.0, 4.0, 7.0/
          DATA (X(5,J),J=1,NIND+NDEP) /1.0, 5.0, 7.0/
          DATA (X(6,J),J=1,NIND+NDEP) /0.0, 6.0, 8.0/
          DATA (X(7,J),J=1,NIND+NDEP) /1.0, 7.0, 9.0/
     !
```

```
          IIND = -NIND
          IDEP = -NDEP
          CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, SCPE=SCPE)
          IEF    = -NIND
          IRSP   =  NCOL
          PRINT  = 'A'
          IOBS   = 1
          SSE    = SCPE(1,1)
          CALL RCASE (X, IRSP, B(1:,1), R, DFE, SSE, CASE, ief=ief, &
                PRINT=PRINT, iobs=iobs, ncoef=ncoef, nrmiss=nrmiss)
    !
          END
```

Output

```
                        * * * Case Analysis * * *
        Obs.    Observed  Predicted   Residual   Leverage  Std. Res.  Jack Res.
                Cook's D     DFFITS   95.0% CI   95.0% CI   95.0% PI   95.0% PI
           1     3.0000     3.1286    -0.1286     0.4714    -0.3886    -0.3430
                 0.0449    -0.3240     2.2609     3.9962     1.5957     4.6614
           2     4.0000     4.1429    -0.1429     0.2857    -0.3714    -0.3273
                 0.0184    -0.2070     3.4674     4.8183     2.7100     5.5757
           3     5.0000     5.1571    -0.1571     0.1857    -0.3826    -0.3376
                 0.0111    -0.1612     4.6126     5.7017     3.7812     6.5331
    Y      4     7.0000     6.1714     0.8286     0.1714     2.0000        Inf
                 0.2759        Inf     5.6482     6.6946     4.8038     7.5391
           5     7.0000     7.1857    -0.1857     0.2429    -0.4689    -0.4178
                 0.0235    -0.2366     6.5630     7.8084     5.7770     8.5945
    X      6     8.0000     8.0000     0.0000     1.0000        NaN        NaN
                    NaN        NaN     6.7364     9.2636     6.2129     9.7871
           7     9.0000     9.2143    -0.2143     0.6429    -0.7878    -0.7423
                 0.3724    -0.9959     8.2011    10.2275     7.5946    10.8339
```

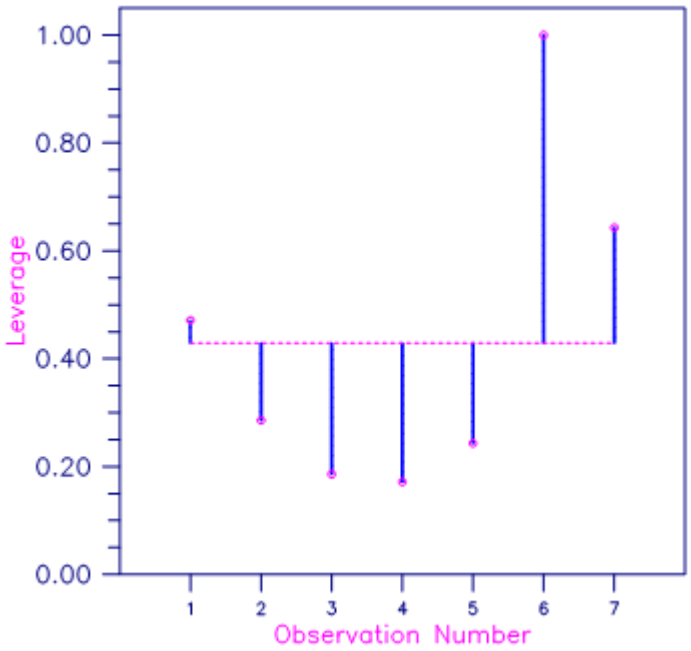


Figure 6, Plot of Leverages h_i and the Average (p/n = 3/7)

Example 2

A one-way analysis of covariance model is fitted to the turkey data discussed by Draper and Smith (1981, pages 243-249). The response variable is turkey weight *y* (in pounds). There are three groups of turkeys corresponding to the three states where they were reared. The age of a turkey (in weeks) is the covariate. The explanatory variables are group, age, and interaction. The model is

$$y_{ij} = \mu + \alpha_i + \beta x_{ij} + \beta_i x_{ij} + \varepsilon_{ij} \quad i = 1, 2, 3; \quad j = 1, 2, \ldots, n_i$$

where $\alpha_3 = 0$ and $\beta_3 = 0$. Routine RGLM is used to fit the model. The option **IDUMMY** = 2 is used. The fitted model gives three separate lines, one for each state where the turkeys were reared. Then, **RCASE** is used to compute case statistics from the fitted model.

```
      USE RCASE_INT
      USE RGLM_INT
      USE AMACH_INT
      INTEGER    IDEP, IEF, INTCEP, LDB, LDCASE, LDR, LDSCPE, LDX, &
                 MAXB, MAXCL, NCLVAR, NCOL, NROW
      PARAMETER  (IDEP=1, IEF=3, INTCEP=1, MAXB=6, MAXCL=3, NCLVAR=1, &
                 NCOL=3, NROW=13, LDB=MAXB, LDCASE=NROW, LDR=MAXB, &
                 LDSCPE=IDEP, LDX=NROW)
!
      INTEGER    IDUMMY, INDCL(NCLVAR), INDDEP(IDEP), &
                 INDEF(4), IOBS, IRANK, IRBEF(IEF+1), IRSP, &
                 NCLVAL(NCLVAR), NCOEF, NRMISS, NVEF(IEF)
      REAL       B(LDB,IDEP), CASE(LDCASE,12), CLVAL(MAXCL), &
                 DFE, R(LDR,MAXB),SCPE(LDSCPE,IDEP), SSE, X(LDX,NCOL)
      CHARACTER  PRINT
!
      DATA (X(1,J),J=1,3)  /25.0, 13.8, 3.0/
      DATA (X(2,J),J=1,3)  /28.0, 13.3, 1.0/
      DATA (X(3,J),J=1,3)  /20.0,  8.9, 1.0/
      DATA (X(4,J),J=1,3)  /32.0, 15.1, 1.0/
      DATA (X(5,J),J=1,3)  /22.0, 10.4, 1.0/
      DATA (X(6,J),J=1,3)  /29.0, 13.1, 2.0/
      DATA (X(7,J),J=1,3)  /27.0, 12.4, 2.0/
      DATA (X(8,J),J=1,3)  /28.0, 13.2, 2.0/
      DATA (X(9,J),J=1,3)  /26.0, 11.8, 2.0/
      DATA (X(10,J),J=1,3) /21.0, 11.5, 3.0/
      DATA (X(11,J),J=1,3) /27.0, 14.2, 3.0/
      DATA (X(12,J),J=1,3) /29.0, 15.4, 3.0/
      DATA (X(13,J),J=1,3) /23.0, 13.1, 3.0/
      DATA INDCL/3/, NVEF/1, 1, 2/, INDEF/3, 1, 1, 3/, INDDEP/2/
!
      IDUMMY = 2
      CALL RGLM (X, INDCL, NVEF, INDEF, IDEP, INDDEP, MAXCL, B, &
         IDUMMY=IDUMMY, NCLVAL=NCLVAL, CLVAL=CLVAL, IRBEF=IRBEF, &
         R=R, DFE=DFE, SCPE=SCPE)
!
      PRINT  = 'A'
```

```
      IRSP   = INDDEP(1)
      IPRED  = 0
      PRINT  = 'A'
      IOBS   = 1
      NCOEF  = IRBEF(IEF+1) - 1
      SSE    = SCPE(1,1)
      CALL RCASE (X, IRSP, B(1:, 1), R, DFE, SSE, CASE, IEF=IEF, &
         NCLVAR=NCLVAR, INDCL=INDCL, NCLVAL=NCLVAL, CLVAL=CLVAL, &
         NVEF=NVEF, INDEF=INDEF, IDUMMY=IDUMMY, IOBS=IOBS, &
      PRINT=PRINT, NCOEF=NCOEF)
!
      END
```

Output

```
                              * * * Case Analysis * * *
Obs.    Observed  Predicted   Residual   Leverage  Std. Res.  Jack Res.
          Cook's D     DFFITS   95.0% CI   95.0% CI   95.0% PI   95.0% PI
 1    13.8000    13.6000     0.2000     0.2000     0.7040     0.6762
        0.0207     0.3381    13.2641    13.9359    12.7773    14.4227
 2    13.3000    13.1901     0.1099     0.3187     0.4192     0.3930
        0.0137     0.2688    12.7661    13.6141    12.3276    14.0526
 3     8.9000     9.1418    -0.2418     0.5824    -1.1779    -1.2178
        0.3225    -1.4383     8.5686     9.7149     8.1970    10.0865
 4    15.1000    15.2143    -0.1143     0.7143    -0.6732    -0.6444
        0.1888    -1.0189    14.5795    15.8490    14.2309    16.1976
 5    10.4000    10.1538     0.2462     0.3846     0.9879     0.9860
        0.1017     0.7795     9.6881    10.6196     9.2701    11.0376
 6    13.1000    13.3300    -0.2300     0.7000    -1.3221    -1.4131
        0.6797    -2.1585    12.7016    13.9584    12.3507    14.3093
 7    12.4000    12.3900     0.0100     0.3000     0.0376     0.0348
        0.0001     0.0228    11.9786    12.8014    11.5337    13.2463
 8    13.2000    12.8600     0.3400     0.3000     1.2795     1.3533
        0.1169     0.8859    12.4486    13.2714    12.0037    13.7163
 9    11.8000    11.9200    -0.1200     0.7000    -0.6898    -0.6615
        0.1850    -1.0104    11.2916    12.5484    10.9407    12.8993
10    11.5000    11.8200    -0.3200     0.6000    -1.5930    -1.8472
        0.6344    -2.2623    11.2382    12.4018    10.8700    12.7700
11    14.2000    14.4900    -0.2900     0.3000    -1.0913    -1.1091
        0.0851    -0.7261    14.0786    14.9014    13.6337    15.3463
12    15.4000    15.3800     0.0200     0.6000     0.0996     0.0922
        0.0025     0.1130    14.7982    15.9618    14.4300    16.3300
13    13.1000    12.7100     0.3900     0.3000     1.4676     1.6330
        0.1538     1.0691    12.2986    13.1214    11.8537    13.5663
```

ROTIN

Computes diagnostics for detection of outliers and influential data points given residuals and the *R* matrix for a fitted general linear model.

Required Arguments

X — NRX by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)
The absolute value of IIND is the number of independent (explanatory) variables. The sign of IIND specifies the following options:

IIND	Meaning
< 0	The data for the -IIND independent variables are given in the first -IIND columns of X.
> 0	The data for the IIND independent variables are in the columns of X whose column numbers are given by the elements of INDIND.
= 0	There are no independent variables.

The regressors are the constant regressor (if INTCEP = 1) and the independent variables.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables. (Input, if IIND is positive)
If IIND is nonpositive, INDIND is not referenced and can be a vector of length one.

R — INTCEP + |IIND| by INTCEP + |IIND| upper triangular matrix containing the *R* matrix. (Input)
The *R* matrix can come from a regression fit based on a *QR* decomposition of the matrix of regressors or based on a Cholesky factorization R^TR of the matrix of sums of squares and crossproducts of the regressors. Elements to the right of a diagonal element of *R* that is zero must also be zero. A zero row indicates a nonfull rank model. For an *R* matrix that comes from a regression fit with linear equality restrictions on the parameters, each row of *R* corresponding to a restriction must have a corresponding diagonal element that is negative. The remaining rows of *R* must have positive diagonal elements.

DFE — Degrees of freedom for error. (Input)

SSE — Sum of squares for error. (Input)

E — Vector of length NRX with the residuals. (Input)
If a residual is not known, e.g., the value for the dependent (response) variable was missing, the input value of the corresponding element of E should equal NaN (not a number).

OTIN — NRX by 6 matrix containing diagnostics for detection of outliers and influential cases. (Output)
The columns of OTIN contain the following:

Col.	Description
1	Residual
2	Leverage (diagonal element of the 'Hat' matrix)
3	Standardized residual
4	Jackknife (deleted) residual
5	Cook's Distance
6	DFFITS

Optional Arguments

NRX — Number of rows of data. (Input)
Default: NRX = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

LDOTIN — Leading dimension of OTIN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDOTIN = size (OTIN,1).

NRMISS — Number of rows of **OTIN** containing NaN (not a number). (Output)
If any row of data contains NaN as a value of the independent variable or weight, elements in columns 2 thru 6 of the corresponding row in **OTIN** are set to NaN. If the residual is missing, elements in columns 3 thru 6 are set to NaN.

FORTRAN 90 Interface

Generic: **CALL ROTIN** (**X**, **IIND**, **INDIND**, **R**, **DFE**, **SSE**, **E**, **OTIN** [, ...])
Specific: The specific interface names are **S_ROTIN** and **D_ROTIN**.

FORTRAN 77 Interface

Single: **CALL ROTIN** (**NRX**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IWT**, **R**, **LDR**, **DFE**, **SSE**, **E**, **OTIN**, **LDOTIN**, **NRMISS**)
Double: The double precision name is **DROTIN**.

Description

The multiple regression model used by routine **ROTIN** is

$$y = X\,\beta + \varepsilon$$

where y is the $n \times 1$ vector of responses, X is the $n \times p$ matrix of regressors, β is the $p \times 1$ vector of regression coefficients, and ε is the $n \times 1$ vector of errors whose elements are independently normally distributed with mean 0 and variance σ^2/w_i. The model used by **ROTIN** also permits linear equality restrictions on β. From a multiple regression model fit using the w_i's as the weights, routine **ROTIN** computes diagnostics for outliers and influential cases. Let x_i be a column vector containing elements of the i-th row of X. Let $W = \text{diag}(w_1, w_2, \ldots, w_n)$. The leverage is defined as

$$h_i = x_i^T \left(X^T W X \right)^{-} x_i w_i$$

(In the case of linear equality restrictions on β, the leverage is defined in terms of the reduced model.) Put $D = \text{diag}(d_1, d_2, \ldots, d_p)$ with $d_j = 1$ if the j-th diagonal element of R is positive and 0 otherwise. The leverage is computed as $h_i = (a^T D a) w_i$ where a is a solution to $R^T a = x_i$. The computation of the remainder of the case diagnostics follows easily from their definitions. See the Diagnostics for Individual Cases section in the chapter introduction for definitions of the case diagnostics.

The type 3 informational errors can occur if the input variables X, R, E and SSE are not consistent with each other or if excessive rounding has occurred in their computation.The type 3 error message with error code 2 arises when X contains a row not in the space spanned by the rows of R. An examination of the model that was fitted and the X for which diagnostics are to be computed is required in order to insure that only linear combinations of

the regression coefficients that can be estimated from the fitted model are specified in *X*. For further details, see the discussion of estimable functions given by Maindonald (1984, pages 166-168) and Searle (1971, pages 180-188).

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2TIN/DR2TIN**. The reference is:

   ```
   CALL R2TIN (NRX, NCOL, X, LDX, INTCEP, IIND, INDIND, IWT, R, LDR, DFE, SSE, E, OTIN,
       LDOTIN, NRMISS, WK)
   ```

   The additional argument is:

   WK — Work vector of length **INTCEP** + |**IIND**|.

2. Informational errors

Type	Code	Description
3	2	The linear combination of the regression coefficients specified is not estimable within the preset tolerance.
3	3	A leverage much greater than 1.0 was computed. It is set to 1.0.
3	4	A deleted residual mean square much less than 0.0 was computed. It is set to 0.0.
4	1	A weight is negative. Weights must be nonnegative.

Examples

Example 1

A multiple linear regression model is fit and case statistics computed for data discussed by Cook and Weisberg (1982, page 103). The fitted model is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2$$

Some of the statistics in row 6 of the output matrix **OTIN** are undefined (0.0/0.0) and are set to NaN (not a number). Some statistics in row 4 of **OTIN** are infinite and are set to machine infinity. The values of NaN and machine infinity can be retrieved by routine **AMACH** (or **DMACH** when using double precision), which is documented in Reference Material.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    INTCEP, LDB, LDOTIN, LDR, LDSCPE, LDX, NCOEF, NCOL, &
                     NDEP, NIND, NROW, J
```

```
      PARAMETER  (INTCEP=1, NCOL=3, NDEP=1, NIND=2, NROW=7, &
                  LDOTIN=NROW, LDSCPE=NDEP, LDX=NROW, &
                  NCOEF=INTCEP+NIND, LDB=NCOEF, LDR=NCOEF)
!
      INTEGER    I, IDEP, IIND, INDDEP(1), INDIND(1), NOUT, NRMISS
      REAL       B(LDB,NDEP), D(NCOEF), DFE, E(NROW), &
                 OTIN(LDOTIN,6), R(LDR,NCOEF), SCPE(LDSCPE,NDEP), &
                 SSE, X(LDX,NCOL), XMAX(NCOEF), XMIN(NCOEF)
      CHARACTER  CLABEL(7)*10, RLABEL(1)*6
!
      DATA CLABEL/'Obs.', 'Residual', 'Leverage', 'Std. Res.', &
           'Jack. Res.', 'Cook''s D', 'DFFITS'/
      DATA RLABEL/'NUMBER'/
!
      DATA (X(1,J),J=1,NIND+NDEP) /1.0, 1.0, 3.0/
      DATA (X(2,J),J=1,NIND+NDEP) /1.0, 2.0, 4.0/
      DATA (X(3,J),J=1,NIND+NDEP) /1.0, 3.0, 5.0/
      DATA (X(4,J),J=1,NIND+NDEP) /1.0, 4.0, 7.0/
      DATA (X(5,J),J=1,NIND+NDEP) /1.0, 5.0, 7.0/
      DATA (X(6,J),J=1,NIND+NDEP) /0.0, 6.0, 8.0/
      DATA (X(7,J),J=1,NIND+NDEP) /1.0, 7.0, 9.0/
!
      IIND = -NIND
      IDEP = -NDEP
      CALL RGIVN (X, IIND, INDIND, IDEP, INDDEP, B, R=R, DFE=DFE, SCPE=SCPE)
      SSE = SCPE(1,1)
!                                  Compute residuals.
      DO 10  I=1, NROW
         E(I) = X(I,NCOL) - B(1,1) - SDOT(NIND, B((INTCEP+1): ,1), &
                1, X(I:, 1), LDX)
   10 CONTINUE
!
      CALL ROTIN (X, IIND, INDIND, R, DFE, SSE, E, OTIN,  &
                  NRMISS=NRMISS)
!
      CALL WRRRL ('OTIN', OTIN, RLABEL, CLABEL, FMT='(F10.3)')
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
!
      END
```

Output

```
                                    OTIN
Obs.   Residual    Leverage   Std. Res.  Jack. Res.    Cook's D      DFFITS
  1      -0.129       0.471      -0.389      -0.343       0.045      -0.324
  2      -0.143       0.286      -0.371      -0.327       0.018      -0.207
  3      -0.157       0.186      -0.383      -0.338       0.011      -0.161
  4       0.829       0.171       2.000         Inf       0.276         Inf
  5      -0.186       0.243      -0.469      -0.418       0.024      -0.237
  6       0.000       1.000         NaN         NaN         NaN         NaN
  7      -0.214       0.643      -0.788      -0.742       0.372      -0.996
NRMISS =   1
```

Example 2

In this example, routine **RNLIN** is first invoked to fit the following nonlinear regression model discussed by Neter, Wasserman, and Kutner (1983, pages 475-478):

$$y_i = \theta_1 e^{\theta_2 x_i} + \varepsilon_i \quad i = 1, 2, \ \ldots, 15$$

Then, **ROTIN** is used to compute case diagnostics. In addition, the leverage output by **ROTIN** is used to construct asymptotic confidence intervals on the mean of the nonlinear regression function evaluated at x_i. The asymptotic 95% confidence intervals are computed using the formula:

$$\hat{y}_i \mp t_{.975,DFE}\sqrt{s^2 h_i}$$

where h_i is the computed leverage, $t_{.975,DFE}$ is the 97.5 percentile of the t distribution with **DFE** degrees of freedom as computed by routine **TIN** (see Chapter 17, "Probability Distribution Functions and Inverses"), and s^2 equals **SSE**/**DFE**.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDOTIN, LDR, NOBS, NPARM, NRX
      PARAMETER  (NOBS=15, NPARM=2, NRX=1, LDOTIN=NRX, LDR=NPARM)
!
      INTEGER    IDERIV, IDUMMY(1), IEND, IOBS, IRANK, J, NOUT, NRMISS
      REAL       A, DE(NPARM, 1), DFE, E(1), FRQ, OTIN(LDOTIN,6), &
                 R(LDR,NPARM), SQRT, SSE, THETA(NPARM), WT, Y, &
                 YHAT
      INTRINSIC  SQRT
      EXTERNAL EXAMPL
!
      DATA THETA/60.0, -0.03/
!
      CALL UMACH (2, NOUT)
!
      IDERIV = 1
      CALL RNLIN (EXAMPL, THETA, R=R, DFE=DFE, SSE=SSE)
!
      WRITE (NOUT,*) ' Obs.  Pred.   Res.   Lev. St Res Del Res Cook '// &
                     'D DFFIT Conf Interval'
      DO 10  IOBS=1, NOBS
         CALL EXAMPL (NPARM, THETA, 0, IOBS, FRQ, WT, E, DE, IEND)
         CALL EXAMPL (NPARM, THETA, 1, IOBS, FRQ, WT, E, DE, IEND)
         CALL EXAMPL (NPARM, THETA, 2, IOBS, FRQ, WT, Y, DE, IEND)
         YHAT = Y - E(1)
         CALL ROTIN (DE, -NPARM, IDUMMY, R, DFE, SSE, E, OTIN, &
                     NRX=NRX, LDX=1, INTCEP=0)
         A = TIN(0.975,DFE)*SQRT((SSE/DFE)*OTIN(1,2))
         WRITE (NOUT,'(F5.1,10F7.2)') Y, YHAT, (OTIN(1,J),J=1,6), &
                                      YHAT - A, YHAT + A
   10 CONTINUE
      END
!
```

```
      SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, &
                         IEND)
      INTEGER    NPARM, IOPT, IOBS, IEND
      REAL       THETA(NPARM), FRQ, WT, E(1), DE(NPARM, 1)
!
      INTEGER    NOBS
      PARAMETER  (NOBS=15)
!
      REAL       EXP, XDATA(NOBS), YDATA(NOBS)
      INTRINSIC  EXP
!
      DATA YDATA/54.0, 50.0, 45.0, 37.0, 35.0, 25.0, 20.0, 16.0, 18.0, &
           13.0, 8.0, 11.0, 8.0, 4.0, 6.0/
      DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
           38.0, 45.0, 52.0, 53.0, 60.0, 65.0/

!
      IF (IOBS .LE. NOBS) THEN
         WT   = 1.0E0
         FRQ  = 1.0E0
         IEND = 0
         IF (IOPT .EQ. 0) THEN
            E(1) = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
         ELSE IF (IOPT .EQ. 1) THEN
            DE(1, 1) = -EXP(THETA(2)*XDATA(IOBS))
            DE(2, 1) = -THETA(1)*XDATA(IOBS)*EXP(THETA(2)*XDATA(IOBS))
         ELSE IF (IOPT .EQ. 2) THEN
            E(1) = YDATA(IOBS)
         END IF
      ELSE
         IEND = 1
      END IF
      RETURN
      END
```

Output

```
 Obs.  Pred.   Res.   Lev. St Res Del Res Cook D DFFIT Conf Interval
54.0  54.15  -0.14   0.40  -0.09  -0.09   0.00  -0.07  51.19  56.53
50.0  48.08   1.92   0.24   1.13   1.14   0.21   0.65  49.84  54.00
45.0  44.42   0.58   0.18   0.33   0.32   0.01   0.15  43.79  47.37
37.0  39.45  -2.45   0.13  -1.34  -1.39   0.13  -0.54  33.04  36.07
35.0  33.67   1.33   0.11   0.72   0.71   0.03   0.24  34.96  37.70
25.0  27.62  -2.63   0.11  -1.42  -1.49   0.12  -0.52  21.00  23.75
20.0  20.94  -0.94   0.12  -0.51  -0.50   0.02  -0.18  17.61  20.51
16.0  17.18  -1.18   0.12  -0.65  -0.63   0.03  -0.23  13.35  16.29
18.0  15.26   2.74   0.12   1.50   1.58   0.15   0.58  19.29  22.20
13.0  13.02  -0.02   0.11  -0.01  -0.01   0.00   0.00  11.56  14.40
 8.0   9.87  -1.87   0.10  -1.01  -1.01   0.06  -0.33   4.81   7.45
11.0   7.48   3.52   0.08   1.88   2.12   0.15   0.62  13.33  15.70
 8.0   7.19   0.81   0.08   0.43   0.42   0.01   0.12   7.64   9.97
 4.0   5.45  -1.45   0.06  -0.77  -0.75   0.02  -0.19   1.53   3.57
 6.0   4.47   1.53   0.05   0.80   0.79   0.02   0.18   6.61   8.45
```

GCLAS

Gets the unique values of each classification variable.

Required Arguments

X — NROW by NCOL matrix containing the data. (Input)

INDCL — Index vector of length NCLVAR containing the column numbers of X that are the classification variables. (Input)

MAXCL — An upper bound on the sum of the number of distinct values taken on by each classification variable. (Input)

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Output, if IDO = 0 or IDO = 1; input/output, if IDO = 2 or IDO = 3) NCLVAL(I) is the number of distinct values for the I-th classification variable.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the values of the classification variables. (Output, if IDO = 0 or IDO = 1; Input/Output, if IDO = 2 or IDO = 3) Since in general the length of CLVAL will not be known in advance, MAXCL (an upper bound for this length) should be used for purposes of dimensioning CLVAL. The first NCLVAL(1) variables contain the values of the first classification variable; the next NCLVAL(2) variables contain the values of the second classification variable; and so on. The last NCLVAL(NCLVAR) variables contain the values of the last classification variable. After invocation of GCLAS with IDO = 3, CLVAL contains the values sorted in ascending order by the classification variable.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of GCLAS for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to GCLAS will be made. Unique values for the classification variables are retrieved from X.

IDO	Action
2	This is an intermediate invocation of GCLAS. Unique values for the classification variables are retrieved from X.
3	This is the final invocation of GCLAS. Unique values for the classification variables are retrieved from X, and the values in CLVAL are sorted in ascending order for each classification variable.

NROW — Number of rows of data in X. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NCLVAR — Number of classification variables. (Input)
Default: NCLVAR = size (INDCL,1).

NMISS — Vector of length NCLVAR containing the number of elements of the data containing NaN for any classification variable. (Output, if IDO = 0 or IDO = 1; input/output if IDO = 2 or IDO = 3)

FORTRAN 90 Interface

Generic: CALL GCLAS (X, INDCL, MAXCL, NCLVAL, CLVAL [, ...])
Specific: The specific interface names are S_GCLAS and D_GCLAS.

FORTRAN 77 Interface

Single: CALL GCLAS (IDO, NROW, NCOL, X, LDX, NCLVAR, INDCL, MAXCL, NCLVAL, CLVAL, NMISS)
Double: The double precision name is DGCLAS.

Description

Routine GCLAS gets the unique values of *m* (Input in NCLVAR) classification variables. The routine can be used in conjunction with routine GRGLM. Routine GRGLM requires the values of the classification variables output by GCLAS in order to generate dummy variables for the general linear model.

In the input array **X**, missing values for a classification variable can be indicated by NaN (not a number). NAN is represented by **AMACH**(6). (See the section Machine-Dependent Constants in the *Reference Material* for a further discussion of **AMACH**, and missing values.) The nonmissing values of the classifications variables are output in **CLVAL**. If for a particular row of **X** a value of a classification variable is missing, nonmissing values of the other classification variables are still used. The number of elements equal to NaN for each classification variable is output in **NMISS**.

Comments

Informational Error

Type	Code	Description
4	1	MAXCL is too small. Increase MAXCL and the dimension of CLVAL.

Example

In the following example, the unique values of two classification variables are obtained from a data set **XX** with six rows. Here, routine **GCLAS** is invoked repeatedly with one row of the data set input into **X** at a time. Initially, **GCLAS** is invoked with **IDO** = 1, then with **IDO** = 2 for each of the six rows of data, and finally with **IDO** = 3.

```
      USE GCLAS_INT
      USE SCOPY_INT
      USE WRRRL_INT
      USE WRIRL_INT

      IMPLICIT   NONE
      INTEGER    LDX, LDXX, MAXCL, NCLVAR, NCOL, NOBS, J
      PARAMETER  (LDX=1, MAXCL=5, NCLVAR=2, NCOL=2, NOBS=6, LDXX=NOBS)
!
      INTEGER    I, IDO, INDCL(NCLVAR), NCLVAL(NCLVAR), NMISS(NCLVAR), &
                 NROW
      REAL       CLVAL(MAXCL), X(LDX,NCOL), XX(LDXX,NCOL)
      CHARACTER  CLABEL(2)*8, RLABEL(1)*17
!
      DATA INDCL/1, 2/, NCLVAL/2, 3/
      DATA (XX(1,J),J=1,NCOL)/10.0,  5.0/
      DATA (XX(2,J),J=1,NCOL)/20.0, 15.0/
      DATA (XX(3,J),J=1,NCOL)/20.0, 10.0/
      DATA (XX(4,J),J=1,NCOL)/10.0, 10.0/
      DATA (XX(5,J),J=1,NCOL)/10.0, 15.0/
      DATA (XX(6,J),J=1,NCOL)/20.0,  5.0/
!
      IDO = 1
      NROW = 0
      CALL GCLAS (X, INDCL, MAXCL, NCLVAL, CLVAL, IDO=IDO, NROW=NROW, &
                  NMISS=NMISS)
      IDO = 2
      NROW = 1
```

```
      DO 10  I=1, NOBS
         CALL SCOPY (NCOL, XX(I:,1), LDXX, X(1:,1), LDX)
         CALL GCLAS (X, INDCL, MAXCL, NCLVAL, CLVAL, IDO=IDO, NROW=NROW,  &
                     NMISS=NMISS)
   10 CONTINUE
      IDO = 3
      NROW = 0
      CALL GCLAS (X, INDCL, MAXCL, NCLVAL, CLVAL, IDO=IDO, &
                  NROW=NROW, NMISS=NMISS)
      I         = 1
      RLABEL(1) = 'Variable   CLVAL:'
      CLABEL(1) = 'None'
      DO 20  J=1, NCLVAR
         WRITE (RLABEL(1)(9:10),'(I2)') J
         CALL WRRRL (' ', CLVAL(I:), RLABEL, CLABEL, 1, NCLVAL(J), 1)
         I = I + NCLVAL(J)
   20 CONTINUE
      RLABEL(1) = 'NUMBER'
      CLABEL(1) = 'Variable'
      CLABEL(2) = 'NMISS'
      CALL WRIRL ('%/', NMISS, RLABEL, CLABEL)
      END
```

Output

```

 Variable 1 CLVAL:  10.00  20.00
 Variable 2 CLVAL:   5.00  10.00  15.00

 Variable  NMISS
        1      0
        2      0
```

GRGLM

Generates regressors for a general linear model.

Required Arguments

X — NROW by NCOL matrix containing the data. (Input)

INDCL — Index vector of length NCLVAR containing the column numbers of X that are the classification variables. (Input)

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Input)
NCLVAL(I) is the number of distinct values for the I-th classification variable.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the values of the classification variables. (Input)
The first NCLVAL(1) elements contain the values of the first classification variable; the next NCLVAL(2) elements contain the values of the second classification variable; and so on. The last NCLVAL(NCLVAR) elements contain the values of the last classification variable.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input)

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(NEF). (Input)
The first NVEF(1) elements give the column numbers of X for each variable in the first effect; the next NVEF(2) elements give the column numbers for each variable in the second effect; and so on. The last NVEF(NEF) elements give the column numbers for each variable in the last effect.

NREG — Number of columns in REG. (Output)

REG — NROW by NREG matrix containing the regressor variables generated from the matrix X. (Output, if IDUMMY > 0)
Since, in general, NREG will not be known in advance, the user may need to invoke GRGLM first with IDUMMY < 0, dimension REG, and then invoke GRGLM with IDUMMY > 0.

Optional Arguments

NROW — Number of rows of data in X. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NCLVAR — Number of classification variables. (Input)
Default: NCLVAR = size (INDCL,1).

NEF — Number of effects (sources of variation) in the model. (Input)
Default: NEF = size (NVEF,1).

IDUMMY — Dummy variable option. (Input)
Default: IDUMMY = 1.
Some indicator variables are defined for the I-th class variable as follows: Let J = NCLVAL(1) + NCLVAL(2) + … + NCLVAL(I - 1). NCLVAL(I) indicator variables are defined such that for K = 1, 2, …, NCLVAL(I) the K-th indicator variable for observation number IOBS takes the value 1.0 if X(IOBS, INDCL(I)) = CLVAL(J + K) and equals 0.0 otherwise. Dummy variables are generated from these indicator variables in one of the three following ways:

IDUMMY	Method
-1, 1	The NCLVAL(I) indicator variables are the dummy variables.
-2, 2	The first NCLVAL(I) - 1 indicator variables are the dummy variables. The last indicator variable is omitted.
-3, 3	The K-th indicator variable minus the NCLVAL(I)-th indicator variable is the K-th dummy variable (K = 1, 2, …, NCLVAL(I) - 1).

If IDUMMY < 0, only NREG is computed; and X, CLVAL, and REG are not referenced.

LDREG — Leading dimension of REG exactly as specified in the dimension statement in the calling program. (Input)
Default: LDREG = size (REG,1).

NRMISS — Number of rows of REG containing NaN (not a number). (Output)
A row of REG contains NaN for a regressor when any of the variables involved in generation of the regressor equals NaN or if a value of one of the classification variables in the model is not given by CLVAL.

FORTRAN 90 Interface

Generic: CALL GRGLM (X, INDCL, NCLVAL, CLVAL, NVEF, INDEF, NREG, REG [, …])
Specific: The specific interface names are S_GRGLM and D_GRGLM.

FORTRAN 77 Interface

Single: CALL GRGLM (NROW, NCOL, X, LDX, NCLVAR, INDCL, NCLVAL, CLVAL, NEF, NVEF, INDEF, IDUMMY, NREG, REG, LDREG, NRMISS)

Double: The double precision name is DGRGLM.

Description

Routine GRGLM generates regressors for a general linear model from a data matrix. The data matrix can contain classification variables as well as continuous variables.

Regressors for effects composed solely of continuous variables are generated as powers and crossproducts. Consider a data matrix containing continuous variables as columns 3 and 4. The effect indices (3,3) (stored in INDEF) generates a regressor whose *i*-th value is the square of the *i*-th value in column 3. The effect indices (3,4) generates a regressor whose *i*-th value is the product of the *i*-th value in column 3 with the *i*-th value in column 4.

Regressors for an effect (source of variation) composed of a single classification variable are generated using indicator variables. Let the classification variable *A* take on values a_1, a_2, ..., a_n (stored in CLVAL). From this classification variable, GRGLM creates *n* indicator variables. For *k* = 1, 2, ..., *n* we have

$$I_k = \begin{cases} 1 & \text{if } A = a_k \\ 0 & \text{otherwise} \end{cases}$$

For each classification variable, another set of variables is created from the indicator variables. We call these new variables *dummy variables*. Dummy variables are generated from the indicator variables in one of three manners:

1. the dummies are the n indicator variables
2. the dummies are the first *n* - 1 indicator variables
3. the *n* - 1 dummies are defined in terms of the indicator variables so that for balanced data, the usual summation restrictions are imposed on the regression coefficients

In particular, for IDUMMY = 1, the dummy variables are $A_k = I_k$ (*k* = 1, 2, ..., *n*). For IDUMMY = 2, the dummy variables are $A_k = I_k$ (*k* = 1, 2, ..., *n* - 1). For IDUMMY = 3, the dummy variables are $A_k = I_k - I_n$ (*k* = 1, 2, ..., *n* - 1). The regressors generated for an effect composed of a single classification variable are the associated dummy variables.

Let m_j be the number of dummies generated for the *j*-th classification variable. Suppose there are two classification variables *A* and *B* with dummies

$$A_1, A_2, \ldots, A_{m_1} \text{ and } B_1, B_2, \ldots, B_{m_2}$$

respectively. The regressors generated for an effect composed of two classification variables *A* and *B* are

$$A \otimes B = \left(A_1, A_2, \ldots, A_{m_1}\right)\left(B_1, B_2, \ldots, B_{m_2}\right)$$
$$= \left(A_1B_1, A_1B_2, \ldots, A_1B_{m_2}, A_2B_1, A_2B_2, \ldots, A_2B_{m_2}, A_{m_1}B_1, A_{m_1}B_2, \ldots, A_{m_1}B_{m_2}\right)$$

More generally, the regressors generated for an effect composed of several classification variables and several continuous variables are given by the Kronecker products of variables, where the order of the variables is specified in **INDEF**. Consider a data matrix containing classification variables in columns 1 and 2 and continuous variables in columns 3 and 4. Label these four columns A, B, X_1, and X_2. The regressors generated by the effect indices (1, 2, 3, 3, 4) is $A \otimes B \otimes X_1X_1X_2$.

Comments

Let the data matrix **X** = (A, B, X_1) where A and B are classification variables, and X_1 is a continuous variable. The model containing the effects A, B, AB, X_1, AX_1, BX_1 and ABX_1 is specified as follows:

NCLVAR = 2, **INDCL** = (1, 2), **NEF** = 7, **NVEF** = (1, 1, 2, 1, 2, 2, 3), and
INDEF = (1, 2, 1, 2, 3, 1, 3, 2, 3, 1, 2, 3).

For this model, suppose **NCLVAL**(1) = 2, **NCLVAL**(2) = 3, and **CLVAL**= (1.0, 2.0, 1.0, 2.0, 3.0). Let A_1, B_1, B_2, and B_3 be the associated indicator variables. Given below, for each **IDUMMY** option, are the regressors in their order of appearance in **REG**.

IDUMMY	REG
1	A_1, A_2, B_1, B_2, B_3, A_1B_1, A_1B_2, A_1B_3, A_2B_1, A_2B_2, A_2B_3, X_1, A_1X_1, A_2X_1, B_1X_1, B_2X_1, B_3X_1, $A_1B_1X_1$, $A_1B_2X_1$, $A_1B_3X_1$, $A_2B_1X_1$, $A_2B_2X_1$, $A_2B_3X_1$
2	A_1, B_1, B_2, A_1B_1, A_1B_2, X_1, A_1X_1, B_1X_1, B_2X_1, $A_1B_1X_1$, $A_1B_2X_1$
3	$A_1 - A_2$, $B_1 - B_3$, $B_2 - B_3$, $(A_1 - A_2)(B_1 - B_2)$, $(A_1 - A_2)(B_2 - B_3)$, X_1, $(A_1 - A_2)X_1$, $(B_1 - B_3)X_1$, $(B_2 - B_3)X_1$, $(A_1 - A_2)(B_1 - B_2)X_1$, $(A_1 - A_2)(B_2 - B_3)X_1$

Within a group of regressors corresponding to an interaction effect, the indicator variables composing the regressors vary most rapidly for the last classification variable, vary next most rapidly for the next to last classification variable, etc.

Example

In this example, regressors are generated for a two-way analysis-of-covariance model containing all the interaction terms. The model could be fitted by a subsequent invocation of routine **RGIVN** with **INTCEP** = 1. The regressors generated with the option **IDUMMY** = 2 are for the model whose mean function is

$$\mu + \alpha_i + \beta_j + \gamma_{ij} + \delta x_{ij} + \zeta_i x_{ij} + \eta_j x_{ij} + \theta_{ij} x_{ij} \qquad i = 1, 2; j = 1, 2, 3$$

where $\alpha_2 = \beta_3 = \gamma_{13} = \gamma_{21} = \gamma_{22} = \gamma_{23} = \zeta_2 = \eta_3 = \theta_{13} = \theta_{21} = \theta_{22} = \theta_{23} = 0$.

```
      USE GRGLM_INT
      USE UMACH_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    LDREG, LDX, LINDEF, MAXCL, NCLVAR, NCOL, NDREG, NEF, &
                 NROW
      PARAMETER  (LINDEF=12, MAXCL=5, NCLVAR=2, NCOL=3, NDREG=20, &
                 NEF=7, NROW=6, LDREG=NROW, LDX=NROW)
!
      INTEGER    IDUMMY, INDCL(NCLVAR), INDEF(LINDEF), J, &
                 NCLVAL(NCLVAR), NOUT, NREG, NRMISS, NVEF(NEF)
      REAL       CLVAL(MAXCL), REG(LDREG,NDREG), X(LDX,NCOL)
      CHARACTER  CLABEL(12)*7, RLABEL(1)*7
!
      DATA INDCL/1, 2/, NCLVAL/2, 3/, CLVAL/1.0, 2.0, 1.0, 2.0, 3.0/
      DATA NVEF/1, 1, 2, 1, 2, 2, 3/, INDEF/1, 2, 1, 2, 3, 1, 3, 2, 3, &
           1, 2, 3/
      DATA (X(1,J),J=1,NCOL)/1.0, 1.0, 1.11/
      DATA (X(2,J),J=1,NCOL)/1.0, 2.0, 2.22/
      DATA (X(3,J),J=1,NCOL)/1.0, 3.0, 3.33/
      DATA (X(4,J),J=1,NCOL)/2.0, 1.0, 4.44/
      DATA (X(5,J),J=1,NCOL)/2.0, 2.0, 5.55/
      DATA (X(6,J),J=1,NCOL)/2.0, 3.0, 6.66/
      DATA RLABEL/'NUMBER'/, CLABEL/' ', 'ALPHA1', 'BETA1', &
           'BETA2', 'GAMMA11', 'GAMMA12', 'DELTA', 'ZETA1', &
           'ETA1', 'ETA2', 'THETA11', 'THETA12'/
!
      IDUMMY = 2
      CALL GRGLM (X, INDCL, NCLVAL, CLVAL, NVEF, INDEF, NREG, REG, &
                  IDUMMY=IDUMMY, NRMISS=NRMISS)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NREG = ', NREG, '  NRMISS = ', NRMISS
      CALL WRRRL ('%/REG', REG, RLABEL, CLABEL, NROW, NREG, FMT='(F7.2)')
      END
```

Output

```
 NREG =   11  NRMISS =   0

                                    REG
     ALPHA1    BETA1    BETA2  GAMMA11  GAMMA12    DELTA    ZETA1     ETA1
1      1.00     1.00     0.00     1.00     0.00     1.11     1.11     1.11
2      1.00     0.00     1.00     0.00     1.00     2.22     2.22     0.00
3      1.00     0.00     0.00     0.00     0.00     3.33     3.33     0.00
4      0.00     1.00     0.00     0.00     0.00     4.44     0.00     4.44
5      0.00     0.00     1.00     0.00     0.00     5.55     0.00     0.00
6      0.00     0.00     0.00     0.00     0.00     6.66     0.00     0.00

       ETA2  THETA11  THETA12
1      0.00     1.11     0.00
2      2.22     0.00     2.22
3      0.00     0.00     0.00
4      0.00     0.00     0.00
```

```
     5      5.55      0.00      0.00
     6      0.00      0.00      0.00
```

RBEST

Selects the best multiple linear regression models.

Required Arguments

COV — NVAR by NVAR matrix containing the variance-covariance matrix or sum of squares and crossproducts matrix. (Input)
Only the upper triangle of COV is referenced. The last column of COV must correspond to the dependent variable.

NOBS — Number of observations. NOBS must be greater than or equal to the number of variables plus 1 (NVAR + 1), when using Adjusted R^2 or Mallows C_p criteria (ICRIT > 1). (Input)

ICOEFX — Index vector of length NTBEST + 1 containing the locations in COEF of the first row for each of the best regressions. (Output)
Here, NTBEST is the total number of best regressions found and is given as follows:

ICRIT	NTBEST
<0	-NBEST * ICRIT
1	NBEST * (NVAR - 1)
2	NBEST
3	NBEST

For I = 1, 2, ..., NTBEST, rows ICOEFX(I), ICOEFX(I) + 1, ..., ICOEFX(I + 1) - 1 of COEF correspond to the I-th regression.

COEF — ICOEFX(NTBEST + 1) - 1 by 5 matrix containing statistics relating to the regression coefficients of the best models. (Output)
An upper bound on the number of rows in COEF is given as follows:

ICRIT	Upper Bound on the Number of Rows in COEF
<0	-NBEST * ICRIT * (1 - ICRIT)/2
1	NBEST * (NVAR - 1) * NVAR/2
2	NBEST * (NVAR - 1)
3	NBEST * (NVAR - 1)

Each row corresponds to a coefficient for a particular regression. The regressions are in order of increasing subset size. Within each subset size, the regressions are ordered so that the better regressions appear first. The statistics in the columns are as follows:

Col.	Description
1	Variable number
2	Coefficient estimate
3	Estimated standard error of the estimate
4	*t*-statistic for the test that the coefficient is zero
5	*p*-value for the two-sided *t* test

(Inferences are conditional on the selected models.)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

ICRIT — Criterion option. (Input)
Default: ICRIT = 1.

ICRIT	Criterion	NSIZE
< 0	R^2	-ICRIT
1	R^2	NVAR - 1
2	Adjusted R^2	NVAR - 1
3	Mallows C_p	NVAR - 1

Subset sizes 1, 2, ..., NSIZE are examined.

NBEST — Number of best regressions to be found. (Input)
If the R^2 criterion is selected, the NBEST best regressions for each subset size examined are found. If the adjusted R^2 or Mallows C_p criterion is selected, the NBEST best overall regressions are found.
Default: NBEST = 1.

NGOOD — Maximum number of good regressions of each subset size to be saved in finding the best regressions. (Input)
NGOOD must be greater than or equal to NBEST. Normally, NGOOD should be less than or equal to 10. It need not ever be larger than the maximum number of subsets for any subset size. Computing time required is inversely related to NGOOD.
Default: NGOOD = 10.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

ICRITX — Index vector of length NSIZE + 1 containing the locations in CRIT of the first element for each subset size. (Output)
(See argument ICRIT for a definition of NSIZE.) For I = 1, 2, ..., NSIZE, element numbers ICRITX(I), ICRITX(I) + 1, ..., ICRITX(I + 1) - 1 of CRIT correspond to the I-th subset size.

CRIT — Vector of length max(ICRITX(NSIZE + 1) - 1, NVAR - 1) containing in its first ICRITX(NSIZE + 1) - 1 elements the criterion values for each subset considered, in increasing subset size order. (Output)
An upper bound on the length of CRIT is max(NGOOD * NSIZE, NVAR - 1). Within each subset size, results are returned in monotone order according to the criterion value with the results for the better regressions given first.

IVARX — Index vector of length NSIZE + 1 containing the locations in INDVAR of the first element for each subset size. (Output)
For I = 1,2, ..., NSIZE, element numbers IVARX(I), IVARX(I) + 1, ..., IVARX (I + 1) - 1 of INDVAR correspond to the I-th subset size.

INDVAR — Index vector of length IVARX(NSIZE + 1) - 1 containing the variable numbers for each subset considered and in the same order as in CRIT. (Output)
An upper bound on the length of INDVAR is NGOOD * NSIZE * (NSIZE + 1)/2.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

FORTRAN 90 Interface

Generic: CALL RBEST (COV, NOBS, ICOEFX, COEF [, ...])

Specific: The specific interface names are S_RBEST and D_RBEST.

FORTRAN 77 Interface

Single: CALL RBEST (NVAR, COV, LDCOV, NOBS, ICRIT, NBEST, NGOOD, IPRINT, ICRITX, CRIT, IVARX, INDVAR, ICOEFX, COEF, LDCOEF)

Double: The double precision name is DRBEST.

Description

Routine RBEST finds the best subset regressions for a regression problem with NVAR - 1 candidate independent variables. Typically, the intercept is forced into all models and is not a candidate variable. In this case, a sum of squares and crossproducts matrix for the independent and dependent variables corrected for the mean is input for COV. Routine CORVC in Chapter 3, "Correlation", can be used to compute the corrected sum of squares and crossproducts. IMSL routine RORDM in Chapter 19, "Utilities," can be used to reorder this matrix, if required. Other possibilities are

1. The intercept is not in the model. A raw (uncorrected) sum of squares and crossproducts matrix for the independent and dependent variables is required for COV. NOBS must be set to one greater than the number of observations. Routine MXTXF (IMSL MATH/LIBRARY) can be used to compute the raw sum of squares and crossproducts matrix.

2. An intercept is to be a candidate variable. A raw (uncorrected) sum of squares and crossproducts matrix for the constant regressor (= 1), independent, and dependent variables is required for COV. In this case, COV contains one additional row and column corresponding to the constant regressor. This row/column contains the sum of squares and crossproducts of the constant regressor with the independent and dependent variables. The remaining elements in COV are the same as in the previous case. NOBS must be set to one greater than the number of observations.

3. There are *m* variables to be forced into the models. A sum of squares and crossproducts matrix adjusted for the *m* variables is required. NOBS must be set to *m* less than the number of observations. Routine RCOV can be used to compute the adjusted sum of squares and crossproducts matrix. This is accomplished by a regression of the candidate variables on the variables to be forced into the models. The error sum of squares and crossproducts matrix, SCPE from RCOV, is the input to COV in routine RBEST.

"Best" is defined, on option, by one of three criteria:

1. R^2 (in percent)

$$R^2 = 100\left(1 - \frac{\text{SSE}_p}{\text{SST}}\right)$$

2. R_a^2 (adjusted R^2 in percent)

$$R_a^2 = 100\left[1 - \left(\frac{n-1}{n-p}\right)\frac{\text{SSE}_p}{\text{SST}}\right]$$

Note that maximizing this criterion is equivalent to minimizing the residual mean square, $\text{SSE}_p/(n - p)$.

3. Mallows' C_p statistic

$$C_p = \frac{SSE_p}{s^2_{\text{NVAR}-1}} + 2p - n$$

Here, n is **NOBS**, and **SST** is the total sum of squares. SSE_p is the error sum of squares in a model containing p regression parameters including β_0 (or p - 1 of the NVAR - 1 candidate variables).

$$s^2_{\text{NVAR}-1}$$

is the error mean square from the model with all NVAR - 1 candidate variables in the model. Hocking (1972) and Draper and Smith (1981, pages 296-302) discuss these criteria.

Routine **RBEST** is based on the algorithm of Furnival and Wilson (1974), this algorithm finds **NGOOD** candidate regressions for each possible subset size. These regressions are used to identify a set of best regressions. In large problems, many regressions are not computed. They may be rejected without computation based on results for other subsets, this yields an efficient technique for considering all possible regressions.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2EST/DR2EST**. The reference is:

```
CALL R2EST (NVAR, COV, LDCOV, NOBS, ICRIT, NBEST, NGOOD, IPRINT, ICRITX, CRIT,
    IVARX, INDVAR, ICOEFX, COEF, LDCOEF, WK, IWK)
```

The additional arguments are as follows:

WK — Work vector of length **NVAR** * (2 * **NGOOD** + 6) + (2 * **NVAR**^{3} + 4 * **NVAR**)/3. The first **NVAR** - 1 locations indicate which variables are in the full model. If **IWK**(I) = 0, then variable I is in the full model, otherwise, the variable has been dropped.

IWK — Integer work vector of length 3 * **NVAR**^{2} + 6 * **NVAR**.

2. Informational errors

Type	Code	Description
3	1	At least one variable is deleted from the full model because COV is singular.
4	3	No variables can enter any model.

Programming Notes

Routine **RBEST** can save considerable CPU time over explicitly computing all possible regressions. However, the routine has some limitations that can cause unexpected results for users that are unaware of the limitations of the software.

1. For **NVAR** > **-** $\log_2(\varepsilon)$ where ε is **AMACH**(4), (See the section "Machine-Dependent Constants" in the *Reference Material*), some results can be incorrect. This limitation arises because the possible models indicated by the model numbers 1, 2, ..., $2^{\mathbf{NVAR}-1}$, are stored as floating-point values, for sufficiently large **NVAR**, the model numbers cannot be stored exactly. On many computers, this means **S_RBEST** (for **NVAR** > 25) and **D_RBEST** (for **NVAR** > 50) can produce incorrect results.

2. Routine **RBEST** eliminates some subsets of candidate variables by obtaining lower bounds on the error sum of squares from fitting larger models. First, the full model containing all **NVAR** **-** 1 is fit sequentially using a forward stepwise procedure in which one variable enters the model at a time, and criterion values and model numbers for all the candidate variables that can enter at each step are stored. If linearly dependent variables are removed from the full model, error code 1 is issued. If this error is issued, some submodels that contain variables removed from the full model because of linear dependency can be overlooked, if they have not already been identified during the initial forward stepwise procedure. If error code 1 is issued and you want the variables that were removed from the full model to be considered in smaller models, you may want to rerun the program with a set of linearly independent variables.

Example

This example uses a data set from Draper and Smith (1981, pages 629**-**630). This data set is input to the matrix *X* by routine **GDATA** (see Chapter 19, "Utilities"). The first four columns contain the independent variables, and the last column contains the dependent variable. Routine **CORVC** in Chapter 3, "Correlation," is invoked to compute the corrected sum of squares and crossproducts matrix. Routine **RBEST** is then invoked to find the best regression for each of the four subset sizes using the R^2 criterion.

```
      USE RBEST_INT
      USE GDATA_INT
      USE CORVC_INT

      IMPLICIT   NONE
```

```
      INTEGER    LDCOEF, LDCOV, LDX, NBEST, NGOOD, NSIZE, NTBEST, NVAR
      PARAMETER  (LDX=13, NBEST=1, NGOOD=10, NVAR=5, &
                 LDCOEF=NBEST*(NVAR-1)*NVAR/2, LDCOV=NVAR, &
                 NSIZE=NVAR-1, NTBEST=NBEST*(NVAR-1))
!
      INTEGER    ICOEFX(NTBEST+1), ICOPT, ICRIT, ICRITX(NSIZE+1), &
                 INCD(1,1), INDVAR(NGOOD*NSIZE*(NSIZE+1)/2), &
                 IPRINT, IVARX(NSIZE+1), NMISS, NOBS, NROW, NVAR1
      REAL       COEF(LDCOEF,5), COV(LDCOV,NVAR), CRIT(NGOOD*NSIZE), &
                 SUMWT, X(LDX,NVAR), XMEAN(NVAR)
!
      CALL GDATA (5, X, NROW, NVAR1)
!
      ICOPT = 1
      CALL CORVC (NVAR, X, COV, ICOPT=ICOPT, NOBS=NOBS)
!
      IPRINT = 1
      CALL RBEST (COV, NOBS, ICOEFX, COEF, IPRINT=IPRINT)
!
      END
```

Output

```
 Regressions with   1 variable(s) (R-squared)
        Criterion         Variables
         67.5              4
         66.6              2
         53.4              1
         28.6              3

 Regressions with   2 variable(s) (R-squared)
        Criterion         Variables
         97.9              1  2
         97.2              1  4
         93.5              3  4
         68.0              2  4
         54.8              1  3

 Regressions with   3 variable(s) (R-squared)
        Criterion         Variables
         98.2              1  2  4
         98.2              1  2  3
         98.1              1  3  4
         97.3              2  3  4

 Regressions with   4 variable(s) (R-squared)
        Criterion         Variables
         98.2              1  2  3  4

          Best Regression with    1 variable(s) (R-squared)
    Variable  Coefficient  Standard Error  t-statistic  p-value
        4      -0.7382          0.1546       -4.775   0.0006
          Best Regression with    2 variable(s) (R-squared)
    Variable  Coefficient  Standard Error  t-statistic  p-value
        1        1.468          0.1213        12.10   0.0000
        2        0.662          0.0459        14.44   0.0000

          Best Regression with    3 variable(s) (R-squared)
```

Variable	Coefficient	Standard Error	t-statistic	p-value
1	1.452	0.1170	12.41	0.0000
2	0.416	0.1856	2.24	0.0517
4	-0.237	0.1733	-1.36	0.2054

Best Regression with 4 variable(s) (R-squared)

Variable	Coefficient	Standard Error	t-statistic	p-value
1	1.551	0.7448	2.083	0.0708
2	0.510	0.7238	0.705	0.5009
3	0.102	0.7547	0.135	0.8959
4	-0.144	0.7091	-0.203	0.8441

RSTEP

Builds multiple linear regression models using forward selection, backward selection, or stepwise selection.

Required Arguments

COV — NVAR by NVAR matrix containing the variance-covariance matrix or sum of squares and crossproducts matrix. (Input)
Only the upper triangle of COV is referenced.

NOBS — Number of observations. (Input)

AOV — Vector of length 13 containing statistics relating to the analysis of variance for the final model in this invocation. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

COEF — NVAR - 1 by 5 matrix containing statistics relating to the regression coefficients for the final model in this invocation. (Output)
The rows correspond to the NVAR - 1 variables with LEVEL(I) nonnegative, i.e., all variables but the

dependent variable. The rows are in the same order as the variables in COV except that the dependent variable is excluded. Each row corresponding to a variable not in the model is for the model supposing the additional variable was in the model.

Col.	Description
1	Coefficient estimate
2	Estimated standard error of the coefficient estimate
3	*t*-statistic for the test that the coefficient is zero
4	*p*-value for the two-sided *t* test
5	Variance inflation factor. The square of the multiple correlation coefficient for the I-th regressor after all others can be obtained from COEF(I, 5) by the formula 1.0 - 1.0/COEF(I, 5).

COVS — NVAR by NVAR matrix that results after COV has been swept on the columns corresponding to the variables in the model. (Output, if INVOKE = 0 or 1; Input/Output, if INVOKE = 2 or 3)
The estimated variance-covariance matrix of the estimated regression coefficients in the final model can be obtained by extracting the rows and columns of COVS corresponding to the independent variables in the final model and multiplying the elements of this matrix by AOV(8). If COV is not needed, COV and COVS can occupy the same storage locations.

Optional Arguments

INVOKE — Invocation option. (Input)
Default: INVOKE = 0.

INVOKE	Action
0	This is the only invocation of RSTEP for this variance-covariance matrix. Initialization, stepping, and wrap-up computations are performed.
1	This is the first invocation of RSTEP, and additional calls to RSTEP will be made. Initialization and stepping is performed.
2	This is an intermediate invocation of RSTEP and stepping is performed.
3	This is the final invocation of RSTEP and stepping is performed.

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LEVEL — Vector of length NVAR containing levels of priority for variables entering and leaving the regression. (Input)
LEVEL(I) = -1 means the I-th variable is the dependent variable. LEVEL(I) = 0 means the I-th variable is never to enter into the model. Other variables must be assigned a positive value to indicate their level of entry into the model. A variable can enter the model only after all variables with smaller nonzero levels of entry have entered. Similarly, a variable can only leave the model after all variables with higher levels of entry have left. Variables with the same level of entry compete for entry (deletion) at each step.

NFORCE — Variables with levels 1, 2, ..., NFORCE are forced into the model as the independent variables. (Input)
Default: NFORCE = 0.

NSTEP — Step length option. (Input)
For nonnegative NSTEP, NSTEP steps are taken. NSTEP = -1 means stepping continues until completion.
Default: NSTEP = -1.

ISTEP — Stepping option. (Input)
Default: ISTEP = -1.

ISTEP	Action
-1	An attempt is made to remove a variable from the model (backward step). A variable is removed if its *p*-value exceeds POUT. During initialization, all candidate independent variables enter the model.
1	An attempt is made to add a variable to the model (forward step). A variable is added if its *p*-value is less than PIN. During initialization, only the forced variables enter the model.
0	A backward step is attempted. If a variable is not removed, a forward step is attempted. This is a stepwise step. Only the forced variables enter the model during initialization.

PIN — Largest *p*-value for entering variables. (Input)
Variables with *p*-values less than PIN may enter the model. A common choice is PIN = 0.05.
Default: PIN = .05.

POUT — Smallest *p*-value for removing variables. (Input)
Variables with *p*-values greater than POUT may leave the model. POUT must be greater or equal to PIN. A common choice is POUT = 0.10 (or 2 * PIN).
Default: POUT = .10.

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100 * AMACH (4) is a common choice. See documentation for AMACH in the Reference Material.
Default: TOL = 1.e-5 for single precision and 2.d – 14 for double precision.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed on the final invocation.
2	Printing is performed after each step and on the final invocation.

SCALE — Vector of length NVAR containing the initial diagonal entries in COV. (Output, if INVOKE = 0 or 1; Input, if INVOKE = 2 or 3)

HIST — Vector of length NVAR containing the recent history of variables. (Output, if INVOKE = 0 or 1; Input/Output, otherwise)

HIST(I)	Meaning
$k > 0$	I-th variable was added to the model during the k-th step.
$k < 0$	I-th variable was deleted from the model during the k-th step.
0	I-th variable has never been in the model.
0.5	I-th variable was added into the model during initialization.

IEND — Completion indicator. (Output)

IEND	Meaning
0	Additional steps may be possible.
1	No additional steps are possible.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOVS — Leading dimension of COVS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVS = size (COVS,1).

FORTRAN 90 Interface

Generic: CALL RSTEP (COV, NOBS, AOV, COEF, COVS [, ...])

Specific: The specific interface names are S_RSTEP and D_RSTEP.

FORTRAN 77 Interface

Single: CALL RSTEP (INVOKE, NVAR, COV, LDCOV, LEVEL, NFORCE, NSTEP, ISTEP, NOBS, PIN, POUT, TOL, IPRINT, SCALE, HIST, IEND, AOV, COEF, LDCOEF, COVS, LDCOVS)

Double: The double precision name is DRSTEP.

Description

Routine RSTEP builds a multiple linear regression model using forward selection, backward selection, or forward stepwise (with a backward glance) selection. The routine RSTEP is designed so that the user can monitor, and perhaps change, the variables added (deleted) to (from) the model after each step. In this case, multiple calls to RSTEP (with INVOKE = 1, 2, 2, ..., 3) are made. Alternatively, RSTEP can be invoked once (with INVOKE = 0) in order to perform the stepping until a final model is selected.

Levels of priority can be assigned to the candidate independent variables. All variables with a priority level of 1 must enter the model before any variable with a priority level of 2. Similarly, variables with a level of 2 must enter before variables with a level of 3, etc.

Variables can also be forced into the model. If equal levels of priority are to be assumed, the levels of priority can all be set to 1.

Typically, the intercept is forced into all models and is not a candidate variable. In this case, a sum of squares and crossproducts matrix for the independent and dependent variables corrected for the mean is input for COV. Routine CORVC in Chapter 3, "Correlation" can be used to compute the corrected sum of squares and crossproducts. Routine RORDM in Chapter 19, "Utilities" can be used to reorder this matrix, if required. Other possibilities are

1. The intercept is not in the model. A raw (uncorrected) sum of squares and crossproducts matrix for the independent and dependent variables is required for COV. NOBS must be set to one greater than the number of observations. IMSL routine MXTXF (IMSL MATH/LIBRARY) can be used to compute the raw sum of squares and crossproducts matrix.

2. An intercept is to be a candidate variable. A raw (uncorrected) sum of squares and crossproducts matrix for the constant regressor (= 1), independent and dependent variables is required for COV. In this case, COV contains one additional row and column corresponding to the constant regressor. This row/column contains the sum of squares and crossproducts of the constant regressor with the independent and dependent variables. The remaining elements in COV are the same as in the previous case. NOBS must be set to one greater than the number of observations.

The stepwise regression algorithm is due to Efroymson (1960). Routine **RSTEP** uses sweeps of **COV** to move variables in and out of the model (Hemmerle 1967, Chapter 3). The SWEEP operator discussed by Goodnight (1979) is used. A description of the stepwise algorithm is given also by Kennedy and Gentle (1980, pages 335-340). The advantage of stepwise model building over all possible regressions (see routine RBEST) is that it is less demanding computationally when the number of candidate independent variables is very large. However, there is no guarantee that the model selected will be the best model (highest R^2) for any subset size of independent variables.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2TEP/DR2TEP**. The reference is:

   ```
   CALL R2TEP (INVOKE, NVAR, COV, LDCOV, LEVEL, NFORCE, NSTEP, ISTEP, NOBS, PIN,
       POUT, TOL, IPRINT, SCALE, HIST, IEND, AOV, COEF, LDCOEF, COVS, LDCOVS, SWEPT,
       IWK)
   ```

   The additional arguments are as follows:

   SWEPT — Work vector of length **NVAR** with information to indicate the independent variables in the model. (Output)
   SWEPT(**I**) = 1.0 indicates that independent variable **I** is in the model. Otherwise, **SWEPT**(**I**) = -1.0. Routine RSUBM can be called with the arguments **COVS** and **SWEPT** to obtain the part of **COVS** pertaining to the current model.

   IWK — Integer work vector of length 2 * **NVAR**.

2. Informational errors

Type	Code	Description
3	1	Based on TOL, there are linear dependencies among the variables to be forced.
4	2	No variables entered the model. Elements of AOV are set to NaN.

Examples

Example 1

Both examples use a data set from Draper and Smith (1981, pages 629-630). A corrected sum of squares and crossproducts matrix for this data is given in the **DATA** statement and can be computed using routine CORVC in Chapter 3, "Correlation". The first four columns are for the independent variables and the last column is for the dependent variable. Here, **RSTEP** is invoked using the backward stepping option.

```
      USE RSTEP_INT

      IMPLICIT   NONE
```

```
      INTEGER    LDCOEF, LDCOV, LDCOVS, NVAR
      PARAMETER  (NVAR=5, LDCOEF=NVAR, LDCOV=NVAR, LDCOVS=NVAR)
!
      INTEGER    IEND, IPRINT, LEVEL(NVAR), NOBS
      REAL       AOV(13), COEF(LDCOEF,5), COV(LDCOV,NVAR), &
                 COVS(LDCOVS,NVAR), HIST(NVAR), SCALE(NVAR)
!
      DATA COV/415.231, 251.077, -372.615, -290.000, 775.962, 251.077, &
           2905.69, -166.538, -3041.00, 2292.95, -372.615, -166.538, &
           492.308, 38.0000, -618.231, -290.000, -3041.00, 38.0000, &
           3362.00, -2481.70, 775.962, 2292.95, -618.231, -2481.70, &
           2715.76/
      DATA LEVEL/4*1, -1/
!
      NOBS   = 13
      IPRINT = 2
      CALL RSTEP (COV, NOBS, AOV, COEF, COVS, IPRINT=IPRINT)
!
      END
```

Output

```
 BACKWARD ELIMINATION
 STEP 0:  4 variable(s) entered.

 Dependent  R-squared   Adjusted  Est. Std. Dev.
 Variable   (percent)  R-squared  of Model Error
        5      98.238     97.356           2.446

               * * * Analysis of Variance * * *
                         Sum of        Mean             Prob. of
 Source           DF     Squares      Square  Overall F  Larger F
 Regression        4      2667.9       667.0    111.480    0.0000
 Error             8        47.9         6.0
 Total            12      2715.8

               * * * Inference on Coefficients * * *
                (Conditional on the Selected Model)
                Coef.    Standard                Prob. of    Variance
 Variable    Estimate       Error  t-statistic   Larger t   Inflation
        1       1.551      0.7448        2.082     0.0709        38.5
        2       0.510      0.7238        0.704     0.5012       254.4
        3       0.102      0.7547        0.135     0.8963        46.9
        4      -0.144      0.7091       -0.204     0.8437       282.5

 STEP 1 :  Variable 3 removed.
 Dependent  R-squared   Adjusted  Est. Std. Dev.
 Variable   (percent)  R-squared  of Model Error
        5      98.234     97.645           2.309

                  * * * Analysis of Variance * * *
                           Sum of        Mean             Prob. of
 Source           DF       Squares      Square  Overall F  Larger F
 Regression        3      2667.8       889.3    166.835    0.0000
 Error             9        48.0         5.3
 Total            12      2715.8

                   * * * Inference on Coefficients * * *
                    (Conditional on the Selected Model)
```

```
                Coef.    Standard                Prob. of    Variance
Variable    Estimate       Error  t-statistic   Larger t   Inflation
       1       1.452      0.1170       12.410     0.0000        1.07
       2       0.416      0.1856        2.242     0.0517       18.78
       4      -0.237      0.1733       -1.365     0.2054       18.94

          * * * Statistics for Variables Not in the Model * * *
                Coef.    Standard  t-statistic   Prob. of    Variance
Variable    Estimate       Error     to enter   Larger t   Inflation
       3       0.102      0.7547        0.135     0.8963       46.87

STEP 2 :  Variable 4 removed.

Dependent  R-squared   Adjusted  Est. Std. Dev.
Variable   (percent)  R-squared  of Model Error
       5      97.868     97.441           2.406

                 * * * Analysis of Variance * * *
                         Sum of        Mean             Prob. of
Source           DF     Squares      Square  Overall F  Larger F
Regression        2      2657.9      1328.9    229.502    0.0000
Error            10        57.9         5.8
Total            12      2715.8

                  * * * Inference on Coefficients * * *
                   (Conditional on the Selected Model)
                Coef.    Standard                Prob. of    Variance
Variable    Estimate       Error  t-statistic   Larger t   Inflation
       1       1.468      0.1213       12.105     0.0000        1.06
       2       0.662      0.0459       14.442     0.0000        1.06

          * * * Statistics for Variables Not in the Model * * *
                Coef.    Standard  t-statistic   Prob. of    Variance
Variable    Estimate       Error     to enter   Larger t   Inflation
       3       0.250      0.1847        1.354     0.2089        3.14
       4      -0.237      0.1733       -1.365     0.2054       18.94

* * * Backward Elimination Summary * * *
        Variable    Step Removed
               3               1
               4               2
```

Example 2

This example uses the data set in Example 1. Here, **RSTEP** is invoked using the forward stepwise option.

```
      USE RSTEP_INT

      IMPLICIT   NONE
      INTEGER    LDCOEF, LDCOV, LDCOVS, NVAR
      PARAMETER  (NVAR=5, LDCOEF=NVAR, LDCOV=NVAR, LDCOVS=NVAR)
!
      INTEGER    IEND, IPRINT, ISTEP, LEVEL(NVAR), NOBS
      REAL       AOV(13), COEF(LDCOEF,5), COV(LDCOV,NVAR), &
                 COVS(LDCOVS,NVAR), HIST(NVAR), SCALE(NVAR)
!
      DATA COV/415.231, 251.077, -372.615, -290.000, 775.962, 251.077, &
           2905.69, -166.538, -3041.00, 2292.95, -372.615, -166.538, &
```

```
            492.308, 38.0000, -618.231, -290.000, -3041.00, 38.0000, &
            3362.00, -2481.70, 775.962, 2292.95, -618.231, -2481.70, &
            2715.76/
      DATA LEVEL/4*1, -1/
!
      ISTEP  = 1
      NOBS   = 13
      IPRINT = 2
      CALL RSTEP (COV, NOBS, AOV, COEF, COVS, ISTEP=ISTEP, IPRINT=IPRINT)
!
      END
```

Output

```
 FORWARD SELECTION
 STEP 0:  No variables entered.

          * * * Statistics for Variables Not in the Model * * *
                 Coef.    Standard  t-statistic   Prob. of    Variance
   Variable    Estimate       Error     to enter   Larger t   Inflation
          1       1.869      0.5264        3.550     0.0046           1
          2       0.789      0.1684        4.686     0.0007           1
          3      -1.256      0.5984       -2.098     0.0598           1
          4      -0.738      0.1546       -4.775     0.0006           1

 STEP 1 :  Variable 4 entered.

 Dependent  R-squared   Adjusted  Est. Std. Dev.
 Variable   (percent)  R-squared  of Model Error
        5      67.454     64.496          8.964

                * * * Analysis of Variance * * *
                         Sum of        Mean             Prob. of
 Source           DF     Squares      Square  Overall F  Larger F
 Regression        1      1831.9      1831.9     22.799    0.0006
 Error            11       883.9        80.4
 Total            12      2715.8

                 * * * Inference on Coefficients * * *
                  (Conditional on the Selected Model)
               Coef.    Standard                Prob. of    Variance
 Variable    Estimate       Error  t-statistic   Larger t   Inflation
        4      -0.738      0.1546       -4.775     0.0006        1.00

          * * * Statistics for Variables Not in the Model * * *
               Coef.    Standard  t-statistic   Prob. of    Variance
 Variable    Estimate       Error     to enter   Larger t   Inflation
        1       1.440      0.1384       10.403     0.0000        1.06
        2       0.311      0.7486        0.415     0.6867       18.74
        3      -1.200      0.1890       -6.348     0.0001        1.00

    STEP 2 :  Variable 1 entered.

 Dependent  R-squared   Adjusted  Est. Std. Dev.
 Variable   (percent)  R-squared  of Model Error
        5      97.247     96.697          2.734

                * * * Analysis of Variance * * *
                         Sum of        Mean             Prob. of
```

```
    Source           DF     Squares      Square  Overall F  Larger F
    Regression        2      2641.0      1320.5    176.636    0.0000
    Error            10        74.8         7.5
    Total            12      2715.8

                     * * * Inference on Coefficients * * *
                      (Conditional on the Selected Model)
                  Coef.    Standard                Prob. of    Variance
    Variable    Estimate       Error  t-statistic   Larger t   Inflation
           1       1.440      0.1384       10.403     0.0000        1.06
           4      -0.614      0.0486      -12.622     0.0000        1.06

              * * * Statistics for Variables Not in the Model * * *
                  Coef.    Standard  t-statistic   Prob. of    Variance
    Variable    Estimate       Error     to enter   Larger t   Inflation
           2       0.416      0.1856        2.242     0.0517       18.78
           3      -0.410      0.1992       -2.058     0.0697        3.46

    * * * Forward Selection Summary * * *
          Variable    Step Entered
                 1             2
                 4             1
```

Example 3

For an extended version of Example 2 that in addition computes the intercept and standard error for the final model from RSTEP, see Example 2 for routine **RSUBM**.

GSWEP

Performs a generalized sweep of a row of a nonnegative definite matrix.

Required Arguments

KROW — Row/column number to be swept. (Input)

A — N by N nonnegative definite matrix whose row KROW is to be swept. (Input/Output)
Only the upper triangle of A is referenced.

Optional Arguments

N — Order of the matrix to be swept. (Input)
Default: N = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

IREV — Reversibility option. (Input)
Default: IREV = 0.

IREV	Action When Linear Dependence Is Declared
0	Elements of row and column KROW of A are set to 0.0. Reversibility of the generalized sweep operator is lost.
1	Elements of row and column KROW of A are left unchanged. Reversibility of the generalized sweep operator is maintained, but some post processing by the user is required. See Comments.

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100 * AMACH(4) is a common choice. See documentation for routine AMACH in the *Reference Material*.
Default: TOL = 1.e-5 for single precision and 2.d –14 for double precision.

SCALE — Vector of length N containing the diagonal scaling matrix used in the tolerance check. (Input)
A common choice for SCALE(I) is the I-th diagonal element of A before any calls to GSWEP have been made. If TOL = 0.0, SCALE is not referenced and can be a vector of length one.

SWEPT — Vector of length **N** with information to indicate what has and has not been swept. (Input/Output)
On the first call to **GSWEP** all elements must equal **-**1.0. On output, **SWEPT**(**KROW**) = 1.0 if the sweep was successful. If a linear dependence is declared, **SWEPT**(**KROW**) remains equal to **-**1.0.

FORTRAN 90 Interface

Generic: **CALL GSWEP** (**KROW**, **A** [, ...])
Specific: The specific interface names are **S_GSWEP** and **D_GSWEP**.

FORTRAN 77 Interface

Single: **CALL GSWEP** (**KROW**, **N**, **A**, **LDA**, **IREV**, **TOL**, **SCALE**, **SWEPT**)
Double: The double precision name is **DGSWEP**.

Description

Routine **GSWEP** computes an upper triangular generalized sweep of a nonnegative definite matrix. The versatility of the SWEEP operator for statistical computations, in particular for regression computations, is discussed by Goodnight (1979).

Routine **GSWEP** is based on UTG2SWEEP and RUTG2SWEEP described by Goodnight (1979, pages 157-158). (A misprint appears twice in "Step 5", page 157 of Goodnight's article. The "a_{ij}" should be replaced by "a_{ik}.") The test for linear dependence is the same as that given by Clarke (1982).

Comments

Say we wish to sweep *k* different rows of the matrix *A*. For purposes of discussion, let these be rows 1, 2, ..., *k* of *A*. Partition *A* into its first *k* rows and columns and the remainder,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

For a nonsingular A_{11}, successive invocations of **GSWEP** with *A* and **KROW** equal to 1, 2, ..., *k* yields

$$\begin{pmatrix} A_{11}^{-1} & A_{11}^{-1}A_{12} \\ - - - & A_{22} - A_{21}A_{11}^{-1}A_{12} \end{pmatrix}$$

Only the elements in the upper triangle of *A* are referenced. Thus, the elements in the lower triangles of the symmetric matrices

$$A_{11}^{-1} \text{ and } A_{22} - A_{21}A_{11}^{-1}A_{12}$$

are not returned. For a singular $A_{11}$and **IREV** equal to zero, a symmetric g_2 inverse of A_{11}, denoted by

$$A_{11}^{g_2}$$

is used. For a singular $A_{11}$and **IREV** not equal to zero, the first k rows of the swept A are not the same as for the **IREV** equal to one case. However,

$$G = A_{11}^{g_2} \text{ and } H = A_{11}^{g_2}A_{12}$$

can be obtained from the output A as follows:

$$g_{ij} = \begin{cases} 0 & \text{if } s_i + s_j \leq 0 \\ a_{ij} & \text{if } s_i + s_j = 2 \text{ and } i \leq j \\ a_{ji} & \text{if } s_i + s_j = 2 \text{ and } i > j \end{cases}$$

and

$$h_{ij} = \begin{cases} 0 & \text{if } i = j \text{ and } s_i = -1 \\ 1 & \text{if } i = j \text{ and } s_i = 1 \\ 0 & \text{if } i \neq j \text{ and } s_i + s_j \neq 0 \\ a_{ij} & \text{if } i \leq j \text{ and } s_i + s_j = 0 \\ -a_{ji} & \text{if } i > j \text{ and } s_i + s_j = 0 \end{cases}$$

H is the Hermite canonical form (also referred to as the Hermite normal form or a rowechelon form) of A_{11}.

Example

We consider the correlation matrix for the first three regressors from the example used by Berk (1976) and discussed by Frane (1977). The matrix is "nearly" singular. The rows of the correlation matrix are swept sequentially with **KROW** equal 1, 2, 3. With a tolerance of 0.001, the sweeps for 1 and 2 are successful. When a sweep on row 3 is attempted a linear dependence is declared. This is because

$$1 - R_{1 \cdot 2,3}^2 = 0.0001 < 0.001$$

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDA, N
      PARAMETER  (N=3, LDA=N)
```

```
      !
            INTEGER    ISETNG, KROW
            REAL       A(LDA,N), SCALE(N), SQRT, SWEPT(N), TOL
            INTRINSIC  SQRT
      !
            A(1,1) = 1.0
            A(1,2) = SQRT(0.99)
            A(1,3) = 0.1*SQRT(0.99)
            A(2,2) = 1.0
            A(2,3) = 0.0
            A(3,3) = 1.0
            TOL    = 0.001
      !                                   Copy diagonal of A to SCALE.
            CALL SCOPY (N, A(1:,1), LDA+1, SCALE, 1)
      !                                   Initialize elements of SWEPT to -1.
            SWEPT = -1.0
            ISETNG = 4
            CALL WROPT (-6, ISETNG, 1)
            CALL WRRRN ('A', A, ITRING=1)
            CALL WRRRN ('SWEPT', SWEPT)
            SWEPT = -1.0
            DO 10  KROW=1, 3
               CALL GSWEP (KROW, A, tol=tol, scale=scale, swept=swept)
               CALL WRRRN ('A', A, ITRING=1)
               CALL WRRRN ('SWEPT', SWEPT)
         10 CONTINUE
            END
```

Output

```
                  A
             1         2         3
1   1.00000   0.99499   0.09950
2             1.00000   0.00000
3                       1.00000

   SWEPT
1  -1.00000
2  -1.00000
3  -1.00000

                  A
             1         2         3
1   1.00000   0.99499   0.09950
2             0.01000  -0.09900
3                       0.99010

   SWEPT
1   1.00000
2  -1.00000
3  -1.00000

                  A
             1         2         3
1   100.000   -99.499     9.950
2             100.000    -9.900
3                         0.010
```

```
       SWEPT
   1   1.00000
   2   1.00000
   3  -1.00000


                  A
             1         2         3
   1   100.000   -99.499     0.000
   2             100.000     0.000
   3                         0.000

       SWEPT
   1   1.00000
   2   1.00000
   3  -1.00000
```

RSUBM

Retrieves a symmetric submatrix from a symmetric matrix.

Required Arguments

A — NA by NA symmetric matrix. (Input)
Only the upper triangle of A is referenced.

SWEPT — Vector of length NA. (Input)
Element A(I, J) is included in submatrix ASUB if and only if SWEPT(I) > 0.0 and SWEPT(J) > 0.0.

NASUB — Order of submatrix ASUB. (Output)
NASUB equals the number of elements in SWEPT that are greater than zero.

ASUB — NASUB by NASUB symmetric matrix containing a submatrix of A. (Output)
If A is not needed, ASUB and A can share the same storage locations.

Optional Arguments

NA — Order of matrix A. (Input)
Default: NA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

LDASUB — Leading dimension of ASUB exactly as specified in the dimension statement of the calling program. (Input)
Default: LDASUB = size (ASUB,1).

FORTRAN 90 Interface

Generic: CALL RSUBM (A, SWEPT, NASUB, ASUB [, ...])
Specific: The specific interface names are S_RSUBM and D_RSUBM.

FORTRAN 77 Interface

Single: CALL RSUBM (NA, A, LDA, SWEPT, NASUB, ASUB, LDASUB)

Double: The double precision name is **DRSUBM**.

Description

Routine **RSUBM** retrieves a symmetric submatrix from a symmetric matrix *A*. If elements *i* and *j* of the input vector **SWEPT** are greater than zero, then the *ij*-th element of **A** is output in the submatrix **ASUB**. Otherwise, the *ij*-th element of *A* will not be included in **ASUB**. (Here, *i* = 1, 2, ..., **NA**, and *j* = 1, 2, ..., **NA**, where **NA** is the order of **A**.)

Routine **RSUBM** can be useful in conjunction with two routines, **GSWEP** and **RSTEP**. The routine **RSUBM** can be used after routine **GSWEP** in order to retrieve the submatrix of *A* that corresponds to the rows/columns that have been successfully swept. In this case, the **SWEPT** vector output from **GSWEP** can be used as the input for the argument **SWEPT** in **RSUBM**. Also, **RSUBM** can be used after routine **RSTEP** in order to retrieve the submatrix of **COVS** that corresponds to the independent variables in the final model. In this case, the **HIST** vector output from **RSTEP** can be used as the input for the argument **SWEPT** in **RSUBM**.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2UBM/DR2UBM**. The reference is:

   CALL R2UBM (**NA**, **A**, **LDA**, **SWEPT**, **NASUB**, **ASUB**, **LDASUB**, **IWK**)

   The additional argument is:

   IWK — Vector of length **NASUB**.

2. Routine **RSUBM** can be used after invoking routines **GSWEP** and **RSTEP** in order to retrieve the submatrix for the variables in the model.

Examples

Example 1

The 2 × 2 symmetric submatrix **ASUB** is retrieved from rows and columns 1 and 4 of the 4 × 4 symmetric matrix **A**.

```
      USE RSUBM_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDASUB, NA
      PARAMETER  (LDASUB=2, NA=4, LDA=NA)
!
      INTEGER    NASUB
      REAL       A(LDA,NA), ASUB(LDASUB,LDASUB), SWEPT(NA)
!
```

```
      DATA SWEPT/1.0, -1.0, -1.0, 1.0/
      DATA A/10.0, 20.0, 40.0, 70.0, 20.0, 30.0, 50.0, 80.0, 40.0,&
          50.0, 60.0, 90.0, 70.0, 80.0, 90.0, 100.0/
!
      CALL RSUBM (A, SWEPT, NASUB, ASUB)
      CALL WRRRN ('ASUB', ASUB)
      END
```

Output

```
      ASUB
        1       2
1    10.0    70.0
2    70.0   100.0
```

Example 2

This example invokes **RSUBM** after routine **RSTEP** in order to retrieve the submatrix of **COVS** that corresponds to the independent variables in the final stepwise model. With this submatrix, routine **BLINF** (IMSL MATH/LIBRARY) is used to compute the estimated standard deviation for the intercept in the final model.

A data set from Draper and Smith (1981, pages 629-630) is used. The means and the corrected sum of squares and crossproducts matrix for this data are given in the **DATA** statements. They can be computed using routine **CORVC** in Chapter 3, "Correlation". The first four entries in **XMEAN** and the first four columns of **COV** correspond to the independent variables, the last entry in **XMEAN** and the last column of **COV** correspond to the dependent variable.

After **RSTEP** is invoked to obtain a model, the intercept is computed using the formula

$$\hat{\beta}_0 = \bar{y} - \sum_{i=1}^{k} \hat{\beta}_i \bar{x}_i$$

where *k* is the number of independent variables in the final model. The estimated standard deviation of the intercept is computed using the formula

$$\text{Est. St. Dev}\left(\hat{\beta}_0\right) = \sqrt{s^2\left(1/n + \bar{x}^T A \bar{x}\right)}$$

where s^2 is the error mean square from the fit (stored in **AOV**(8)), *n* is the number of observations, $\bar{x}$ is the subvector of means for the independent variables in the final model (in this case the first mean and the fourth mean), and *A* is the submatrix (in this case with rows and columns 1 and 4) of the matrix **COVS** that is output by **RSTEP**.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCOEF, LDCOV, LDCOVS, NVAR
      PARAMETER  (NVAR=5, LDCOEF=NVAR, LDCOV=NVAR, LDCOVS=NVAR)
!
```

```
      INTEGER    I, IEND, INVOKE, IPRINT, ISTEP, J, LEVEL(NVAR), &
                 NFORCE, NIND, NOBS, NOUT, NSTEP
      REAL       AOV(13), B0, COEF(LDCOEF,5), &
                 COV(LDCOV,NVAR), COVS(LDCOVS,NVAR), HIST(NVAR), PIN, &
                 POUT, SCALE(NVAR), SEB0, SQRT, TOL, XMEAN(NVAR)
      INTRINSIC  SQRT
!
      DATA COV/415.231, 251.077, -372.615, -290.000, 775.962, 251.077, &
           2905.69, -166.538, -3041.00, 2292.95, -372.615, -166.538, &
           492.308, 38.0000, -618.231, -290.000, -3041.00, 38.0000, &
           3362.00, -2481.70, 775.962, 2292.95, -618.231, -2481.70, &
           2715.76/
      DATA XMEAN/7.46154, 48.1538, 11.7692, 30.0000, 95.4231/
      DATA LEVEL/4*1, -1/
!
      J = 0
      ISTEP  = 1
      NOBS   = 13
      IPRINT = 1
      CALL RSTEP (COV, NOBS, AOV, COEF, COVS, NVAR=NVAR, ISTEP=ISTEP, &
      IPRINT=IPRINT, HIST=HIST)
!                                Compute intercept
      B0 = XMEAN(NVAR)
      DO 10  I=1, NVAR - 1
         IF (HIST(I) .GT. 0.0) THEN
            B0       = B0 - XMEAN(I)*COEF(I,1)
            J        = J + 1
            XMEAN(J) = XMEAN(I)
         END IF
   10 CONTINUE
!                                Compute standard error of intercept
      CALL RSUBM (COVS, HIST, NIND, COVS)
      SEB0 = 1.0/NOBS + BLINF(COVS, XMEAN, XMEAN, NRA=NIND, NCA=NIND)
      SEB0 = SQRT(AOV(8)*SEB0)
!                                Print intercept and standard error
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) ' '
      WRITE (NOUT,99999) 'Intercept ', B0
      WRITE (NOUT,99999) 'Std. Error', SEB0
99999 FORMAT (1X, A, F10.3)
!
      END
```

Output

```
FORWARD SELECTION

Dependent  R-squared   Adjusted  Est. Std. Dev.
Variable   (percent)  R-squared  of Model Error
       5      97.247     96.697          2.734

                 * * * Analysis of Variance * * *
                         Sum of        Mean             Prob. of
Source           DF     Squares      Square  Overall F  Larger F
Regression        2      2641.0      1320.5    176.636    0.0000
Error            10        74.8         7.5
Total            12      2715.8

                * * * Inference on Coefficients * * *
```

(Conditional on the Selected Model)

Variable	Coef. Estimate	Standard Error	t-statistic	Prob. of Larger t	Variance Inflation
1	1.440	0.1384	10.403	0.0000	1.06
4	-0.614	0.0486	-12.622	0.0000	1.06

* * * Statistics for Variables Not in the Model * * *

Variable	Coef. Estimate	Standard Error	t-statistic to enter	Prob. of Larger t	Variance Inflation
2	0.416	0.1856	2.242	0.0517	18.7
3	-0.410	0.1992	-2.058	0.0697	3.46

* * * Forward Selection Summary * * *

Variable	Step Entered
1	2
4	1

Intercept	103.097
Std. Error	2.124

RCURV

Fits a polynomial curve using least squares.

Required Arguments

XDATA — Vector of length NOBS containing the *x* values. (Input)

YDATA — Vector of length NOBS containing the *y* values. (Input)

B — Vector of length NDEG + 1 containing the coefficients

$$\hat{\beta}$$

(Output)
The fitted polynomial is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \ \ldots \ + \hat{\beta}_k x^k$$

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (XDATA,1).

NDEG — Degree of polynomial. (Input)
Default: NDEG = size (B,1) – 1.

SSPOLY — Vector of length NDEG + 1 containing the sequential sums of squares. (Output)
SSPOLY(1) contains the sum of squares due to the mean. For *i* = 1, 2, …, NDEG, SSPOLY(*i* + 1) contains the sum of squares due to x^i adjusted for the mean, *x*, x^2, …, and x^{i-1}.

STAT — Vector of length 10 containing statistics described below. (Output)

i	Statistics
1	Mean of *x*
2	Mean of *y*
3	Sample variance of *x*
4	Sample variance of *y*
5	*R*-squared (in percent)
6	Degrees of freedom for regression
7	Regression sum of squares
8	Degrees of freedom for error
9	Error sum of squares
10	Number of data points (*x*, *y*) containing NaN (not a number) as a *x* or *y* value

FORTRAN 90 Interface

Generic: CALL RCURV (XDATA, YDATA, B [, ...])

Specific: The specific interface names are S_RCURV and D_RCURV.

FORTRAN 77 Interface

Single: CALL RCURV (NOBS, XDATA, YDATA, NDEG, B, SSPOLY, STAT)

Double: The double precision name is DRCURV.

Description

Routine RCURV computes estimates of the regression coefficients in a polynomial (curvilinear) regression model. In addition to the computation of the fit, RCURV computes some summary statistics. Sequential sums of squares attributable to each power of the independent variable (stored in SSPOLY) are computed. These are useful in assessing the importance of the higher order powers in the fit. Draper and Smith (1981, pages 101-102) and Neter and Wasserman (1974, pages 278-287) discuss the interpretation of the sequential sums of squares. The statistic R^2 (stored in STAT(5)) is the percentage of the sum of squares of *y* about its mean explained by the polynomial curve. Specifically,

$$R^2 = \frac{\sum_{i=1}^{n}\left(\hat{y}_i - \overline{y}\right)^2}{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)^2} 100\%$$

where

$$\hat{y}_i$$

is the fitted y value at x_i and

$$\overline{y}$$

(stored in **STAT**(2)) is the mean of y. This statistic is useful in assessing the overall fit of the curve to the data. R^2 must be between 0% and 100%, inclusive. R^2 = 100% indicates a perfect fit to the data.

Routine **RCURV** computes estimates of the regression coefficients in a polynomial model using orthogonal polynomials as the regressor variables. This reparameterization of the polynomial model in terms of orthogonal polynomials has the advantage that the loss of accuracy resulting from forming powers of the x-values is avoided. All results are returned to the user for the original model.

The routine **RCURV** is based on the algorithm of Forsythe (1957). A modification to Forsythe's algorithm suggested by Shampine (1975) is used for computing the polynomial coefficients. A discussion of Forsythe's algorithm and Shampine's modification appears in Kennedy and Gentle (1980, pages 342-347).

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2URV/DR2URV**. The reference is:

   ```
   CALL R2URV (NOBS, XDATA, YDATA, NDEG, B, SSPOLY, STAT, WK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 11 * NOBS + 11 * NDEG + 5 + (NDEG + 1) * (NDEG + 3).

   IWK — Work vector of length NOBS.

2. Informational errors

Type	Code	Description
4	3	Each (*x*, *y*) point contains NaN (not a number). There are no valid data.
4	7	The *x* values are constant. At least NDEG + 1 distinct *x* values are needed to fit a NDEG polynomial.
3	4	The *y* values are constant. A zero order polynomial is fit. High order coefficients are set to zero.
3	5	There are too few observations to fit the desired degree polynomial. High order coefficients are set to zero.
3	6	A perfect fit was obtained with a polynomial of degree less than NDEG. High order coefficients are set to zero.

3. If NDEG is greater than 10, the accuracy of the results may be questionable.

Example

A polynomial model is fitted to data discussed by Neter and Wasserman (1974, pages 279-285). The data set contains the response variable *y* measuring coffee sales (in hundred gallons) and the number of self-service coffee dispensers. Responses for fourteen similar cafeterias are in the data set.

```
      USE RCURV_INT
      USE WRRRN_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    NDEG, NOBS
      PARAMETER  (NDEG=2, NOBS=14)
!
      REAL       B(NDEG+1), SSPOLY(NDEG+1), STAT(10), XDATA(NOBS), &
                 YDATA(NOBS)
      CHARACTER  CLABEL(11)*15, RLABEL(1)*4
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Mean of X', 'Mean of Y', &
                 'Variance X', 'Variance Y', 'R-squared', &
                 'DF Reg.', 'SS Reg.', 'DF Error', 'SS Error', &
                 'Pts. with NaN'/
      DATA XDATA/0., 0., 1., 1., 2., 2., 4., 4., 5., 5., 6., 6., 7., &
           7./
      DATA YDATA/508.1, 498.4, 568.2, 577.3, 651.7, 657.0, 755.3, &
           758.9, 787.6, 792.1, 841.4, 831.8, 854.7, 871.4/
!
      CALL RCURV (XDATA, YDATA, B, SSPOLY=SSPOLY, STAT=STAT)
!
      CALL WRRRN ('B', B, 1, NDEG+1, 1)
      CALL WRRRN ('SSPOLY', SSPOLY, 1, NDEG+1, 1)
      CALL WRRRL ('%/STAT', STAT, RLABEL, CLABEL, 1, 10, 1, FMT='(2W10.4)')
      END
```

Output

```
              B
        1         2         3
    503.3      78.9      -4.0

                SSPOLY
             1           2           3
    7077152.0    220644.2      4387.7

                                    STAT
    Mean of X   Mean of Y  Variance X  Variance Y   R-squared     DF Reg.
        3.571       711.0       6.418     17364.8       99.69           2

     SS Reg.    DF Error    SS Error  Pts. with NaN
    225031.9           11       710.5              0
```

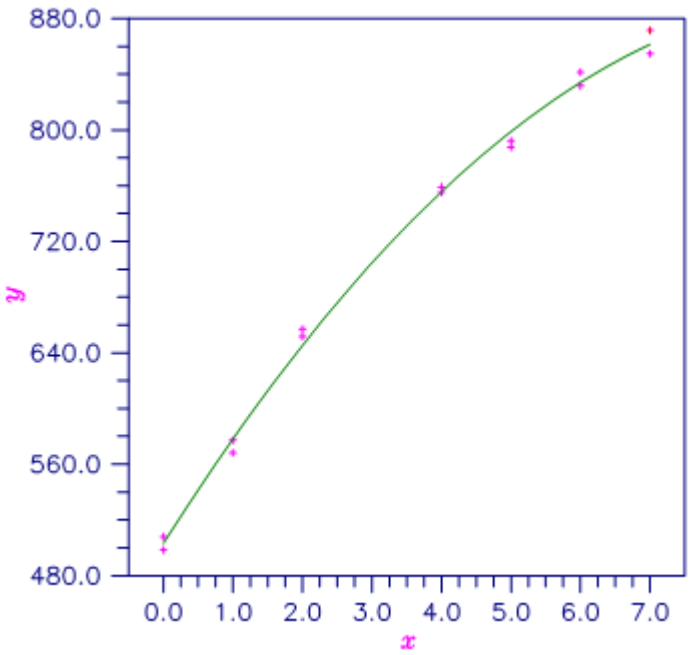


Figure 7, Plot of Data and Second Degree Polynomial Fit

RPOLY

Analyzes a polynomial regression model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

IND — Column number IND of X contains the data for the independent (explanatory) variable. (Input)

MAXDEG — Maximum degree of polynomial to be fit. (Input)

NDEG — Degree of final polynomial regression. (Output)

COEF — NDEG + 1 by 4 matrix containing statistics relating to the coefficients of the polynomial model. (Output)
Row 1 corresponds to the intercept. Row 1 + *i* corresponds to the coefficient of x^i. The columns are described as follows:

Col.	Description
1	Estimated coefficient
2	Estimated standard error of the estimated coefficient
3	*t*-statistic for the test the coefficient is zero
4	*p*-value for the two-sided *t* test

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)

IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies. If X(*i*, IFRQ) = 0.0, none of the remaining elements of row *i* of X are referenced, and updating of statistics is skipped for row *i*.

Default: IFRQ = 0.

IWT — Weighting option. (Input)

IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights, and the computed prediction interval uses AOV (8) = X(*i*, IWT) for the estimated variance of a future response.

Default: IWT = 0.

IPRED — Prediction interval option. (Input)

IPRED = 0 means that prediction intervals are desired for a single future response. For positive IPRED, column number IPRED of X contains the number of future responses for which a prediction interval is desired on the average of the future responses.

Default: IPRED = 0.

CONPCM — Confidence level for two-sided interval estimates on the mean in percent. (Input)

Default: CONPCM = 95.0.

CONPCP — Confidence level for two-sided prediction intervals in percent. (Input)

Default: CONPCP = 95.0.

ICRIT — Criterion option. (Input)

Default: ICRIT = 0.

ICRIT	Meaning
0	Fit a MAXDEG-th degree polynomial.
1	Fit the lowest degree polynomial with an R^2 (in percent) of at least CRIT.
2	Fit the lowest degree polynomial with a lack-of-fit *F* test not significant at level CRIT percent.

CRIT — Criterion in percent. (Input, if ICRIT = 1 or ICRIT = 2, not referenced if ICRIT = 0)

Default: CRIT = 95.0.

CRIT	Meaning of CRIT
1	R^2 (in percent) that the fitted polynomial must achieve. A common choice is 95.0.
2	Significance level (in percent) for the lack-of-fit test that the fitted polynomial must not exceed. A common choice is 5.0.

LOF — Lack of fit option. (Input)

If ICRIT = 2, LOF must equal 1.

Default: LOF = 0.

LOF	Action
0	TLOF is not computed.
1	TLOF is computed.

IPRINT — Printing option. (Input)

Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	AOV, SQSS, COEF, TLOF are printed.
2	AOV, SQSS, COEF, TLOF, unusual cases in CASE and plots of the data,and the fitted polynomial are printed.
3	AOV, SQSS, COEF, TLOF, CASE, plots of the data, the fitted polynomial, and the residuals are printed.

AOV — Vector of length 15 that contains statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

SQSS — NDEG by 4 matrix containing sequential statistics for the polynomial model. (Output)
Row *i* corresponds to x^i(*i* = 1, 2, …, NDEG). The columns are described as follows:

Col.	Description
1	Degrees of freedom
2	Sum of squares
3	*F*-statistic
4	*p*-value

LDSQSS — Leading dimension of SQSS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSQSS = size (SQSS,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

TLOF — NDEG by 4 matrix containing tests of lack of fit for each degree of the polynomial. (Output, if LOF = 1)
Row *i* corresponds to x^i(*i* = 1, 2, …, NDEG). The columns are described as follows:

Col.	Description
1	Degrees of freedom
2	Lack-of-fit sum of squares
3	*F* test for lack of fit of the polynomial model of degree *i*
4	*p*-value for the *F* test

If LOF = 0, TLOF is not referenced and can be a 1 by 1 array.

LDTLOF — Leading dimension of TLOF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTLOF = size (TLOF,1).

CASE — NOBS by 12 matrix containing the case statistics. (Output)
Columns 1 through 12 contain the following:

Col.	Description
1	Observed response
2	Predicted response
3	Residual
4	Leverage
5	Standardized residual
6	Jackknife residual
7	Cook's distance
8	DFFITS
9, 10	Confidence interval on the mean
11, 12	Prediction interval

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of CASE containing NaN (not a number). (Output)

FORTRAN 90 Interface

Generic: CALL RPOLY (X, IRSP, IND, MAXDEG, NDEG, COEF [, ...])
Specific: The specific interface names are S_RPOLY and D_RPOLY.

FORTRAN 77 Interface

Single: CALL RPOLY (NOBS, NCOL, X, LDX, IRSP, IND, IFRQ, IWT, IPRED, CONPCM, CONPCP, MAXDEG, ICRIT, CRIT, LOF, IPRINT, NDEG, AOV, SQSS, LDSQSS, COEF, LDCOEF, TLOF, LDTLOF, CASE, LDCASE, NRMISS)

Double: The double precision name is DRPOLY.

Description

Routine RPOLY computes estimates of the regression coefficients in a polynomial (curvilinear) regression model. The degree of the polynomial can be specified, or the degree of the polynomial can be determined by RPOLY under one of two criteria:

1. If some of the *x* settings are repeated, the lowest degree polynomial can be fit whose lack of fit is not significant at a specified level.
2. The lowest degree polynomial can be fitted with an R^2 that meets a specified lower bound.

In addition to the computation of the fit, RPOLY computes and prints summary statistics (analysis of variance, sequential sums of squares, *t* tests for the coefficients, tests for lack of fit), case statistics (diagnostics for individual cases, confidence and prediction intervals), and plots (data, fitted data, and residuals).

Routine RPOLY computes estimates of the regression coefficients in a polynomial regression model using orthogonal polynomials. The reparameterization of the polynomial model in terms of orthogonal polynomials has the advantage that the loss of accuracy resulting from forming powers of the *x* settings is avoided. All results are returned to the user for the original model.

Often a predicted value and a confidence interval are desired for a setting of the independent variable not used in computing the regression fit. This is accomplished by including an extra row in the data matrix with the desired setting of the independent variable and with the response set equal to NaN (not a number). NaN can be retrieved by AMACH(6), which is documented in the Reference Material. The row of the data matrix containing NaN will be omitted from the computations for determining the regression fit, and a prediction and a confidence interval for the missing response will be computed from the given setting of the independent variable.

Routine RPOLY is based on the algorithm of Forsythe (1957). A modification to Forsythe's algorithm suggested by Shampine (1975) is used for computing the polynomial coefficients. A discussion of Forsythe's algorithm and Shampine's modification appears in Kennedy and Gentle (1980, pages 342-347). A modification to Forsythe's algorithm is made for the inclusion of weights (Kelly 1967, page 68).

Comments

1. Workspace may be explicitly provided, if desired, by use of R2OLY/DR2OLY. The reference is:

```
CALL R2OLY (NOBS, NCOL, X, LDX, IRSP, IND, IFRQ, IWT, IPRED, CONPCM, CONPCP,
    MAXDEG, ICRIT, CRIT, LOF, IPRINT, NDEG, AOV, SQSS, LDSQSS, COEF, LDCOEF,
    TLOF, LDTLOF, CASE, LDCASE, NRMISS, WK, IWK)
```

The additional arguments are as follows:

WK — Work vector of length $\texttt{MAXDEG}^2 + 8 * \texttt{MAXDEG} + 8 * \texttt{NOBS} + 5$

IWK — Work vector of length NOBS.

2. Informational errors

Type	Code	Description
4	1	An invalid weight is encountered. Weights must be nonnegative.
4	2	An invalid frequency is encountered. Frequencies must be nonnegative.
4	7	The independent variable is constant. At least two distinct settings of the independent variable are needed.
4	8	The number of future observations for a prediction interval must be positive.
3	4	The response is constant. A zero degree polynomial is fit.
3	5	There are too few observations to fit the desired degree polynomial. NDEG is set to one less than the number of valid observations.
3	6	A perfect fit to the data was obtained with a polynomial of lower degree than MAXDEG.

Example

A polynomial model is fitted to data discussed by Neter and Wasserman (1974, pages 279-285). The data set contains the response variable *y* measuring coffee sales (in hundred gallons) and the number of self-service coffee dispensers. Responses for fourteen similar cafeterias are in the data set. Some of the cafeterias have the same number of dispensers so that lack of fit of the model can be assessed.

```
      USE RPOLY_INT

      IMPLICIT   NONE
      INTEGER    LDCASE, LDCOEF, LDSQSS, LDTLOF, LDX, MAXDEG, NCOL, &
                 NOBS, J
      PARAMETER  (MAXDEG=2, NCOL=2, NOBS=14, LDCASE=NOBS, &
                 LDCOEF=MAXDEG+1, LDSQSS=MAXDEG, LDTLOF=MAXDEG, &
                 LDX=NOBS)
!
      INTEGER    ICRIT, IFRQ, IND, IPRED, IPRINT, IRSP, IWT, LOF, &
                 NDEG, NRMISS
      REAL       AOV(15), CASE(LDCASE,12), COEF(LDCOEF,4), CONPCM, &
                 CONPCP, CRIT, SQSS(LDSQSS,4), TLOF(LDTLOF,4), &
                 X(LDX,NCOL)
!
      DATA (X(1,J),J=1,2)  /0.0, 508.1/
      DATA (X(2,J),J=1,2)  /5.0, 787.6/
```

```
      DATA (X(3,J),J=1,2)  /0.0, 498.4/
      DATA (X(4,J),J=1,2)  /1.0, 568.2/
      DATA (X(5,J),J=1,2)  /2.0, 651.7/
      DATA (X(6,J),J=1,2)  /7.0, 854.7/
      DATA (X(7,J),J=1,2)  /2.0, 657.0/
      DATA (X(8,J),J=1,2)  /4.0, 755.3/
      DATA (X(9,J),J=1,2)  /6.0, 831.8/
      DATA (X(10,J),J=1,2) /4.0, 758.9/
      DATA (X(11,J),J=1,2) /5.0, 792.1/
      DATA (X(12,J),J=1,2) /6.0, 841.4/
      DATA (X(13,J),J=1,2) /7.0, 871.4/
      DATA (X(14,J),J=1,2) /1.0, 577.3/
!
      IRSP   = 2
      IND    = 1
      LOF    = 1
      IPRINT = 1
      CALL RPOLY (X, IRSP, IND, MAXDEG, NDEG, COEF, lof=lof, IPRINT=IPRINT,&
      aov=aov,sqss=sqss, tlof=tlof, case=case, nrmiss=nrmiss)
!
      END
```

Output

```
R-squared   Adjusted  Est. Std. Dev.              Coefficient of
(percent)  R-squared  of Model Error        Mean  Var. (percent)
   99.685     99.628           8.037       711.0            1.13

                  * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
Source               DF     Squares      Square  Overall F  Larger F
Regression            2    225031.9    112515.9   1741.748    0.0000
Residual             11       710.6        64.6
Corrected Total      13    225742.5

          * * * Inference on Coefficients * * *
                     Standard                 Prob. of
Coef.    Estimate       Error  t-statistic  Larger |t|
   1       503.3       4.791      105.054      0.0000
   2        78.9       3.453       22.865      0.0000
   3        -4.0       0.482       -8.242      0.0000

            * * * Sequential Statistics * * *
Degree of   Degrees of      Sum of               Prob. of
Polynomial     Freedom     Squares  F-statistic  Larger F
         1           1    220644.1     3415.574    0.0000
         2           1      4387.7       67.922    0.0000

            * * * Tests of Lack of Fit * * *
Degree of   Degrees of      Sum of               Prob. of
Polynomial     Freedom     Squares  F-statistic  Larger F
         1           5      4793.7       22.031    0.0004
         2           4       406.0        2.332    0.1547
```

RCOMP

Generates an orthogonal central composite design.

Required Arguments

XMIN — Vector of length NVAR with the minimum values. (Input)
XMIN(*i*) is the minimum for the *i*-th variable.

XMAX — Vector of length NVAR with the maximum values. (Input)
XMAX(*i*) is the maximum for the *i*-th variable.

NCENTR — Number of center points. (Input)
NCENTR must be greater than 0.

X — NPTS by NVAR matrix containing the orthogonal central composite design. (Output)
Design settings for variable I are contained in column I of X. (I = 1, 2, ..., NVAR)

Optional Arguments

NVAR — Number of explanatory variables. (Input)
NVAR must be greater than or equal to 2 and less than or equal to 12.
Default: NVAR = size (XMIN,1).

IFREP — Option for the fractional replicate of the 2NVAR design selected. (Input)
IFREP is referenced only if NVAR is greater than or equal to 5. In the following table, the design points in the fractional replicate part of the design are defined using modulo 2 arithmetic. Each variable is coded 0 or 1 to represent the low and high values of the variable.
Default: IFREP = 0.

Defining Equations

5 $$x_1 + x_2 + x_3 + x_4 + x_5 = \begin{cases} 0 & \text{if IFREP} = 0 \\ 1 & \text{if IFREP} = 1 \end{cases}$$

6 $$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = \begin{cases} 0 & \text{if IFREP} = 0 \\ 1 & \text{if IFREP} = 1 \end{cases}$$

Defining Equations

7
$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 = \begin{cases} 0 & \text{if IFREP} = 0 \\ 1 & \text{if IFREP} = 1 \end{cases}$$

8
$$\begin{matrix}(x_1 + x_2 + x_3 + x_4 + x_5, \\ x_4 + x_5 + x_6 + x_7 + x_8)\end{matrix} = \begin{cases} (0,0) & \text{if IFREP} = 0 \\ (0,1) & \text{if IFREP} = 1 \\ (1,0) & \text{if IFREP} = 2 \\ (1,1) & \text{if IFREP} = 3 \end{cases}$$

9
$$\begin{matrix}(x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \\ x_4 + x_5 + x_6 + x_7 + x_8 + x_9)\end{matrix} = \begin{cases} (0,0) & \text{if IFREP} = 0 \\ (0,1) & \text{if IFREP} = 1 \\ (1,0) & \text{if IFREP} = 2 \\ (1,1) & \text{if IFREP} = 3 \end{cases}$$

10
$$\begin{matrix}(x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \\ x_1 + x_2 + x_3 + x_7 + x_8 + x_9, \\ x_1 + x_2 + x_4 + x_5 + x_7 + x_8 + x_{10})\end{matrix} = \begin{cases} (0,0,0) & \text{if IFREP} = 0 \\ (0,0,1) & \text{if IFREP} = 1 \\ \vdots & \\ (1,1,1) & \text{if IFREP} = 7 \end{cases}$$

11
$$\begin{matrix}(x_1 + x_2 + x_3 + x_4 + x_5 + x_6, \\ x_1 + x_2 + x_6 + x_9 + x_{10}, \\ x_1 + x_5 + x_6 + x_7 + x_{10} + x_{11}, \\ x_1 + x_3 + x_5 + x_8 + x_{11})\end{matrix} = \begin{cases} (0,0,0,0) & \text{if IFREP} = 0 \\ (0,0,0,1) & \text{if IFREP} = 1 \\ \vdots & \\ (1,1,1,1) & \text{if IFREP} = 15 \end{cases}$$

12
$$\begin{matrix}(x_1 + x_2 + x_3 + x_4 + x_9 + x_{10}, \\ x_1 + x_2 + x_5 + x_6 + x_9 + x_{11}, \\ x_1 + x_4 + x_5 + x_7 + x_{11} + x_{12}, \\ x_1 + x_2 + x_7 + x_8 + x_{10} + x_{11})\end{matrix} = \begin{cases} (0,0,0,0) & \text{if IFREP} = 0 \\ (0,0,0,1) & \text{if IFREP} = 1 \\ \vdots & \\ (1,1,1,1) & \text{if IFREP} = 15 \end{cases}$$

NPTS — Number of design points. (Output)

NVAR	NPTS
2 thru 4	2^{NVAR} + 2 * NVAR + NCENTR
5 thru 7	2^{NVAR-1} + 2 * NVAR + NCENTR
8 or 9	2^{NVAR-2} + 2 * NVAR + NCENTR
10	2^{NVAR-3} + 2 * NVAR + NCENTR
11 or 12	2^{NVAR-4} + 2 * NVAR + NCENTR

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

FORTRAN 90 Interface

Generic: `CALL RCOMP (XMIN, XMAX, NCENTR, X [, ...])`
Specific: The specific interface names are **S_RCOMP** and **D_RCOMP**.

FORTRAN 77 Interface

Single: `CALL RCOMP (NVAR, XMIN, XMAX, NCENTR, IFREP, NPTS, X, LDX)`
Double: The double precision name is **DRCOMP**.

Description

Routine **RCOMP** generates an orthogonal central composite design from the minimum and maximum value for each of *n* (input in **NVAR**) variables, where $2 \le n \le 12$. An orthogonal central composite design is a 2^{-k} replicate of a 2^n factorial design, i.e., a 2^{n-k} fractional factorial, augmented by $2n$ axial points and *m* (input in **NCENTR**) center points. The values of *n* and *k* used by **RCOMP** are given by the following table:

2, 3, 4	0
5, 6, 7	1
8, 9	2
10	3
11, 12	4

The fractional factorial part of all designs generated by **RCOMP** are of resolution *V* or greater. This means the fractions allow the overall mean, all the main effects, and all the two-factor interactions to be estimated. For a further discussion, see John (1971, pages 148-157).

Experimental designs for fitting a second-order response surface must contain at least three levels of each variable in order for the regression coefficients to be estimated. Orthogonal central composite designs provide a useful alternative to the 3^n factorial design, which can require an excessive number of design points. On a *per observation basis*, the orthogonal central composite design is no worse than the 3^n factorial design with regard to efficiency for estimating the regression coefficients of the square and crossproduct variables (see Meyers 1971, pages 134-136). The design assumes three factor and higher-way interactions are negligible.

Meyers (1971, chapter 7) and John (1971, pages 204-206) discuss the generation of the design. The number of design points (stored in **NPTS**) is $2^{n-k} + 2n + m$. Each variable in the design appears at five different levels. For a second-order response surface model with the *x* variables coded {-α, -1, 0, 1, α} and with pure quadratic terms corrected for the mean

$$c = \frac{2^{n-k} + 2\alpha^2}{2^{n-k} + 2n + m}$$

the design produces a diagonal X^TX matrix. Let

$$\alpha = \left(\frac{\left(\sqrt{2^{n-k} + 2n + m} - \sqrt{2^{n-k}} \right)^2 2^{n-k}}{4} \right)^{1/4}$$

and let the minimum and maximum value of the *j*-th variable be denoted by x_{1j} and x_{2j}, respectively. The following table gives the formulas for the coded and decoded variable settings:

Coded Setting for Variable *j*	Decoded Setting for Variable *j*
-α	x_{1j}
-1	$\frac{x_{1j} - x_{2j}}{2\alpha} + \frac{x_{1j} + x_{2j}}{2}$
0	$\frac{x_{1j} + x_{2j}}{2}$
1	$\frac{x_{2j} - x_{1j}}{2\alpha} + \frac{x_{1j} + x_{2j}}{2}$
α	x_{2j}

Example

This example uses two variables and their respective minimum and maximum values to generate an orthogonal central composite design with four center points.

```
          USE RCOMP_INT
          USE UMACH_INT
          USE WRRRN_INT

          IMPLICIT   NONE
          INTEGER    NVAR, NCENTR, LDX, NPTS, NOUT
          PARAMETER  (NVAR=2, NCENTR=4, LDX=2**NVAR+2*NVAR+NCENTR)
          REAL       X(LDX,NVAR), XMAX(NVAR), XMIN(NVAR)
          DATA       XMIN /251.0,73.0/ XMAX/295.0, 87.0/
     !
          CALL RCOMP (XMIN, XMAX, NCENTR, X, NPTS=NPTS)
     !
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) 'NPTS = ', NPTS
          CALL WRRRN ('X', X)
          END
```

Output

```
     NPTS =   12

              X
              1       2
      1   291.2    85.8
      2   291.2    74.2
      3   254.8    85.8
      4   254.8    74.2
      5   273.0    80.0
      6   273.0    80.0
      7   273.0    80.0
      8   273.0    80.0
      9   251.0    80.0
     10   295.0    80.0
     11   273.0    73.0
     12   273.0    87.0
```

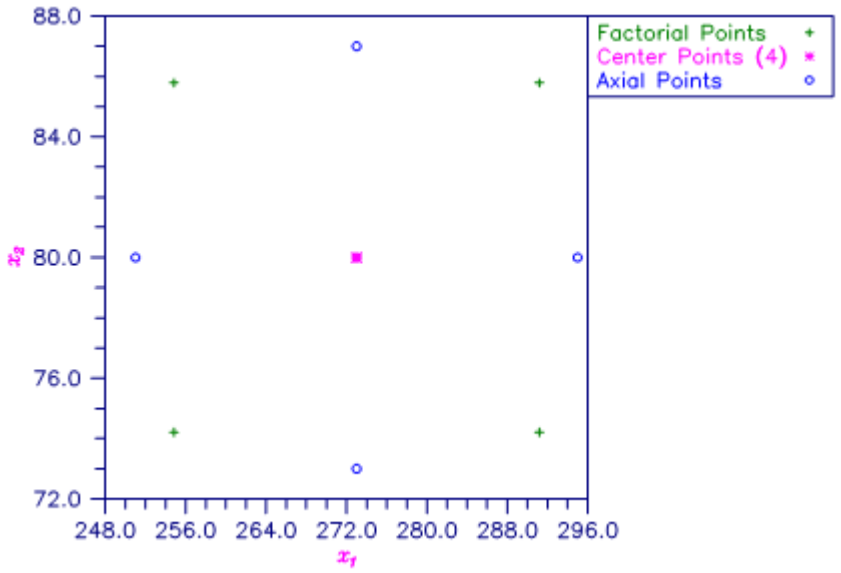


Figure 8, Orthogonal Central Composite Design With Four Center Points

RFORP

Fits an orthogonal polynomial regression model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

IND — Column number IND of X contains the data for the independent (explanatory) variable. (Input)

MAXDEG — Maximum degree of polynomial to be fit. (Input)

NDEG — Degree of final polynomial regression. (Output)

A — Vector of length MAXDEG containing constants used to generate orthogonal polynomials. (Output)
Only the first NDEG elements of A are referenced.

B — Vector of length MAXDEG containing constants used to generate orthogonal polynomials. (Output)
Only the first NDEG elements of B are referenced.

SCOEF — Vector of length 1 + MAXDEG containing the regression coefficients α of the fitted model using the scaled version of $x(z)$. (Output)
Only the first 1 + NDEG elements of SCOEF are referenced.

$$\hat{\alpha}_0 = \text{SCOEF}(1)$$

is the estimated intercept and equals the response mean.

$$\hat{\alpha}_i = \text{SCOEF}(1 + i)$$

contains the estimated coefficient for the i-th order orthogonal polynomial using the scaled version of $x(z)$.

D — Vector of length MAXDEG + 1 containing the diagonal elements of the (diagonal) sums of squares and crossproducts matrix. (Output)
The sum of squares due to the i-th degree orthogonal polynomial is given by

$$D(i+1) * \hat{\alpha}_i^2$$

Only the first NDEG + 1 elements of D are referenced.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies. If X(*i*, IFRQ) = 0.0, none of the remaining elements of row *i* of X are referenced, and updating of statistics is skipped for row *i*.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights.
Default: IWT = 0.

ICRIT — Criterion option. (Input)
Default: ICRIT = 0.

ICRIT	Meaning
0	Fit a MAXDEG-th degree polynomial.
1	Fit the lowest degree polynomial with an R^2 (in percent) of at least CRIT.
2	Fit the lowest degree polynomial with a lack-of-fit *F* test not significant at level CRIT percent.

CRIT — Criterion in percent. (Input, if ICRIT = 1 or ICRIT = 2)
Default: CRIT = 95.0.

ICRIT	Meaning of CRIT
1	R^2 (in percent) that the fitted polynomial must achieve. A common choice is 95.0.
2	Significance level (in percent) for the lack-of-fit test that the fitted polynomial must not exceed. A common choice is 5.0.

LOF — Lack-of-fit option. (Input)
If ICRIT = 2, LOF must equal 1.
Default: LOF = 0.

LOF	Action
0	DFPE and SSPE are not computed.
1	DFPE and SSPE are computed.

SMULTC — Multiplicative constant used to compute a scaled version of *x*, say *z*, on the interval -2 to 2, inclusive. (Output)

SADDC — Additive constant used to compute a scaled version of *x*(*z*) on the interval -2 to 2, inclusive. (Output)

DFE — Degrees of freedom for error. (Output)

SSE — Sum of squares for error. (Output)

DFPE — Degrees of freedom for pure error. (Output, if LOF = 1)

SSPE — Sum of squares for pure error. (Output, if LOF = 1)

NRMISS — Number of rows of data encountered that contain any missing values for the independent, response, weight, or frequency variables. (Output)
NaN (not a number) is used as the missing value code. Any row of X containing NaN as a value of the independent, response, weight, or frequency variables is omitted from the fit.

FORTRAN 90 Interface

Generic: CALL RFORP (X, IRSP, IND, MAXDEG, NDEG, A, B, SCOEF, D [, ...])
Specific: The specific interface names are S_RFORP and D_RFORP.

FORTRAN 77 Interface

Single: CALL RFORP (NOBS, NCOL, X, LDX, IRSP, IND, IFRQ, IWT, MAXDEG, ICRIT, CRIT, LOF, NDEG, SMULTC, SADDC, A, B, SCOEF, D, DFE, SSE, DFPE, SSPE, NRMISS)
Double: The double precision name is DRFORP.

Description

Routine **RFORP** computes estimates of the regression coefficients in a polynomial regression model using orthogonal polynomials. The reparameterization of the polynomial model in terms of orthogonal polynomials has the advantage that the loss of accuracy resulting from forming powers of the *x* values is avoided. The design of **RFORP** assumes that further computations such as summary statistics or case statistics are needed. For this reason, the results returned by **RFORP** are for the reparameterized model in terms of orthogonal polynomials. This enables computational accuracy to be maintained for the subsequent computations. Routine RSTAP can be used to compute summary statistics for the original polynomial model given the results from **RFORP**. Routine RCASP can be used to compute case statistics for the original polynomial model given the results from **RFORP**.

The degree of the polynomial can be specified, or the degree of the polynomial can be determined by **RFORP** under one of two criteria:

1. If some of the *x* values are repeated, the lowest degree polynomial can be fitted whose lack of fit is not significant at a specified level.
2. The lowest degree polynomial can be fitted with an R^2 that meets a specified lower bound.

Routine **RFORP** is based on the algorithm of Forsythe (1957). A modification to Forsythe's algorithm is made for the inclusion of weights (Kelly 1967, page 68).

Let x_i be a value of the independent variable. The x_i's are scaled to the interval [-2, 2] for computational accuracy. The scaled version of the independent variable is computed by the formula $z_i = mx_i + c$. The multiplicative scaling constant *m* (stored in **SMULTC**) is

$$m = \frac{4}{\max_i\left(x_i\right) - \min_i\left(x_i\right)}$$

The additive constant *c* (stored in **SADDC**) is

$$c = \frac{2\left(\min_i\left(x_i\right) + \max_i\left(x_i\right)\right)}{\min_i\left(x_i\right) - \max_i\left(x_i\right)}$$

Orthogonal polynomials are evaluated using the three-term recurrence relationship

$$p_j(z) = (z - a_j)p_{j-1}(z) - b_j p_{j-2}(z)$$

beginning with the initial polynomials

$$p_0(z) = 1 \quad \text{and} \quad p_1(z) = z - a_1$$

The a_j's and b_j's (stored in **A** and **B**) are computed to make the $p_j(z)$'s orthogonal with respect to the the set of weights w_i, and over the set z_i.

The fitted model is

$$\hat{y}_i = \hat{\alpha}_0 + \hat{\alpha}_1 p_1(z_i) + \cdots + \hat{\alpha}_k p_k(z_i)$$

The

$$\hat{\alpha}_j \text{ values}$$

(stored in **SCOEF**) are computed (Shampine 1975) by

$$\hat{\alpha}_j = \frac{\sum_{i=1}^{n} e_i w_i p_j(z_i)}{d_j}$$

where $e_i = y_i - p_{j-1}(z_i)$ and

$$d_j = \sum_{i=1}^{n} w_i \left[p_j(z_i) \right]^2$$

The d_j's (stored in **D**) can be used to compute the sum of squares due to the j-th orthogonal polynomial by

$$Q_j = d_j \hat{\alpha}_j^2$$

A more complete description of Forsythe's algorithm and the modification of Shampine appears in Kennedy and Gentle (1980, pages 342-347).

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2ORP/DR2ORP**. The reference is:

   CALL R2ORP (**NOBS**, **NCOL**, **X**, **LDX**, **IRSP**, **IND**, **IFRQ**, **IWT**, **MAXDEG**, **ICRIT**, **CRIT**, **LOF**, **NDEG**, **SMULTC**, **SADDC**, **A**, **B**, **SCOEF**, **D**, **DFE**, **SSE**, **DFPE**, **SSPE**, **NRMISS**, **WK**, **IWK**)

   The additional arguments are as follows:

   WK — Work vector of length 8 * **NOBS**.

   IWK — Work vector of length **NOBS**.

2. Informational errors

Type	Code	Description
3	4	The response variable is constant. A zero order polynomial is fit. High order coefficients are set to zero.
3	5	There are too few observations to fit the desired degree polynomial. High order coefficients are set to zero.
3	6	A perfect fit is obtained with a polynomial of lower degree than MAXDEG.
4	1	An invalid weight is encountered.
4	2	An invalid frequency is encountered.
4	3	Each row of X contains a missing value.
4	7	The independent variable is constant. At least two distinct settings of the independent variable are needed.

3. The orthogonal polynomials evaluated at each scaled *x* value (*z*) are computed from A and B as follows:

   POLY(I, 1) = Z(I) - A(1)

   POLY(I, 2) = (Z(I) - A(2)) * POLY(I, 1) - B(2)

   POLY (I, J) = (Z(I) - A(J)) * POLY(I, J - 1) - B(J) * POLY(I, J - 2)

   for J = 3 through NDEG.

Example

A polynomial model is fitted to data discussed by Neter and Wasserman (1974, pages 279-285). The data set contains the response variable *y* measuring coffee sales (in hundred gallons) and the number of self-service coffee dispensers. Responses for fourteen similar cafeterias are in the data set, some of the cafeterias have the same number of dispensers so that lack of fit of the model can be assessed.

```
      USE RFORP_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDX, MAXDEG, NCOL, NOBS, J
      PARAMETER  (MAXDEG=2, NCOL=2, NOBS=14, LDX=NOBS)
!
      INTEGER    IND, IRSP, LOF, NDEG, NOUT, NRMISS
      REAL       A(MAXDEG), B(MAXDEG), CRIT, D(MAXDEG+1), DFE, DFPE, &
                 SADDC, SCOEF(MAXDEG+1), SMULTC, SSE, SSPE, X(LDX,NCOL)
!
      DATA (X(1,J),J=1,2)  /0.0, 508.1/
      DATA (X(2,J),J=1,2)  /5.0, 787.6/
      DATA (X(3,J),J=1,2)  /0.0, 498.4/
```

```
      DATA (X(4,J),J=1,2)  /1.0, 568.2/
      DATA (X(5,J),J=1,2)  /2.0, 651.7/
      DATA (X(6,J),J=1,2)  /7.0, 854.7/
      DATA (X(7,J),J=1,2)  /2.0, 657.0/
      DATA (X(8,J),J=1,2)  /4.0, 755.3/
      DATA (X(9,J),J=1,2)  /6.0, 831.8/
      DATA (X(10,J),J=1,2) /4.0, 758.9/
      DATA (X(11,J),J=1,2) /5.0, 792.1/
      DATA (X(12,J),J=1,2) /6.0, 841.4/
      DATA (X(13,J),J=1,2) /7.0, 871.4/
      DATA (X(14,J),J=1,2) /1.0, 577.3/
!
      IRSP  = 2
      IND   = 1
      LOF   = 1
      CALL RFORP (X, IRSP, IND, MAXDEG, NDEG, A, B, SCOEF, D, lof=lof, &
                  smultc=smultc, saddc=saddc, dfe=dfe, sse=sse, &
                  dfpe=dfpe, sspe=sspe, nrmiss=nrmiss)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NDEG =   ', NDEG
      CALL WRRRN ('A', A, 1, NDEG, 1)
      CALL WRRRN ('B', B, 1, NDEG, 1)
      WRITE (NOUT,*) 'SMULTC = ', SMULTC
      WRITE (NOUT,*) 'SADDC  = ', SADDC
      CALL WRRRN ('SCOEF', SCOEF, 1, NDEG+1, 1)
      CALL WRRRN ('D', D, 1, NDEG+1, 1)
      WRITE (NOUT,*) 'DFE =    ', DFE
      WRITE (NOUT,*) 'SSE =    ', SSE
      WRITE (NOUT,*) 'DFPE =   ', DFPE
      WRITE (NOUT,*) 'SSPE =   ', SSPE
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
 NDEG = 2

        A
      1         2
0.04082  -0.07996

      B
    1       2
0.000   1.946
SMULTC =    0.571429
SADDC  =    -2.00000

        SCOEF
    1       2       3
711.0    90.0   -12.2

          D
    1       2       3
14.00   27.24   29.69
DFE =        11.0000
SSE =        710.594
DFPE =       7.00000
```

```
    SSPE =       304.626
    NRMISS =   0
```

RSTAP

Computes summary statistics for a polynomial regression model given the fit based on orthogonal polynomials.

Required Arguments

A — Vector of length **NDEG** containing constants used to generate orthogonal polynomials. (Input)

B — Vector of length **NDEG** containing constants used to generate orthogonal polynomials. (Input)

SMULTC — Multiplicative constant used to compute the scaled version of *x*, say *z*, on the interval **-**2 to 2, inclusive. (Input)

SADDC — Additive constant used to compute the scaled version of *x*(*z*) on the interval **-**2 to 2, inclusive. (Input)

SCOEF — Vector of length **NDEG** + 1 containing the regression coefficients of the fitted model using the scaled version of the original data. (Input)
SCOEF(1) is the estimated intercept. **SCOEF**(1 + *i*) contains the estimated coefficient for the *i*-th order orthogonal polynomial using *z*.

D — Vector of length **NDEG** + 1 containing the diagonal elements of the (diagonal) sums of squares and crossproducts matrix. (Input)

DFE — Degrees of freedom for error. (Input)

SSE — Sum of squares for error. (Input)

COEF — **NDEG** + 1 by 4 matrix containing statistics relating to the coefficients of the polynomial model. (Output)
Row 1 corresponds to the intercept. Row 1 + *i* corresponds to the coefficient of x^i. The columns are described as follows:

Col.	Description
1	Estimated coefficient
2	Estimated standard error of estimated coefficient
3	*t*-statistic for the test that the coefficient is zero
4	*p*-value for the two-sided *t* test

Optional Arguments

NDEG — Degree of the polynomial regression. (Input)
Default: NDEG = size (A,1).

LOF — Lack of fit test option. (Input)
Default: LOF = 0.

LOF	Action
0	No lack of fit test is performed.
1	Lack of fit test is performed.

DFPE — Degrees of freedom for pure error. (Input, if LOF = 1)
If LOF = 0, DFPE is not referenced.
Default: DFPE = 1.0.

SSPE — Sum of squares for pure error. (Input, if LOF = 1)
If LOF = 0, SSPE is not referenced.
Default: SSPE = 0.0.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	AOV, SQSS, COEF are printed.

AOV — Vector of length 15 that contains statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for the model
2	Degrees of freedom for error
3	Total (corrected) degrees of freedom
4	Sum of squares for the model

I	AOV(I)
5	Sum of squares for error
6	Total (corrected) sum of squares
7	Model mean square
8	Error mean square
9	Overall *F* -statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimate of the standard deviation
14	Overall mean of *y*
15	Coefficient of variation (in percent)

SQSS — NDEG by 4 matrix containing sequential statistics for the polynomial model. (Output)
Row *i* corresponds to x^i(*i* = 1, 2, …, NDEG). The columns are described as follows:

Col.	Description
1	Degrees of freedom
2	Sum of squares
3	*F* -statistic
4	*p*-value

LDSQSS — Leading dimension of SQSS exactly as specified in the dimension statement of the calling program. (Input)
Default: LDSQSS = size (SQSS,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement of the calling program. (Input)
Default: LDCOEF = size (COEF,1).

TLOF — NDEG by 4 matrix containing tests of lack of fit for each degree of the polynomial. (Output, if LOF = 1)
If LOF = 0, TLOF is not referenced and can be a 1 by 1 array. Row *i* corresponds to x^i(*i* = 1, 2, …, NDEG). The columns are described as follows:

Col.	Description
1	Degrees of freedom
2	Lack of fit sum of squares

Col.	Description
3	*F* test for lack of fit of the polynomial model of degree *i*
4	*p*-value for the *F* test

LDTLOF — Leading dimension of TLOF exactly as specified in the dimension statement of the calling program. (Input)
Default: LDTLOF = size (TLOF,1).

FORTRAN 90 Interface

Generic: CALL RSTAP (A, B, SMULTC, SADDC, SCOEF, D, DFE, SSE, COEF [, ...])
Specific: The specific interface names are S_RSTAP and D_RSTAP.

FORTRAN 77 Interface

Single: CALL RSTAP (NDEG, A, B, SMULTC, SADDC, SCOEF, D, DFE, SSE, LOF, DFPE, SSPE, IPRINT, AOV, SQSS, LDSQSS, COEF, LDCOEF, TLOF, LDTLOF)
Double: The double precision name is DRSTAP.

Description

Routine RSTAP transforms a polynomial regression model, fitted using orthogonal polynomials, into a polynomial function of the original independent variable. In addition, summary statistics (analysis of variance, *t* tests, tests for lack of fit) are computed. Results from routine RFORP, which produces the fit using orthogonal polynomials, are used for input.

The fitted model from RFORP is

$$\hat{y}_i = \hat{\alpha}_0 p_0(z_i) + \hat{\alpha}_1 p_1(z_i) + \quad \cdots \quad + \hat{\alpha}_k p_k(z_i)$$

where the z_i's are the settings of the independent variable x scaled to the interval [-2, 2] and where the $p_j(z)$'s are the orthogonal polynomials. The "$X^T X$" matrix for this model is a diagonal matrix with elements d_j (stored in D). The orthogonal polynomials can be expressed as

$$p_j(z) = \sum_{m=0}^{j} \delta_{jm} z^m$$

First, **RSTAP** computes

$$\hat{\gamma}_j = \sum_{m=j}^{k} \hat{\alpha}_m \delta_{mj}$$

to produce the fit for the polynomial function in terms of the scaled independent variable as given by

$$\hat{y}_i = \hat{\gamma}_0 + \hat{\gamma}_1 z_i + \cdots + \hat{\gamma}_k z_i^k$$

The variances and covariances for the estimated coefficients in this model are given by

$$\text{cov}(\hat{\gamma}_i, \hat{\gamma}_j) = \sigma^2 \sum_{m=\max(i,j)}^{k} \delta_{im} \delta_{jm} d_j^{-1} \quad i,j = 0, 1, \ldots, m$$

Second, **RSTAP** computes

$$\hat{\beta}_j$$

as a linear combination of the

$$\hat{\gamma}_j\text{'s}$$

by the formula

$$\hat{\beta}_j = \sum_{m=0}^{k-j} (-1)^m 2^{2j+m} \binom{j+m}{m} \left(\min_i x_i + \max_i x_i \right)^m \left(\frac{1}{\max_i x_i - \min_i x_i} \right)^{j+m} \hat{\gamma}_{j+m}$$

in order to produce the fit for the polynomial function in terms of the original independent variable as given by

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_k x_i^k$$

The variance of

$$\hat{\beta}_j$$

computed from the variances and covariances of the

$$\hat{\gamma}_j\text{'s}$$

using the usual formula for computing variances of linear combinations of correlated random variables. The sequential sum of squares due to x^j(stored in **SQSS**) is computed by

$$Q_j = d_j \hat{\alpha}_j^2$$

Comments

Workspace may be explicitly provided, if desired, by use of R2TAP/DR2TAP.

The reference is:

CALL R2TAP (NDEG, A, B, SMULTC, SADDC, SCOEF, D, DFE, SSE, LOF, DFPE, SSPE, IPRINT, AOV, SQSS, LDSQSS, COEF, LDCOEF, TLOF, LDTLOF, WK)

The additional argument is:

WK — Work vector of length (NDEG + 1) * (NDEG + 7).

Example

A polynomial model is fitted to data discussed by Neter and Wasserman (1974, pages 279-285). The data set contains the response variable *y* measuring coffee sales (in hundred gallons) and the number of self-service coffee dispensers. Responses for fourteen similar cafeterias are in the data set and some of the cafeterias have the same number of dispensers so that lack of fit of the model can be assessed.

```
      USE RSTAP_INT
      USE RFORP_INT
      IMPLICIT NONE

      INTEGER    LDCOEF, LDSQSS, LDTLOF, LDX, MAXDEG, NCOL, NOBS, J
      PARAMETER  (MAXDEG=2, NCOL=2, NOBS=14, LDCOEF=MAXDEG+1, &
                 LDSQSS=MAXDEG, LDTLOF=MAXDEG, LDX=NOBS)
!
      INTEGER    IND, IPRINT, IRSP, LOF, NDEG, NRMISS
      REAL       A(MAXDEG), AOV(15), B(MAXDEG), COEF(MAXDEG+1,4), &
                 CRIT, D(MAXDEG+1), DFE, DFPE, SADDC, SCOEF(MAXDEG+1), &
                 SMULTC, SQSS(LDSQSS,4), SSE, SSPE, TLOF(MAXDEG,4), &
                 X(LDX,NCOL)
!
      DATA (X(1,J),J=1,2)  /0.0, 508.1/
      DATA (X(2,J),J=1,2)  /5.0, 787.6/
      DATA (X(3,J),J=1,2)  /0.0, 498.4/
      DATA (X(4,J),J=1,2)  /1.0, 568.2/
      DATA (X(5,J),J=1,2)  /2.0, 651.7/
      DATA (X(6,J),J=1,2)  /7.0, 854.7/
      DATA (X(7,J),J=1,2)  /2.0, 657.0/
      DATA (X(8,J),J=1,2)  /4.0, 755.3/
      DATA (X(9,J),J=1,2)  /6.0, 831.8/
      DATA (X(10,J),J=1,2) /4.0, 758.9/
      DATA (X(11,J),J=1,2) /5.0, 792.1/
      DATA (X(12,J),J=1,2) /6.0, 841.4/
      DATA (X(13,J),J=1,2) /7.0, 871.4/
      DATA (X(14,J),J=1,2) /1.0, 577.3/
!
      IRSP  = 2
      IND   = 1
```

```
      LOF   = 1
      CALL RFORP (X, IRSP, IND, MAXDEG, NDEG, A, B, SCOEF, D, LOF=LOF, &
                  SMULTC=SMULTC, SADDC=SADDC, DFE=DFE, SSE=SSE, &
                  DFPE=DFPE, SSPE=SSPE)
!
      IPRINT = 1
      CALL RSTAP (A, B, SMULTC, SADDC, SCOEF, D, DFE, SSE, COEF, &
                  NDEG=NDEG, LOF=LOF, DFPE=DFPE, SSPE=SSPE, IPRINT=IPRINT, &
                  AOV=AOV, SQSS=SQSS, TLOF=TLOF)
      END
```

Output

```
R-squared   Adjusted  Est. Std. Dev.              Coefficient of
(percent)  R-squared  of Model Error        Mean  Var. (percent)
   99.685     99.628           8.037       711.0            1.13

                   * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
Source               DF     Squares      Square  Overall F  Larger F
Regression            2    225031.9    112515.9   1741.748    0.0000
Residual             11       710.6        64.6
Corrected Total      13    225742.5

           * * * Inference on Coefficients * * *
                        Standard                Prob. of
Coef.    Estimate       Error  t-statistic  Larger |t|
    1       503.3       4.791      105.054      0.0000
    2        78.9       3.453       22.865      0.0000
    3        -4.0       0.482       -8.242      0.0000

             * * * Sequential Statistics * * *
Degree of   Degrees of      Sum of               Prob. of
Polynomial     Freedom     Squares  F-statistic  Larger F
         1           1    220644.1     3415.574    0.0000
         2           1      4387.7       67.922    0.0000

            * * * Tests of Lack of Fit * * *
Degree of   Degrees of      Sum of               Prob. of
Polynomial     Freedom     Squares  F-statistic  Larger F
         1           5      4793.7       22.031    0.0004
         2           4       406.0        2.332    0.1547
```

RCASP

Computes case statistics for a polynomial regression model given the fit based on orthogonal polynomials.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

IND — Column number IND of X contains the data for the independent (explanatory) variable. (Input)

NDEG — Degree of the polynomial regression. (Input)

SMULTC — Multiplicative constant used to compute a scaled version of *x* on the interval -2 to 2, inclusive. (Input)

SADDC — Additive constant used to compute a scaled version of *x* on the interval -2 to 2, inclusive. (Input)

A — Vector of length NDEG containing constants used to generate orthogonal polynomials. (Input)

B — Vector of length NDEG containing constants used to generate orthogonal polynomials. (Input)

SCOEF — Vector of length NDEG + 1 containing the regression coefficients

$$\hat{\alpha}$$

of the fitted model using the scaled version of *x*(*z*). (Input)

$$\hat{\alpha}_0 = \text{SCOEF}(1)$$

is the estimated intercept and equals the response mean.

$$\hat{\alpha}_i = \text{SCOEF}(1 + i)$$

contains the estimated coefficient for the *i*-th order orthogonal polynomial using the scaled version of *x*(*z*).

D — Vector of length NDEG + 1 containing the diagonal elements of the (diagonal) sums of squares and crossproducts matrix. (Input)

SSE — Sum of squares for error. (Input)

DFE — Degrees of freedom for error. (Input)

CASE — NOBS by 12 matrix containing the case statistics. (Output)
Columns 1 through 12 contain the following:

Col.	Description
1	Observed response
2	Predicted response
3	Residual
4	Leverage
5	Standardized residual
6	Jackknife residual
7	Cook's distance
8	DFFITS
9, 10	Confidence interval on the mean
11, 12	Prediction interval

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive IWT, column number IWT of X contains the weights, and the computed prediction interval uses SSE/(DFE * X(*i*, IWT)) for the estimated variance of a future response.
Default: IWT = 0.

IPRED — Prediction interval option. (Input)
IPRED = 0 means that prediction intervals are desired for a single future response. For positive IPRED, column number IPRED of X contains the number of future responses for which a prediction interval is desired on the average of the future responses.
Default: IPRED = 0.

CONPCM — Confidence level for two-sided interval estimates on the mean, in percent. (Input)
Default: CONPCM = 95.0.

CONPCP — Confidence level for two-sided prediction intervals, in percent. (Input)
Default: CONPCP = 95.0.

PRINT — Printing option. (Input)
Default: PRINT = 'N'.
PRINT is a character string indicating what is to be printed. The PRINT string is composed of one-character print codes to control printing. These print codes are given as follows:

PRINT(I:I)	Printing that occurs
'A'	All
'N'	None
'1'	Observed response
'2'	Predicted response
'3'	Residual
'4'	Leverage
'5'	Standardized residual
'6'	Jackknife residual
'7'	Cook's distance
'8'	DFFITS
'M'	Confidence interval on the mean
'P'	Prediction interval
'X'	Influential cases (unusual "*x*-value")
'Y'	Outlier cases (unusual "*y*-value")

The concatenated print codes 'A', 'N', '1', ..., 'P' that comprise the PRINT string give the combination of statistics to be printed. Concatenation of these codes with print codes 'X' or 'Y' restricts printing to cases determined to be influential or outliers. Here are a few examples:

PRINT	Printing Action
'A'	All.
'N'	None.
'46'	Leverage and jackknife residual for all cases.
'AXY'	All statistics are printed for cases that are highly influential or are outliers.
'46XY'	Leverage and jackknife residual are printed for cases that are highly influential or are outliers.

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of CASE containing NaN (not a number). (Output)

FORTRAN 90 Interface

Generic: CALL RCASP (X, IRSP, IND, NDEG, SMULTC, SADDC, A, B, SCOEF, D, SSE, DFE, CASE [, ...])

Specific: The specific interface names are S_RCASP and D_RCASP.

FORTRAN 77 Interface

Single: CALL RCASP (NOBS, NCOL, X, LDX, IRSP, IND, IWT, IPRED, CONPCM, CONPCP, NDEG, SMULTC, SADDC, A, B, SCOEF, D, SSE, DFE, PRINT, CASE, LDCASE, NRMISS)

Double: The double precision name is DRCASP.

Description

Routine RCASP assumes a polynomial model

$$y_i = \beta_0 + \beta_1 x_i + \cdots + \beta_k x_i^k + \varepsilon_i \quad i = 1, 2, \ldots, n$$

where the observed values of the y_i's constitute the response, the x_i's are the settings of the independent variable, the β_j's are the regression coefficients and the ε_i's are the errors that are independently distributed normal with mean 0 and variance σ^2/w_i. Given the results of a polynomial regression, fitted using orthogonal polynomials and weights w_i, routine RCASP produces predicted values, residuals, confidence intervals, prediction intervals, and diagnostics for outliers and influential cases.

Often a predicted value and confidence interval are desired for a setting of the independent variable not used in computing the regression fit. This can be accomplished by including the independent variable setting as part of the data matrix and by setting the response equal to NaN (not a number). NaN can be retrieved by AMACH(6).

Results from routine RFORP, which produces the fit using orthogonal polynomials, are used for input. The fitted model from RFORP is

$$\hat{y}_i = \hat{\alpha}_o p_o(z_i) + \hat{\alpha}_1 p_1(z_i) + \cdots + \hat{\alpha}_k p_k(z_i)$$

where the z_i's are settings of the independent variable x scaled to the interval [-2, 2] and where the $p_j(z)$'s are the orthogonal polynomials. The "$X^T X$" matrix for this model is a diagonal matrix with elements d_j (stored in D). The case statistics are easily computed from this model and are equal to those from the original polynomial model with the β_j's as the regression coefficients.

The leverage is computed as

$$h_i = w_i \sum_{j=0}^{k} d_j^{-1} p_j^2(z_i)$$

The estimated variance of

$$\hat{y}_i$$

is given by $h_i s^2 / w_i$. The computation of the remainder of the case statistics follows easily from their definitions. See the Diagnostics for Individual Cases section in the chapter introduction for definitions of the case diagnostics.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2ASP/DR2ASP. The reference is:

   CALL R2ASP (NOBS, NCOL, X, LDX, IRSP, IND, IWT, IPRED, CONPCM, CONPCP, NDEG, SMULTC, SADDC, A, B, SCOEF, D, SSE, DFE, PRINT, CASE, LDCASE, NRMISS, WK)

   The additional argument is:

   WK — Work vector of length NDEG + 1.

2. Informational errors

Type	Code	Description
4	1	A weight is negative. Weights must be nonnegative.
4	8	The number of future observations for a prediction interval must be positive.
3	9	A leverage much greater than one is computed. It is set to one.
3	10	A deleted residual mean square much less than zero is computed. It is set to 0.0.

Example

A polynomial model is fitted to data discussed by Neter and Wasserman (1974, pages 279-285). The data set contains the response variable *y* measuring coffee sales (in hundred gallons) and the number of self-service coffee dispensers. Responses for fourteen similar cafeterias are in the data set.

```
      USE RCASP_INT
      USE RFORP_INT

      IMPLICIT   NONE
      INTEGER    LDCASE, LDCOEF, LDSQSS, LDTLOF, LDX, MAXDEG, NCOL, &
```

```
                    NOBS,  J
          PARAMETER (MAXDEG=2, NCOL=2, NOBS=14, LDCASE=NOBS, &
                    LDCOEF=MAXDEG+1, LDSQSS=MAXDEG, LDTLOF=MAXDEG, &
                    LDX=NOBS)
     !
          INTEGER   ICRIT, IFRQ, IND, IPRED, IRSP, IWT, LOF, NDEG, NRMISS
          REAL      A(MAXDEG), B(MAXDEG), CASE(LDCASE,12), CONPCM, &
                    CONPCP, CRIT, D(MAXDEG+1), DFE, DFPE, SADDC, &
                    SCOEF(MAXDEG+1), SMULTC, SSE, SSPE, X(LDX,NCOL)
          CHARACTER PRINT*1
     !
          DATA (X(1,J),J=1,2)  /0.0, 508.1/
          DATA (X(2,J),J=1,2)  /5.0, 787.6/
          DATA (X(3,J),J=1,2)  /0.0, 498.4/
          DATA (X(4,J),J=1,2)  /1.0, 568.2/
          DATA (X(5,J),J=1,2)  /2.0, 651.7/
          DATA (X(6,J),J=1,2)  /7.0, 854.7/
          DATA (X(7,J),J=1,2)  /2.0, 657.0/
          DATA (X(8,J),J=1,2)  /4.0, 755.3/
          DATA (X(9,J),J=1,2)  /6.0, 831.8/
          DATA (X(10,J),J=1,2) /4.0, 758.9/
          DATA (X(11,J),J=1,2) /5.0, 792.1/
          DATA (X(12,J),J=1,2) /6.0, 841.4/
          DATA (X(13,J),J=1,2) /7.0, 871.4/
          DATA (X(14,J),J=1,2) /1.0, 577.3/
     !
          IRSP  = 2
          IND   = 1
          LOF   = 1
          CALL RFORP (X, IRSP, IND, MAXDEG, NDEG, A, B, SCOEF, D, LOF=LOF, &
                      SMULTC=SMULTC, SADDC=SADDC, DFE=DFE, SSE=SSE)
     !
          PRINT  = 'A'
          CALL RCASP (X, IRSP, IND, NDEG, SMULTC, SADDC, A, B, SCOEF, D, SSE, &
                      DFE, CASE, PRINT=PRINT)
     !
          END
```

Output

```
                          * * * Case Analysis * * *
     Obs.    Observed  Predicted   Residual   Leverage  Std. Res.  Jack. Res
              Cook's D     DFFITS   95.0% CI   95.0% CI   95.0% PI   95.0% PI
        1   508.1000   503.3459     4.7541     0.3554     0.7367     0.7204
            0.0997     0.5349   492.8003   513.8916   482.7510   523.9409
        2   787.6000   798.8150   -11.2150     0.1429    -1.5072    -1.6132
            0.1262    -0.6586   792.1288   805.5012   779.9034   817.7266
        3   498.4000   503.3459    -4.9460     0.3554    -0.7664    -0.7511
            0.1079    -0.5577   492.8003   513.8916   482.7510   523.9409
        4   568.2000   578.3177   -10.1177     0.1507    -1.3660    -1.4293
            0.1104    -0.6021   571.4498   585.1857   559.3412   597.2943
        5   651.7000   645.3505     6.3495     0.1535     0.8586     0.8476
            0.0446     0.3609   638.4200   652.2810   626.3513   664.3498
        6   854.7000   861.4297    -6.7297     0.3650    -1.0508    -1.0563
            0.2116    -0.8008   850.7420   872.1175   840.7617   882.0978
        7   657.0000   645.3505    11.6495     0.1535     1.5753     1.7069
            0.1500     0.7268   638.4200   652.2810   626.3513   664.3498
        8   755.3000   755.5992    -0.2992     0.1897    -0.0414    -0.0394
```

	0.0001	-0.0191	747.8945	763.3038	736.3040	774.8943
9	831.8000	834.0919	-2.2919	0.1429	-0.3080	-0.2949
	0.0053	-0.1204	827.4056	840.7782	815.1804	853.0035
10	758.9000	755.5992	3.3008	0.1897	0.4562	0.4392
	0.0162	0.2125	747.8945	763.3038	736.3040	774.8943
11	792.1000	798.8150	-6.7150	0.1429	-0.9024	-0.8942
	0.0452	-0.3650	792.1288	805.5012	779.9034	817.7266
12	841.4000	834.0919	7.3081	0.1429	0.9821	0.9804
	0.0536	0.4002	827.4056	840.7782	815.1804	853.0035
13	871.4000	861.4297	9.9703	0.3650	1.5567	1.6809
	0.4643	1.2745	850.7420	872.1175	840.7617	882.0978
14	577.3000	578.3177	-1.0178	0.1507	-0.1374	-0.1311
	0.0011	-0.0552	571.4498	585.1857	559.3412	597.2943

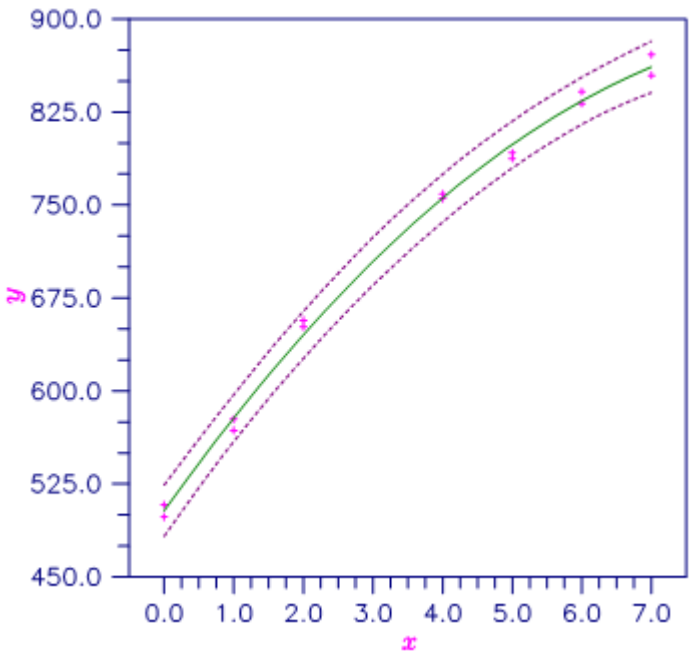


Figure 9, Second Degree Polynomial Fit With 95% One-at-a-Time Prediction Intervals

OPOLY

Generates orthogonal polynomials with respect to *x*-values and specified weights.

Required Arguments

X — Vector of length **N** containing the *x*-values. (Input)

NDEG — Degree of highest degree orthogonal polynomial to be generated. (Input)

SMULTC — Multiplicative constant used to compute a scaled version of *x* on the interval **-**2 to 2, inclusive. (Output)

SADDC — Additive constant used to compute a scaled version of *x* on the interval **-**2 to 2, inclusive. (Output)

SX — Vector of length **N** containing the scaled version of *x* on the interval **-**2 to 2, inclusive, computed as follows: **SX**(*i*) = **SMULTC * X**(*i*) + **SADDC** where *i* = 1, 2, …, **N**. (Output)
If **X** is not needed, **SX** and **X** can occupy the same storage locations.

A — Vector of length **NDEG** containing constants used to generate orthogonal polynomials. (Output)

B — Vector of length **NDEG** containing constants used to generate orthogonal polynomials. (Output)

POLY — Matrix, **N** by **NDEG**, containing the orthogonal polynomials evaluated at **SX**(*i*) for *i* = 1, 2, …, **N**. (Output)

Optional Arguments

N — Number of *x*-values. (Input)
Default: **N** = size (**POLY**,1).

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For **IWT** = 1, **WT** contains the weights.
Default: **IWT** = 0.

WT — Vector of length **N** containing the weights. (Input, if **IWT** = 1)
If **IWT** = 0, **WT** is not referenced and can be a vector of length one.

LDPOLY — Leading dimension of **POLY** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDPOLY** = size (**POLY**,1).

FORTRAN 90 Interface

Generic: CALL OPOLY (X, NDEG, SMULTC, SADDC, SX, A, B, POLY [, ...])
Specific: The specific interface names are S_OPOLY and D_OPOLY.

FORTRAN 77 Interface

Single: CALL OPOLY (N, X, IWT, WT, NDEG, SMULTC, SADDC, SX, A, B, POLY, LDPOLY)
Double: The double precision name is DOPOLY.

Description

Routine OPOLY generates orthogonal polynomials over a set of x_i's and with respect to weights w_i. The routine OPOLY is based on the algorithm of Forsythe (1957). (See also Kennedy and Gentle 1980.) A modification to Forsythe's algorithm is made for the inclusion of weights (Kelly 1967, page 68).

Let x_i be a value of the independent variable. The x_i's are scaled to the interval [-2, 2] for computational accuracy. The scaled version of the independent variable is computed by the formula $z_i = mx_i + c$. The multiplicative scaling constant m (stored in SMULTC) is

$$m = \frac{4}{\max_i(x_i) - \min_i(x_i)}$$

The additive constant c (stored in SADDC) is

$$c = \frac{2(\min_i(x_i) + \max_i(x_i))}{\min_i(x_i) - \max_i(x_i)}$$

Orthogonal polynomials are generated using the three-term recurrence relationship

$$p_j(z) = (z - a_j)p_{j-1}(z) - b_j p_{j-2}(z)$$

beginning with the initial polynomials

$$p_0(z) = 1 \quad \text{and} \quad p_1(z) = z - a_1$$

The a_j's and b_j's (stored in A and B) are computed to make the $p_j(z)$'s orthogonal, with respect to the the set of weights w_i, and over the set z_i.

Comments

1. Informational error

Type	Code	Description
3	8	N must be greater than NDEG in order for higher order polynomials to be nonzero. Columns N + 1 through NDEG of POLY are set to zero.

2. The orthogonal polynomials evaluated at each scaled **X** value are computed from **A** and **B** as follows:
   POLY(**I**, 1) = **SX**(**I**) **-** **A**(1)
   POLY(**I**, 2) = (**SX**(**I**) **-** **A**(2)) ***** **POLY**(**I**, 1) **-** **B**(2)
   POLY(**I**, **J**) = (**SX**(**I**) **-** **A**(**J**)) ***** **POLY**(**I**, **J** **-** 1) **-** **B**(**J**) ***** **POLY**(**I**, **J** **-** 2) for **J** = 3 through **NDEG**.

3. If **NDEG** is greater than 10, the accuracy of the results may be questionable.

Example

First-degree and second-degree orthogonal polynomials are generated using equally spaced *x* values 1, 2, ..., 12. (Equally spaced *x* values are not required by **OPOLY**.)

```
      USE OPOLY_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDPOLY, N, NDEG
      PARAMETER  (N=12, NDEG=2, LDPOLY=N)
!
      INTEGER    NOUT
      REAL       A(NDEG), B(NDEG), POLY(LDPOLY,NDEG), SADDC, SMULTC, &
                 SX(N), WT(1), X(N)
!
      DATA X/1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, &
           12.0/
!
      CALL OPOLY (X, NDEG, SMULTC, SADDC, SX, A, B, POLY)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) SMULTC, SADDC
99999 FORMAT (' SMULTC = ', F7.3, '  SADDC = ', F7.3)
      CALL WRRRN ('A', A, 1, NDEG, 1)
      CALL WRRRN ('B', B, 1, NDEG, 1)
      CALL WRRRN ('POLY', POLY)
      END
```

Output

```
 SMULTC =   0.364  SADDC =  -2.364

           A
         1           2
-5.960E-08  -1.009E-07

      B
    1       2
```

```
    0.000   1.576

           POLY
             1       2
 1  -2.000   2.424
 2  -1.636   1.102
 3  -1.273   0.044
 4  -0.909  -0.749
 5  -0.545  -1.278
 6  -0.182  -1.543
 7   0.182  -1.543
 8   0.545  -1.278
 9   0.909  -0.749
10   1.273   0.044
11   1.636   1.102
12   2.000   2.424
```

GCSCP

Generates centered variables, squares, and crossproducts.

Required Arguments

X — NRX by NVAR matrix containing the data. (Input)

XMEAN — Vector of length NVAR containing the means of the variables. (Input)

CSCP — NRX by NVAR * (NVAR + 3)/2 matrix containing the centered variables, squares, and crossproducts. (Output)

Columns	Description
1 to NVAR	Centered variables
NVAR+ 1 to 2 * NVAR	Squared variables
2 * NVAR + 1 to NVAR * (NVAR + 3)/2	Crossproducts

If X is not needed, X and the first NVAR columns of CSCP may occupy the same storage locations.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of GCSCP for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to GCSCP will be made. Initialization and updating for the data in X are performed.
2	This is an intermediate or final invocation of GCSCP and updating for the data in X is performed.

NRX — Number of rows of data in X. (Input)
Default: NRX = size (X,1).

NVAR — Number of variables. (Input)
Default: NVAR = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICEN — Centering option. (Input)
If IDO = 1 or IDO = 2, ICEN must equal 0.
Default: ICEN = 0.

ISUB	Action
0	CSCP contains the centered variables in columns 1 through NVAR. Square and crossproduct variables are generated from these centered variables in the remaining columns of CSCP.
1	First, the action taken when ICEN = 0 is performed. Next, the means of the square and crossproduct variables are subtracted from the square and crossproduct variables.

SCPM — Vector of length NVAR * (NVAR + 1)/2 containing the means of the generated square and crossproduct variables. (Output, if IDO = 0 or 1; input/output, if IDO = 2)

Elements	Description
1 to NVAR	Squared variable means
NVAR+ 1 to NVAR * (NVAR + 1)/2	Crossproduct variable means

LDCSCP — Leading dimension of CSCP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCSCP = size (CSCP,1).

NRMISS — Number of rows of data encountered in calls to GCSCP that contain any missing values for the variables. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2)
NaN (not a number) is used as the missing value code.
Default: NRMISS = 0.

NVOBS — Number of valid observations. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2)
Number of rows of data encountered in calls to GCSCP that do not contain any missing values for the variables.

FORTRAN 90 Interface

Generic: CALL GCSCP (X, XMEAN, CSCP [, ...])
Specific: The specific interface names are S_GCSCP and D_GCSCP.

FORTRAN 77 Interface

Single: CALL GCSCP (IDO, NRX, NVAR, X, LDX, ICEN, XMEAN, SCPM, CSCP, LDCSCP, NRMISS, NVOBS)

Double: The double precision name is DGCSCP.

Description

Routine GCSCP centers a data set consisting of independent variable settings and generates (using the centered variables) the settings for all possible squared and crossproduct variables in standard order. The routine GCSCP is designed so that you can partition a large data set into submatrices (requiring less space) and make multiple calls to GCSCP (with IDO = 1, 2, 2 ..., 2). Alternatively, one invocation of GCSCP (with IDO = 0) can be made with the entire data set contained in X.

Let *n* be the number of rows in the entire data set, and let *m* (stored in NVAR) be the number of variables. Let x_{ij} be the *i*-th setting of the *j*-th variable (*i* = 1, 2, ..., *n*; *j* = 1, 2, ..., *m*). Denote the means (stored in XMEAN) by

$$\overline{x}_j \left(j = 1, 2, \ \ldots, m \right)$$

The settings of the *j*-th centered variable (stored in the *j*-th column of CSCP) are given by

$$z_{ij} = x_{ij} - \overline{x}_j$$

The settings of the *j*-th squared variable (stored in the (*m* + *j*)-th column of CSCP) are given by

$$\begin{cases} z_{ij}^2 & \text{if ICEN} = 0 \\ z_{ij}^2 - \overline{z_j^2} & \text{if ICEN} = 1 \end{cases}$$

where

$$\overline{z_j^2} = \sum_{i=1}^{n} \frac{z_{ij}^2}{n}$$

(stored in the (*m* + *j*)-th column of SCPM) is the mean of the *j*-th squared variable. The settings of the *jk* crossproduct variable (stored in the

$$k - j + mj - \frac{j\left(j-1\right)}{2}$$

column of CSCP) are given by

$$\begin{cases} z_{ij}z_{ik} & \text{if ICEN} = 0 \\ z_{ij}z_{ik} - \overline{z_j z_k} & \text{if ICEN} = 1 \end{cases}$$

where

$$\overline{z_j z_k} = \sum_{i=1}^{n} \frac{z_{ij}z_{ik}}{n}$$

(stored in the

$$k - j + mj - \frac{j(j-1)}{2}$$

location of **SCPM**) is the mean of the *jk*-th (*j* < *k*) crossproduct variable.

Comments

Crossproduct variables are ordered as follows: (1, 2), (1, 3), …, (1, **NVAR**), (2, 3), (2, 4), …, (2, **NVAR**), …, (**NVAR** - 1, **NVAR**).

Examples

Example 1

With data containing 4 rows and 3 variables, **GCSCP** is used to center the variables and to generate (using the centered variables) the square and crossproduct variables. The data is input in one invocation (**IDO** = 0), and the generated squared and crossproduct variables are centered (**ICEN** = 1). On output, **SCPM** contains the means in standard order, i.e.,

$$\overline{z_1^2}, \overline{z_2^2}, \overline{z_3^2}, \overline{z_1 z_2}, \overline{z_1 z_3}, \overline{z_2 z_3}$$

Also, **CSCP** contains the variables in standard order, i.e.,

$$z_1, z_2, z_3, z_1^2 - \overline{z_1^2}, z_2^2 - \overline{z_2^2}, z_3^2 - \overline{z_3^2}, z_1 z_2 - \overline{z_1 z_2}, z_1 z_3 - \overline{z_1 z_3}, z_2 z_3 - \overline{z_2 z_3}$$

```
      USE GCSCP_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDCSCP, LDX, NRX, NVAR, J, ICEN
```

```
      PARAMETER  (NRX=4, NVAR=3, LDCSCP=NRX, LDX=NRX)
!
      INTEGER    NOUT, NRMISS, NVOBS
      REAL       CSCP(LDCSCP,NVAR*(NVAR+3)/2), SCPM(NVAR*(NVAR+1)/2), &
                 X(LDX,NVAR), XMEAN(NVAR)
!
      DATA (X(1,J),J=1,NVAR)/10.0,  8.0, 11.0/
      DATA (X(2,J),J=1,NVAR)/ 5.0, 15.0,  1.0/
      DATA (X(3,J),J=1,NVAR)/ 3.0,  2.0,  4.0/
      DATA (X(4,J),J=1,NVAR)/ 6.0,  3.0,  4.0/
      DATA XMEAN/6.0, 7.0, 5.0/
!
      ICEN = 1
      CALL GCSCP (X, XMEAN, CSCP, ICEN=ICEN, scpm=scpm, nrmiss=nrmiss, &
                  nvobs=nvobs)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' NRMISS = ', NRMISS
      CALL WRRRN ('SCPM', SCPM, 1, NVAR*(NVAR+1)/2, 1)
      CALL WRRRN ('CSCP', CSCP)
      END
```

Output

```
 NRMISS =   0

                  SCPM
    1       2       3       4       5
6.50   26.50   13.50    2.75    7.75   -4.25

                                    CSCP
        1       2       3       4       5       6       7       8       9
1    4.00    1.00    6.00    9.50  -25.50   22.50    1.25   16.25   10.25
2   -1.00    8.00   -4.00   -5.50   37.50    2.50  -10.75   -3.75  -27.75
3   -3.00   -5.00   -1.00    2.50   -1.50  -12.50   12.25   -4.75    9.25
4    0.00   -4.00   -1.00   -6.50  -10.50  -12.50   -2.75   -7.75    8.25
```

Example 2

With data containing 4 rows and 3 variables, `GCSCP` is used to center the variables and to generate (using the centered variables) the square and crossproduct variables. The data is input in multiple invocations (`IDO` = 1, 2, 2, 2). Here, the square and crossproduct variables, generated using the centered variables, cannot be centered (`ICEN` = 0).

```
      USE GCSCP_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDCSCP, LDX, NRX, NVAR, J
      PARAMETER  (LDX=4, NRX=1, NVAR=3, LDCSCP=NRX)
!
      INTEGER    I, IDO, MISS, NOUT, NRMISS, NVOBS
      REAL       CSCP(LDCSCP,NVAR*(NVAR+3)/2), SCPM(NVAR*(NVAR+1)/2), &
                 X(LDX,NVAR), XMEAN(NVAR)
!
```

```
      DATA (X(1,J),J=1,NVAR)/10.0,  8.0, 11.0/
      DATA (X(2,J),J=1,NVAR)/ 5.0, 15.0,  1.0/
      DATA (X(3,J),J=1,NVAR)/ 3.0,  2.0,  4.0/
      DATA (X(4,J),J=1,NVAR)/ 6.0,  3.0,  4.0/
      DATA XMEAN/6.0, 7.0, 5.0/
!
      CALL UMACH (2, NOUT)
      MISS = 0
      DO 10  I=1, 4
         IF (I .EQ. 1) THEN
            IDO = 1
         ELSE
            IDO = 2
         END IF
         CALL GCSCP (X(I:,1:), XMEAN, CSCP, IDO=IDO, NRX=NRX, scpm=scpm,  &
                     nrmiss=nrmiss, nvobs=nvobs)
         MISS = MISS + NRMISS
         CALL WRRRN ('CSCP', CSCP)
   10 CONTINUE
      CALL WRRRN ('SCPM', SCPM, 1, NVAR*(NVAR+1)/2, 1)
      WRITE (NOUT,*) ' MISS = ', MISS
      END
```

Output

```
                                  CSCP
      1       2       3       4       5       6       7       8       9
   4.00    1.00    6.00   16.00    1.00   36.00    4.00   24.00    6.00

                                  CSCP
      1       2       3       4       5       6       7       8       9
  -1.00    8.00   -4.00    1.00   64.00   16.00   -8.00    4.00  -32.00

                                  CSCP
      1       2       3       4       5       6       7       8       9
  -3.00   -5.00   -1.00    9.00   25.00    1.00   15.00    3.00    5.00

                                  CSCP
      1       2       3       4       5       6       7       8       9
   0.00   -4.00   -1.00    0.00   16.00    1.00    0.00    0.00    4.00

                         SCPM
      1       2       3       4       5       6
   6.50   26.50   13.50    2.75    7.75   -4.25
  MISS =   0
```

TCSCP

Transforms coefficients from a second order response surface model generated from squares and crossproducts of centered variables to a model using uncentered variables.

Required Arguments

XMEAN — Vector of length NVAR containing the means of the variables. (Input)

SCPM — Vector of length NVAR(NVAR + 1)/2 containing the means of the generated square and crossproduct variables. (Input)

Elements	Description
1 to NVAR	Squared variable means
NVAR+ 1 to NVAR * (NVAR + 1)/2	Crossproduct variable means

BC — Vector of length NVAR * (NVAR + 3)/2 + 1 containing the coefficients for the centered variables. (Input)
Here, the fitted model is

$$\hat{y} = \mathrm{BC}(1) + \sum_{j=1}^{\mathrm{NVAR}} \mathrm{BC}(j) * z_j + \sum_{j=1}^{\mathrm{NVAR}} \mathrm{BC}(j) * \left(z_j^2 - \mathrm{SCPM}(j)\right)$$
$$+ \sum_{j=1}^{\mathrm{NVAR}} \sum_{k=j+1}^{\mathrm{NVAR}} \mathrm{B}(1 + \mathrm{NVAR} + m) * \left(z_j z_k - \mathrm{SCPM}(m(j,k))\right)$$

where $z_j = x_j$ - XMEAN(j) and $m_{jk} = j$ * NVAR - $j(j$ - 1)/2 + k - j. These regression coefficients can come from a regression using variables generated by routine GCSCP with the option ICEN = 1.

B — Vector of length NVAR * (NVAR + 3)/2 + 1 containing the coefficients of the uncentered variables. (Output)
Here, the model uses the original x variables, i.e.,

$$\hat{y} = \mathrm{B}(1) + \sum_{j=1}^{\mathrm{NVAR}} \mathrm{B}(j) * x_j + \sum_{j=1}^{\mathrm{NVAR}} \mathrm{B}(j) * x_j^2$$
$$+ \sum_{j=1}^{\mathrm{NVAR}} \sum_{k=j+1}^{\mathrm{NVAR}} \mathrm{B}(1 + \mathrm{NVAR} + m) * x_j x_k$$

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (XMEAN,1).

FORTRAN 90 Interface

Generic: CALL TCSCP (XMEAN, SCPM, BC, B [, ...])
Specific: The specific interface names are S_TCSCP and D_TCSCP.

FORTRAN 77 Interface

Single: CALL TCSCP (NVAR, XMEAN, SCPM, BC, B)
Double: The double precision name is DTCSCP.

Description

Routine TCSCP transforms coefficients from a second-order response surface model fitted using squares and crossproducts of centered variables into a model using the original uncentered variables. Let x_{ij} be the *i*-th setting of the *j*-th variable (*i* = 1, 2, ..., *n*; *j* = 1, 2, ..., *m*). Denote the means (stored in XMEAN) by

$$\overline{x_j}\left(j = 1,\ 2,\ \ \ldots\ ,\ m\right)$$

The settings of the *j*-th centered variable are given by

$$z_{ij} = x_{ij} - \overline{x_j}$$

The settings of the *j*-th squared variable are given by

$$z_{ij}^2 - \overline{z_j^2}$$

where

$$\overline{z_j^2} = \sum_{i=1}^{n} \frac{z_{ij}^2}{n}$$

(stored in (m + j)-th column of **SCPM**) is the mean of the j-th squared variable. The settings of the jk crossproduct variable are given by

$$z_{ij}z_{ik} - \overline{z_j z_k}$$

where

$$\overline{z_j z_k} = \sum_{i=1}^{n} \frac{z_{ij}z_{ik}}{n}$$

(stored in the

$$k - j + mj - \frac{j(j-1)}{2}$$

location of **SCPM**) is the mean of the jk-th (j < k) crossproduct variable. The fitted model is

$$\hat{y}_i = \hat{\alpha}_0 + \sum_{j=1}^{m} \hat{\alpha}_j z_{ij} + \sum_{j=1}^{m} \hat{\alpha}_{jj}\left(z_{ij}^2 - \overline{z_j^2}\right) + \sum_{j=2}^{m-1}\sum_{k=j+1}^{m} \hat{\alpha}_{jk}\left(z_{ij}z_{ik} - \overline{z_j z_k}\right)$$

TCSCP transforms the

$$\hat{\alpha}_j\text{'s, the } \hat{\alpha}_{jj}\text{'s, and the } \hat{\alpha}_{jk}\text{'s}$$

to regression coefficients for the original independent variables. The fitted transformed model is

$$\hat{y}_i = \hat{\beta}_0 + \sum_{j=1}^{m} \hat{\beta}_j x_{ij} + \sum_{j=1}^{m} \hat{\beta}_{jj} x_{ij}^2 + \sum_{j=2}^{m-1}\sum_{k=j+1}^{m} \hat{\beta}_{jk} x_{ij} x_{ik}$$

where

$$\hat{\beta}_0 = \hat{\alpha}_0 - \sum_{j=1}^{m} \hat{\alpha}_j \overline{x_j} - \sum_{j=1}^{m} \hat{\alpha}_{jj} \overline{z_j^2} - \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} \hat{\alpha}_{jk} \overline{z_j z_k} + \sum_{j=1}^{m} \hat{\alpha}_{jj} \overline{x_j^2} + \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} \hat{\alpha}_{jk} \overline{x_j x_k}$$

$$\hat{\beta}_j = \hat{\alpha}_j - 2\hat{\alpha}_{jj}\overline{x_j} - \sum_{k=1}^{m-1} \hat{\alpha}_{kj} \overline{x_k} \sum_{k=j+1}^{m} \hat{\alpha}_{jk} \overline{x_k}$$

$$\hat{\beta}_{jj} = \hat{\alpha}_{jj}$$

$$\hat{\beta}_{jk} = \hat{\alpha}_{jk}$$

Comments

Crossproduct variables are ordered as follows: (1, 2), (1, 3), …, (1, NVAR), (2, 3), (2, 4), …, (2, NVAR), …, (NVAR - 1, NVAR).

Example

This example transforms coefficients from a second-order response surface model with three independent variables fitted using squares and crossproducts of centered variables into a model using the original uncentered variables.

```
      USE TCSCP_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    NVAR
      PARAMETER  (NVAR=3)
!
      REAL       B(NVAR*(NVAR+3)/2+1), BC(NVAR*(NVAR+3)/2+1), &
                 SCPM(NVAR*(NVAR+1)/2), XMEAN(NVAR)
!
      DATA XMEAN/10.0, 11.0, 6.0/
      DATA SCPM/12.0, 5.0, 2.0, 3.0, 7.0, 1.0/
      DATA BC/1.0, 2.0, 3.0, 0.0, 5.0, 0.0, 7.0, 0.0, 9.0, 10.0/
!
      CALL TCSCP (XMEAN, SCPM, BC, B)
!
      CALL WRRRN ('B', B, 1, NVAR*(NVAR+3)/2+1, 1)
!
      END
```

Output

```
                                    B
```

```
        1         2         3         4         5         6         7         8
   1753.0    -152.0     -57.0    -284.0       5.0       0.0       7.0       0.0

        9        10
      9.0      10.0
```

RNLIN

Fits a nonlinear regression model.

Required Arguments

FUNC — User-supplied SUBROUTINE to return the weight, frequency, residual, and optionally the derivative of the residual at the given parameter vector THETA for a single observation. The usage is:

CALL FUNC (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, IEND),

where

NPARM – Number of unknown parameters in the regression function. (Input)

THETA – Vector of length NPARM containing parameter values. (Input)

IOPT – Function/derivative evaluation option. (Input)

IOPT	Meaning
0	Evaluate the function.
1	Evaluate the derivative

If IDERIV = 0, only IOPT = 0 is used.

IOBS – Observation number. (Input)
The function is evaluated at the IOBS-th observation.

FRQ – Frequency for the observation. (Output)

WT – Weight for the observation. (Output)
Use WT = 1.0 for equal weighting (unweighted least squares).

E – Error (residual) for the IOBS-th observation. (Output, if IOPT = 0)

DE – Vector of length NPARM containing the partial derivatives of the residual for the IOBS-th observation. (Output, if IOPT = 1)
If IDERIV = 0, DE is not referenced and can be a vector of length one.

IEND – Completion indicator. (Output)

IEND	Meaning
0	IOBS is less than or equal to the number of observations.
1	IOBS is greater than the number of observations. WT, FRQ, E, and DE are not output.

FUNC must be declared EXTERNAL in the calling program.

THETA — Vector of length NPARM containing parameter values. (Input/Output)
On input, THETA must contain the initial estimate. On output, THETA contains the final estimate.

Optional Arguments

NPARM — Number of unknown parameters in the regression function. (Input)
Default: NPARM = size (THETA,1).

IDERIV — Derivative option. (Input)
Default: IDERIV = 0.

IDERIV	Meaning
0	Derivatives are obtained by finite differences.
1	Derivatives are supplied by FUNC.

R — NPARM by NPARM upper triangular matrix containing the *R* matrix from a *QR* decomposition of the Jacobian. (Output)

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R, 1).

IRANK — Rank of *R*. (Output)
IRANK less than NPARM may indicate the model is overparameterized.

DFE — Degrees of freedom for error. (Output)

SSE — Sums of squares for error. (Output)

FORTRAN 90 Interface

Generic: CALL RNLIN (FUNC, THETA [, ...])
Specific: The specific interface names are S_RNLIN and D_RNLIN.

FORTRAN 77 Interface

Single: CALL RNLIN (FUNC, NPARM, IDERIV, THETA, R, LDR, IRANK, DFE, SSE)
Double: The double precision name is DRNLIN.

Description

Routine RNLIN fits a nonlinear regression model using least squares. The nonlinear regression model is

$$y_i = f(x_i;\theta) + \varepsilon_i \quad i = 1, 2, \ldots, n$$

where the observed values of the y_i's constitute the responses or values of the dependent variable, the known x_i's are the vectors of the values of the independent (explanatory) variables, θ is the vector of p regression parameters, and the ε_i's are independently distributed normal errors with mean zero and variance σ^2. For this model, a least squares estimate of θ is also a maximum likelihood estimate of θ.

The residuals for the model are

$$e_i(\theta) = y_i - f(x_i;\theta) \quad i = 1, 2, \ldots, n$$

A value of θ that minimizes

$$\sum_{i=1}^{n} [\, e_i(\theta) \,]^2$$

is a least squares estimate of θ. Routine RNLIN is designed so that these residuals are input one at a time from a user-supplied subroutine. This permits RNLIN to handle the case when n is large and the data cannot reside in an array but must reside on some secondary storage device.

Routine RNLIN is based on MINPACK routines LMDIF and LMDER by Moré et al. (1980). The routine RNLIN uses a modified Levenberg-Marquardt method to generate a sequence of approximations to a minimum point. Let

$$\hat{\theta}_c$$

be the current estimate of θ. A new estimate is given by

$$\hat{\theta}_c + s_c$$

where s_c is a solution to

$$\left(J(\hat{\theta}_c)^T J(\hat{\theta}_c) + \mu_c I\right) s_c = J(\hat{\theta}_c)^T e(\hat{\theta}_c)$$

Here,

$$J\left(\hat{\theta}_c\right)$$

is the Jacobian evaluated at

$$\hat{\theta}_c$$

The algorithm uses a "trust region" approach with a step bound of δ_c. A solution of the equations is first obtained for $\mu_c = 0$. If $\|s_c\|_2 < \delta_c$, this update is accepted. Otherwise, μ_c is set to a positive value and another solution is obtained. The method is discussed by Levenberg (1944), Marquardt (1963), and Dennis and Schnabel (1983, pages 129-147, 218-338).

If **IDERIV** = 0, forward finite differences are used to estimate the Jacobian numerically. If **IDERIV** = 1, the Jacobian is computed analytically via the user-supplied subroutine. With **IDERIV** = 0 and single precision arithmetic, the estimate of the Jacobian may be so poor that the algorithm terminates at a noncritical point. In such instances, **IDERIV** = 1 or double precision arithmetic is recommended.

Routine **RNLIN** does not actually store the Jacobian but uses fast Givens transformations to construct an orthogonal reduction of the Jacobian to upper triangular form (stored in **R**). The reduction is based on fast Givens transformations (see routines **SROTMG** and **SROTM**, Golub and Van Loan 1983, pages 156-162, Gentleman 1974). This method has two main advantages: (1) the loss of accuracy resulting from forming the crossproduct matrix used in the equations for s_c is avoided, and (2) the $n \times p$ Jacobian need not be stored saving space when $n > p$.

A weighted least squares fit can also be performed. This is appropriate when the variance of ε_i in the nonlinear regression model is not constant but instead is σ^2/w_i. Here, the w_i's are weights input via the user-supplied subroutine. For the weighted case, **RNLIN** computes a minimum weighted sum of squares for error (stored in SSE).

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2LIN/DR2LIN**. The reference is:

   CALL R2LIN (FUNC, NPARM, IDERIV, THETA, R, LDR, IRANK, DFE, SSE, IPARAM, RPARAM, SCALE, IWK, WK)

   The additional arguments are as follows:

IPARAM — Vector of length 6 containing convergence parameters. (Input/Output)
On input, set IPARAM(1) = 0 for default convergence parameter settings. If IPARAM(1) = 0, the remaining elements of IPARAM, and the arguments RPARAM and SCALE need not be initialized.

I	Name	IPARAM(I)
1	INIT	Initialization flag. (Input) INIT = 0 means use default settings for IPARAM, RPARAM, and SCALE. INIT = 1 means use the input IPARAM and RPARAM settings.
2	NDIGIT	Number of good digits in the residuals. (Input, if IPARAM(1) = 1)
3	ITER	Number of iterations. (Input/Output, if IPARAM(1) = 1; Output, otherwise) On input, this is the maximum number of iterations allowed. The default is 100. On output, it is the actual number of iterations.
4	NFCN	Number of SSE evaluations. (Input/Output, if IPARAM(1) = 1; Output, if IPARAM(1) = 0) On input, this is the maximum number of evaluations allowed. The default is 400. On output, it is the actual number of evaluations.
5	NJAC	Number of Jacobian evaluations. (Input, if IPARAM(1) = 1 and IDERIV = 1; Output, if IDERIV = 1) On input, this is the maximum number of Jacobian evaluations allowed. The default is 100. On output, it is the number of Jacobian evaluations.
6	MODE	Scaling option. (Input, if IPARAM(1) = 1) If IPARAM(6) = 1, the values for SCALE are set internally. The default is 1. Otherwise, SCALE must be input.

RPARAM — Vector of length 7 containing convergence parameters. (Input, if IPARAM(1) = 1) In the following table, the default settings are given in parentheses. EPS = AMACH(4) (see the documentation for IMSL routine AMACH).

I	Name	RPARAM(I)
1	FJACTL	Scaled gradient tolerance (SQRT(EPS) for single precision; $EPS^{1/3}$ for double precision) Convergence is declared if |WK(I)| * max{|THETA(I)|, 1.0/SCALE(I)}/SSE is less than FJACTL for I = 1, 2, …, NPARM, where WK(I) is the I-th component of the gradient vector.
2	STEPTL	Scaled step tolerance ($EPS^{2/3}$) Convergence is declared if |WK(NPARM + I)|/max{|THETA(I)|, 1.0/SCALE(I)} is less than STEPTL for I = 1, 2, …, NPARM, where WK(NPARM + I) is the I-th component of the last step.
3	RFTOL	Relative function tolerance (max{10^{-10} $EPS^{2/3}$} for single precision, max{10^{-20}, $EPS^{2/3}$} for double precision) Convergence is declared if the change in SSE is less than or equal to RFTOL * SSE in absolute value.
4	AFTOL	Absolute function tolerance (max{10^{-20}, EPS^2} for single precision; max{10^{-40}, EPS^2} for double precision) Convergence is declared if SSE is less than AFTOL.
5	FALSTL	False convergence tolerance (100.0 * EPS)
6	STEPMX	Maximum allowable step size (1000 * MAX(TOL1, TOL2) where TOL1 = SNRM2(NPARM, SCXTH, 1); TOL2 = SNRM2(NPARM, SCALE, 1) and SCXTH is the elementwise product of SCALE and THETA, i.e., SCXTH(I) = SCALE(I) * THETA(I).)
7	DELTA	Size of initial trust region radius (based on the initial scaled Cauchy step)

SCALE — Vector of length NPARM. (Input/Output, if IPARAM(1) = 1 and IPARAM(6) = 0; Output, if IPARAM(6) = 1)
A common choice is to set all elements of SCALE to 1.0. If good starting values for THETA are known and nonzero, a good choice is SCALE(I) = 1.0/|THETA(I)|. Otherwise, for example, if THETA(I) is known to be in the interval (-10^5, 10^5), set SCALE(I) = 10^{-5}. Or, for example, if THETA(I) is known to be in the interval (10^3, 10^5), set SCALE(I) = 10^{-4}.

IWK — Work vector of length NPARM.

WK — Work vector of length 11 * NPARM + 4. (Output)
The first NPARM components of WK are the gradient at the solution. The second NRARM components of WK give the last step.

2. Informational errors

Type	Code	Description
3	1	Both the scaled actual and predicted reductions in the function are less than or equal to the relative function convergence tolerance.
3	2	The iterates appear to be converging to a noncritical point. Incorrect gradient information, a discontinuous function, or stopping tolerances being too tight may be the cause.
4	3	Maximum number of iterations is exceeded.
4	4	Maximum number of function evaluations is exceeded.
4	5	Maximum number of Jacobian evaluations is exceeded for `IDERIV` = 1.
3	6	Five consecutive steps of the same size have been taken. Either the function is unbounded below, or it has a finite asymptote in some direction, or the stepsize is too small.
2	7	Scaled step tolerance is satisfied, the current point may be an approximate local solution, or the algorithm is making very slow progress and is not near a solution, or `STEPTL` is too big.

3. The first stopping criterion for `RNLIN` occurs when `SSE` is less than the absolute function tolerance. The second stopping criterion occurs when the norm of the scaled gradient is less than the given gradient tolerance. The third stopping criterion occurs when the scaled distance between the last two steps is less than the step tolerance. The third stopping criterion also generates error code 7. The fourth stopping criterion occurs when the relative change in `SSE` is less than `RFTOL`. The fourth stopping criterion also generates error code 1. See Dennis and Schnabel (1983, pages 159-161, 278-280, and 347-348) for a discussion of stopping criteria and choice of tolerances.

4. To use some nondefault convergence parameters, first call `R8LIN`, then reset the corresponding convergence parameters to the desired value and call `R2LIN`. For example, the following code could be used if nondefault convergence parameters are to be used:

```
!
   CALL R8LIN (IPARAM, RPARAM)
!  R8LIN outputs IPARAM(1) = 1 to indicate some
!  nondefault convergence parameters are to be set.
!  R8LIN outputs the remaining elements of IPARAM
!  and RPARAM as their default values.
!
!  Set some nondefault convergence parameters.
   IPARAM(3) = 20
   IPARAM(6) = 0
   SCALE(1) = 0.1
   SCALE(2) = 10.0
```

```
!
   CALL R2LIN (FUNC, NPARM, IDERIV, THETA, R, &
        LDR,IRANK, DFE, SSE, IPARAM, &
        RPARAM, SCALE, IWK, WK)
```

If double precision is being used, then **DR8LIN** and **DR2LIN** are called and **RPARAM** is declared double precision.

Programming Notes

Nonlinear regression allows substantial flexibility over linear regression because the user can specify the functional form of the model. This added flexibility can cause unexpected convergence problems for users that are unaware of the limitations of the software. Also, in many cases, there are possible remedies that may not be immediatedly obvious. The following is a list of possible convergence problems and some remedies that the user can try. There is not a one-to-one correspondence between the problems and the remedies. Remedies for some problems may also be relevant for the other problems.

1. A local minimum is found. Try a different starting value. Good starting values often can be obtained by fitting simpler models. For example, for a nonlinear function

$$f(x;\theta) = \theta_1 e^{\theta_2 x}$$

good starting values can be obtained from the estimated linear regression coefficients

$$\hat{\beta}_0 \text{ and } \hat{\beta}_1$$

from a simple linear regression of ln y on ln x. The starting values for the nonlinear regression in this case would be

$$\theta_1 = e^{\hat{\beta}_0} \text{ and } \theta_2 = \hat{\beta}_1$$

If an approximate linear model is not clear, then simplify the model by reducing the number of nonlinear regression parameters. For example, some nonlinear parameters for which good starting values are known could be set to these values in order to simplify the model for computing starting values for the remaining parameters.

2. The estimate of **θ** is incorrectly returned as the same or very close to the initial estimate

   - The scale of the problem may be orders of magnitude smaller than the assumed default of 1 causing premature stopping. For example, in single precision if **SSE** is less than **AMACH**(4)**2, the routine stops. See Example 3, which shows how to shut down some of the

stopping criteria that may not be relevant for your particular problem and which also shows how to improve the speed of convergence by the input of the scale of the model parameters.

- The scale of the problem may be orders of magnitude larger than the assumed default causing premature stopping. The information with regard to the input of the scale of the model parameters in Example 3 is also relevant here. In addition, the maximum allowable step size, **RPARAM**(6) in Example 3, may need to be increased.
- The residuals are input with accuracy much less than machine accuracy causing premature stopping because a local minimum is found. Again see Example 3 to see generally how to change some default tolerances. If you cannot improve the precision of the computations of the residual, you need to set **IPARAM**(2) to indicate the actual number of good digits in the residuals.

3. The model is discontinuous as a function of **θ**. You may have a mistake in the subroutine you supplied. Note that the function *f*(*x*; **θ**) can be a discontinuous function of *x*.
4. The *R* matrix returned by **RNLIN** is inaccurate. Use the double precision version **DRNLIN**. If **IDERIV** = 1, check your derivatives or try using **IDERIV** = 0. If **IDERIV** = 0, try using **IDERIV** = 1.
5. Overflow occurs during the computations. Print out **θ** and the residual in the subroutine you supplied. Make sure the code you supply does not overflow at some value of **θ**.
6. The estimate of **θ** is going to infinity. You may need to reparameterize or change your function. For example, a parameterization in terms of the reciprocals may be appropriate.
7. Some components of **θ** are outside known bounds. Routine **RNLIN** does not handle bounds on the parameters, but you can artificially impose some by setting the residuals unusually large outside the bounds. Although this introduces a discontinuity in the model function, this often works and allows you to use **RNLIN** without having to resort to a more general nonlinear optimization routine.

Examples

Example 1

This example uses data discussed by Neter, Wasserman, and Kutner (1983, pages 475-478). A nonlinear model

$$y_i = \theta_1 e^{\theta_2 x_i} + \varepsilon_i \quad i = 1, 2, \ldots, 15$$

is fitted. The option **IDERIV** = 0 is used.

The user must supply a SUBROUTINE to return the residual, weight, and frequency for a single observation at the given value of the regression parameter vector θ. This subroutine, called EXAMPL here, must be declared EXTERNAL in the calling program and must have the specified calling sequence.

```
      USE RNLIN_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDR, NOBS, NPARM
      PARAMETER  (NOBS=15, NPARM=2, LDR=NPARM)
!
      INTEGER    IRANK, NOUT
      REAL       DFE, R(LDR,NPARM), SSE, THETA(NPARM)
      EXTERNAL   EXAMPL
!
      DATA THETA/60.0, -0.03/
!
      CALL UMACH (2, NOUT)
!
      CALL RNLIN (EXAMPL, THETA, r=r, irank=irank, dfe=dfe, sse=sse)
      WRITE (NOUT,*) 'THETA = ', THETA
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  DFE = ', DFE, '  SSE = ', &
                     SSE
      CALL WRRRN ('R', R)
      END
!
      SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, &
                         IEND)
      INTEGER    NPARM, IOPT, IOBS, IEND
      REAL       THETA(NPARM), FRQ, WT, E, DE(1)
!
      INTEGER    NOBS
      PARAMETER  (NOBS=15)
!
      REAL       EXP, XDATA(NOBS), YDATA(NOBS)
      INTRINSIC  EXP
!
      DATA YDATA/54.0, 50.0, 45.0, 37.0, 35.0, 25.0, 20.0, 16.0, 18.0, &
           13.0, 8.0, 11.0, 8.0, 4.0, 6.0/
      DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
           38.0, 45.0, 52.0, 53.0, 60.0, 65.0/
!
      IF (IOBS .LE. NOBS) THEN
         WT   = 1.0E0
         FRQ  = 1.0E0
         IEND = 0
         E    = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
      ELSE
         IEND = 1
      END IF
      RETURN
      END
```

Output

```
 THETA =     58.6045   -3.95835E-02
```

```
 IRANK =   2  DFE =     13.0000  SSE =     49.4593

           R
           1        2
1      1.9   1139.8
2      0.0   1139.7
```

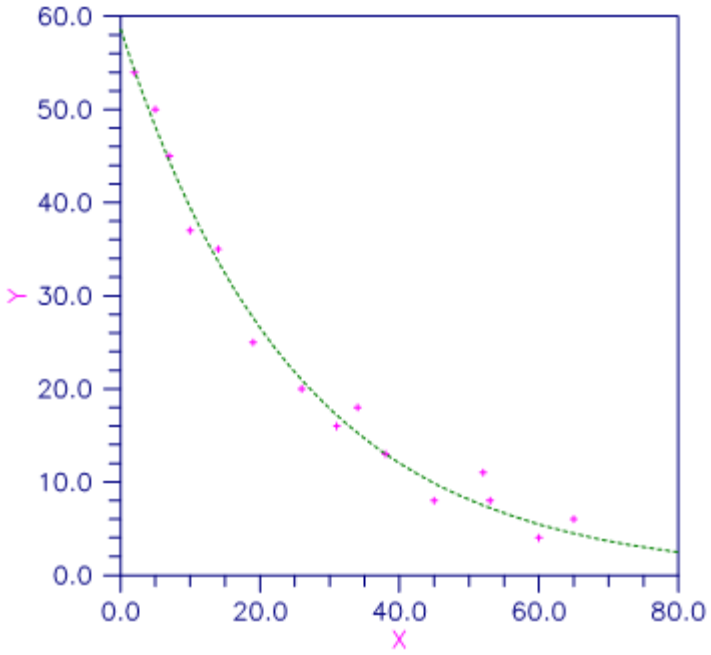


Figure 10, Plot of the Nonlinear Fit

Example 2

This example fits the model in Example 1 with the option **IDERIV** = 1.

```
      USE RNLIN_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDR, NOBS, NPARM
      PARAMETER  (NOBS=15, NPARM=2, LDR=NPARM)
!
      INTEGER    IDERIV, IRANK, NOUT
      REAL       DFE, R(LDR,NPARM), SSE, THETA(NPARM)
      EXTERNAL   EXAMPL
!
      DATA THETA/60.0, -0.03/
!
      CALL UMACH (2, NOUT)
!
```

```
      IDERIV = 1
      CALL RNLIN (EXAMPL, THETA, IDERIV=IDERIV, r=r, irank=irank, dfe=dfe, &
                  sse=sse)
      WRITE (NOUT,*) 'THETA = ', THETA
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  DFE = ', DFE, '  SSE = ', &
                     SSE
      CALL WRRRN ('R', R)
      END
!
      SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, &
                         IEND)
      INTEGER    NPARM, IOPT, IOBS, IEND
      REAL       THETA(NPARM), FRQ, WT, E, DE(NPARM)
!
      INTEGER    NOBS
      PARAMETER  (NOBS=15)
!
      REAL       EXP, XDATA(NOBS), YDATA(NOBS)
      INTRINSIC  EXP
!
      DATA YDATA/54.0, 50.0, 45.0, 37.0, 35.0, 25.0, 20.0, 16.0, 18.0, &
           13.0, 8.0, 11.0, 8.0, 4.0, 6.0/
      DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
           38.0, 45.0, 52.0, 53.0, 60.0, 65.0/
!
      IF (IOBS .LE. NOBS) THEN
         WT   = 1.0E0
         FRQ  = 1.0E0
         IEND = 0
         IF (IOPT .EQ. 0) THEN
            E = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
         ELSE
            DE(1) = -EXP(THETA(2)*XDATA(IOBS))
            DE(2) = -THETA(1)*XDATA(IOBS)*EXP(THETA(2)*XDATA(IOBS))
         END IF
      ELSE
         IEND = 1
      END IF
      RETURN
      END
```

Output

```
 THETA =     58.6034   -3.95812E-02
 IRANK =   2  DFE =     13.0000  SSE =     49.4593

          R
          1        2
1      1.9   1140.1
2      0.0   1139.9
```

Example 3

This example fits the model in Example 1, but the data for y is 10^{-10} times the values in Example 1. In order to solve this problem without rescaling y, we use some nondefault convergence tolerances and scales. This is accomplished by invoking routine **R8LIN**, setting some elements of **IPARAM**, **RPARAM**, and **SCALE**, and then

invoking R2LIN. Here, we set the absolute function tolerance to 0.0. The default value would cause the program to terminate after one iteration because the residual sum of squares is roughly 10^{-19} Also, we set the relative function tolerance to 0.0. The gradient stopping condition is properly scaled for this problem so we leave it at its default value. Finally, we set SCALE(I) equal to the absolute value of the reciprocal of the starting value.

Note in the output that the estimate of θ_1 is 10^{-10} times the estimate in Example 1. Note also that the invocation of R2LIN in place of RNLIN allows the printing of additional information that is output in IPARAM (number iterations and number of SSE evaluations) and output in WK (gradient at solution and last step).

```
      USE UMACH_INT
      USE R8LIN_INT
      USE R2LIN_INT
      USE WROPT_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDR, NOBS, NPARM, IDERIV, ISETNG
      PARAMETER  (NOBS=15, NPARM=2, LDR=NPARM)
!
      INTEGER    IPARAM(6), IRANK, IWK(NPARM), NOUT
      REAL       ABS, DFE, R(LDR,NPARM), RPARAM(7), SCALE(NPARM), SSE, &
                 THETA(NPARM), WK(11*NPARM+4)
      INTRINSIC  ABS
      EXTERNAL   EXAMPL
!
      DATA THETA/6.0E-9, -0.03/
!
      CALL UMACH (2, NOUT)
!
      CALL R8LIN (IPARAM, RPARAM)
      RPARAM(3) = 0.0
      RPARAM(4) = 0.0
      IPARAM(6) = 0
      SCALE(1)  = 1.0/ABS(THETA(1))
      SCALE(2)  = 1.0/ABS(THETA(2))
      CALL R2LIN (EXAMPL, NPARM, IDERIV, THETA, R, LDR, IRANK, DFE, &
                  SSE, IPARAM, RPARAM, SCALE, IWK, WK)
      WRITE (NOUT,*) 'THETA = ', THETA
      WRITE (NOUT,*) 'IRANK = ', IRANK, '  DFE = ', DFE, '  SSE = ', &
                      SSE
      WRITE (NOUT,*) 'Number of iterations = ', IPARAM(3)
      WRITE (NOUT,*) 'Number of SSE evaluations = ', IPARAM(4)
      ISETNG=2
      CALL WROPT (-6, ISETNG, 1)
      CALL WRRRN ('Gradient at solution', WK, 1, NPARM, 1)
      CALL WRRRN ('Last step taken', WK((NPARM+1):), 1, NPARM, 1)
      CALL WRRRN ('R', R)
      END
!
      SUBROUTINE EXAMPL (NPARM, THETA, IOPT, IOBS, FRQ, WT, E, DE, &
                         IEND)
      INTEGER    NPARM, IOPT, IOBS, IEND
      REAL       THETA(NPARM), FRQ, WT, E, DE(1)
!
      INTEGER    NOBS
```

```
      PARAMETER  (NOBS=15)
!
      REAL       EXP, XDATA(NOBS), YDATA(NOBS)
      INTRINSIC  EXP
!
      DATA YDATA/54.0E-10, 50.0E-10, 45.0E-10, 37.0E-10, 35.0E-10, &
           25.0E-10, 20.0E-10, 16.0E-10, 18.0E-10, 13.0E-10, 8.0E-10, &
           11.0E-10, 8.0E-10, 4.0E-10, 6.0E-10/
      DATA XDATA/2.0, 5.0, 7.0, 10.0, 14.0, 19.0, 26.0, 31.0, 34.0, &
           38.0, 45.0, 52.0, 53.0, 60.0, 65.0/
!
      IF (IOBS .LE. NOBS) THEN
         WT   = 1.0E0
         FRQ  = 1.0E0
         IEND = 0
         E    = YDATA(IOBS) - THETA(1)*EXP(THETA(2)*XDATA(IOBS))
      ELSE
         IEND = 1
      END IF
      RETURN
      END
```

Output

```
 THETA =     5.86076E-09   -3.95879E-02
 RANK =   2  DFE =     13.0000  SSE =     4.94593E-19
 Number of iterations =   5
 Number of SSE evaluations =   13

     Gradient at solution
           1             2
 6.86656E-14  -1.73762E-20

       Last step taken
           1             2
 -3.24588E-14   3.65805E-07

             R
              1             2
 1   1.87392E+00   1.13981E-07
 2   0.00000E+00   1.13934E-07
```

Example 4

For an extended version of Example 2 that in addition computes the estimated asymptotic variance-covariance matrix of the estimated nonlinear regression parameters, see Example 2 for routine **RCOVB**. The example also computes confidence intervals for the parameters.

Example 5

For an extended version of Example 2 that in addition computes standardized residuals, leverages, and confidence intervals on the mean response, see Example 2 for routine **ROTIN**.

RLAV

Fits a multiple linear regression model using the least absolute values criterion.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)
The absolute value of IIND is the number of independent (explanatory) variables. The sign of IIND specifies the following options:

IIND	Meaning
< 0	The data for the -IIND independent variables are given in the first -IIND columns of X.
> 0	The data for the IIND independent variables are in the columns of X whose column numbers are given by the elements of INDIND.
= 0	There are no independent variables.

The regressors are the constant regressor (if INTCEP = 1) and the independent variables.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables. (Input, if IIND is positive)
If IIND is negative, INDIND is not referenced and can be a vector of length one.

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

B — Vector of length INTCEP + |IIND| containing a LAV solution for the regression coefficients. (Output)
If INTCEP = 1, B(1) contains the intercept estimate. B(INTCEP + I) contains the coefficient estimate for the I-th independent variable.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

INTCEP — Intercept option. (Input)
Default: **INTCEP** = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IRANK — Rank of the matrix of regressors. (Output)
If **IRANK** is less than **INTCEP** + |**IIND**|, linear dependence of the regressors was declared.

SAE — Sum of the absolute values of the errors. (Output)

ITER — Number of iterations performed. (Output)

NRMISS — Number of rows of data containing NaN (not a number) for the dependent or independent variables. (Output)
If a row of data contains NaN for any of these variables, that row is excluded from the computations.

FORTRAN 90 Interface

Generic: **CALL RLAV** (**X**, **IIND**, **INDIND**, **IRSP**, **B** [, ...])
Specific: The specific interface names are **S_RLAV** and **D_RLAV**.

FORTRAN 77 Interface

Single: **CALL RLAV** (**NOBS**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IRSP**, **B**, **IRANK**, **SAE**, **ITER**, **NRMISS**)
Double: The double precision name is **DRLAV**.

Description

Routine **RLAV** computes estimates of the regression coefficients in a multiple linear regression model. The criterion satisfied is the minimization of the sum of the absolute values of the deviations of the observed response y_i from the fitted response

$$\hat{y}_i$$

for a set on *n* observations. Under this criterion, known as the L_1 or LAV (least absolute value) criterion, the regression coefficient estimates minimize

$$\sum_{i=1}^{n} \left| y_i - \hat{y}_i \right|$$

The estimation problem can be posed as a linear programming problem. The special nature of the problem, however, allows for considerable gains in efficiency by the modification of the usual simplex algorithm for linear programming. These modifications are described in detail by Barrodale and Roberts (1973, 1974).

In many cases, the algorithm can be made faster by computing a least-squares solution prior to the invocation of **RLAV**. This is particularly useful when a least-squares solution has already been computed. The procedure is as follows:

1. Fit the model using least squares and compute the residuals from this fit.
2. Fit the residuals from Step 1 on the regressor variables in the model using **RLAV**.
3 Add the two estimated regression coefficient vectors from Steps 1 and 2. The result is an L_1 solution.

When multiple solutions exist for a given problem, routine **RLAV** may yield different estimates of the regression coefficients on different computers, however, the sum of the absolute values of the residuals should be the same (within rounding differences). The informational error indicating nonunique solutions may result from rounding accumulation. Conversely, because of rounding the error may fail to result even when the problem does have multiple solutions.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2AV/DR2AV**. The reference is:

   ```
   CALL R2AV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND,
       IRSP, B, IRANK, SAE, ITER, NRMISS, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length **NOBS**

   WK — Work vector of length **NOBS** * (|**IIND**| + 5) + 2 * |**IIND**| + 4

2. Informational error

Type	Code	Description
3	1	The solution may not be unique.
4	1	Calculations terminated prematurely due to rounding. This occurs only when rounding errors cause a pivot to be encountered whose magnitude is less than AMACH(4) and is indicative of a large ill-conditioned problem.

Example

A straight line fit to a data set is computed under the LAV criterion.

```
      USE RLAV_INT
      USE UMACH_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    LDX, NCOEF, NCOL, NOBS, J
      PARAMETER  (NCOEF=2, NCOL=2, NOBS=8, LDX=NOBS)
!
      INTEGER    IIND, INDIND(1), IRANK, IRSP, ITER, NOUT, &
                 NRMISS
      REAL       B(NCOEF), SAE, X(LDX,NCOL)
      CHARACTER  CLABEL(1)*4, RLABEL(1)*4
!
      DATA (X(1,J),J=1,NCOL) /1.0, 1.0/
      DATA (X(2,J),J=1,NCOL) /4.0, 5.0/
      DATA (X(3,J),J=1,NCOL) /2.0, 0.0/
      DATA (X(4,J),J=1,NCOL) /2.0, 2.0/
      DATA (X(5,J),J=1,NCOL) /3.0, 1.5/
      DATA (X(6,J),J=1,NCOL) /3.0, 2.5/
      DATA (X(7,J),J=1,NCOL) /4.0, 2.0/
      DATA (X(8,J),J=1,NCOL) /5.0, 3.0/
!
      IIND   = -1
      IRSP   = 2
!
      CALL RLAV (X, IIND, INDIND, IRSP, B, irank=irank, sae=sae, &
                 iter=iter, nrmiss=nrmiss)
!
      CALL UMACH (2, NOUT)
      RLABEL(1) = 'B ='
      CLABEL(1) = 'NONE'
      CALL WRRRL (' ', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(F6.2)')
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'SAE = ', SAE
      WRITE (NOUT,*) 'ITER = ', ITER
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
 B =    0.50    0.50
IRANK =   2
SAE =     6.00000
```

```
    ITER =   2
    NRMISS =   0
```

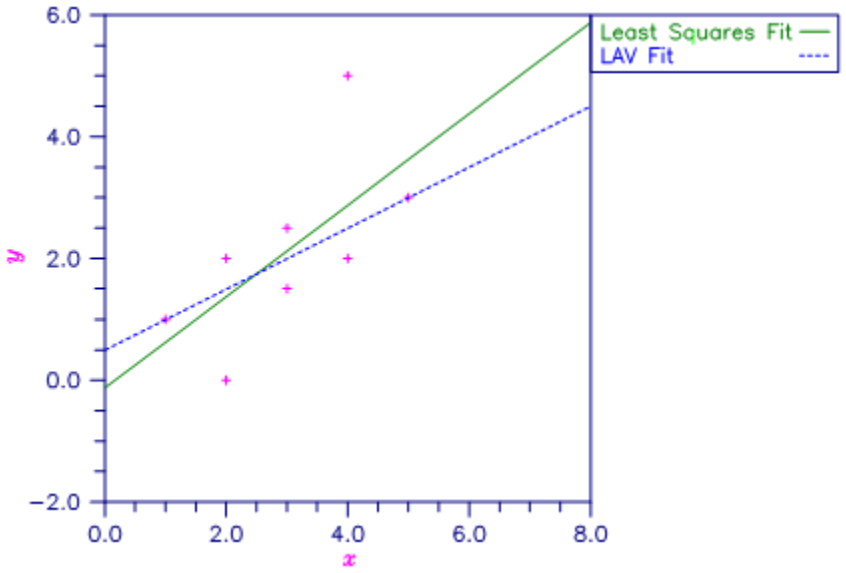


Figure 11, Least Squares and Least Absolute Value Fitted Lines

RLLP

Fits a multiple linear regression model using the L_p norm criterion.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)

IIND	Meaning
< 0	The first -IIND columns of X contain the independent (explanatory) variables.
> 0	The IIND independent variables are specified by the column numbers in INDIND.
= 0	There are no independent variables.

There are NCOEF = INTCEP + |IIND| regressors—the constant regressor (if INTCEP = 1) and the independent variables.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables. (Input, if IIND is positive)
If IIND is negative, INDIND is not referenced and can be a vector of length one.

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

P — The p in the L_p norm. (Input)
p must be greater than or equal to 1.0. A common choice for p is between 1.0 and 2.0, inclusively.

B — Vector of length NCOEF containing an L_p solution for the regression coefficients. (Output)
If INTCEP = 1, B(1) contains the intercept estimate. B(INTCEP + I) contains the coefficient estimate for the I-th independent variable.

Optional Arguments

NOBS — Number of rows in X. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

INTCEP — Intercept option. (Input)
Default: **INTCEP** = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive **IFRQ**, column number **IFRQ** of **X** contains the frequencies.
Default: **IFRQ** = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive **IWT**, column number **IWT** of **X** contains the weights.
Default: **IWT** = 0.

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100 * **AMACH**(4) is a common choice. See documentation for IMSL routine AMACH in the *Reference Material*.
Default: **TOL** = 1.e-5 for single precision and 2.d –14 for double precision.

MAXIT — Maximum number of iterations permitted. (Input)
A common choice is **MAXIT** = 100.
Default: **MAXIT** = 100.

EPS — Convergence criterion. (Input)
If the maximum relative difference in residuals from the *k*-th to (*k* + 1)-st iterations is less than **EPS**, convergence is declared. **EPS** = 100 * **AMACH**(4) is a common choice.
Default: **EPS** = 1.e-5 for single precision and 2.d –14 for double precision.

R — **NCOEF** by **NCOEF** upper triangular matrix containing the *R* matrix from a *QR* decomposition of the matrix of regressors. (Output)

LDR — Leading dimension of **R** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDR** = size (**R**,1).

IRANK — Rank of the matrix of regressors. (Output)
If **IRANK** is less than **NCOEF**, linear dependence of the regressors is declared.

DFE — Sum of the frequencies minus **IRANK**. (Output) In a least squares fit (p = 2), **DFE** is called the degrees of freedom for error.

E — Vector of length **NOBS** containing the residuals. (Output)

SCALE2 — Square of the scale constant used in an L_p analysis. (Output)
An estimated asymptotic variance-covariance matrix of the regression coefficients is SCALE2 * $(R^TR)^{-1}$.

ELP — L_p norm of the residuals. (Output)

ITER — Number of iterations performed. (Output)

NRMISS — Number of rows of data that contain any missing values for the independent, dependent, weight, or frequency variables. (Output)
NaN (not a number) is used as the missing value code. Any row of **X** containing NaN as a value of the independent, dependent, weight, or frequency variables is omitted from the analysis.

FORTRAN 90 Interface

Generic: CALL RLLP (X, IIND, INDIND, IRSP, P, B [, ...])
Specific: The specific interface names are S_RLLP and D_RLLP.

FORTRAN 77 Interface

Single: CALL RLLP (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND, IRSP, IFRQ, IWT, P, TOL, MAXIT, EPS, B, R, LDR, IRANK, DFE, E, SCALE2, ELP, ITER, NRMISS)
Double: The double precision name is DRLLP.

Description

Routine **RLLP** computes estimates of the regression coefficients in a multiple linear regression model $y = X\beta + \varepsilon$ under the criterion of minimizing the L_p norm of the deviations for $i = 1, \ldots, n$ of the observed response y_i from the fitted response

$$\hat{y}_i$$

for a set on n observations and for $p \geq 1$. For the case **IWT** = 0 and **IFRQ** = 0 the estimated regression coefficient vector,

$$\hat{\beta}$$

(output in **B**) minimizes the L_p norm

$$\left(\sum_{i=1}^{n} |y_i - \hat{y}_i|^p \right)^{1/p}$$

The choice $p = 1$ yields the maximum likelihood estimate for **β** when the errors have a Laplace distribution. The choice $p = 2$ is best for errors that are normally distributed. Sposito (1989, pages 36-40) discusses other reasonable alternatives for p based on the sample kurtosis of the errors.

Weights are useful if the errors in the model have known unequal variances

$$\sigma_i^2$$

In this case, the weights should be taken as

$$w_i = 1 / \sigma_i^2$$

Frequencies are useful if there are repetitions of some observations in the data set. If a single row of data corresponds to n_i observations, set the frequency $f_i = n_i$. In general, **RLLP** minimizes the L_p norm

$$\left(\sum_{i=1}^{n} f_i |\sqrt{w_i} \left(y_i - \hat{y}_i \right)|^p \right)^{1/p}$$

The asymptotic variance-covariance matrix of the estimated regression coefficients is given by

$$\text{asy.var}\left(\hat{\beta} \right) = \lambda^2 (R^T R)^{-1}$$

where R is from the QR decomposition of the matrix of regressors (output in **R**) and where an estimate of λ^2 is output in **SCALE2**. An estimated asymptotic variance-covariance matrix of the estimated regression coefficients can be computed following the call to **RLLP** by invoking routine **RCOVB** with **R** and **SCALE2**.

In the discussion that follows, we will first present the algorithm with frequencies and weights all taken to be one. Later, we will present the modifications to handle frequencies and weights different from one.

Routine **RLLP** uses Newton's method with a line search for $p > 1.25$ and, for $p \leq 1.25$, uses a modification due to Ekblom (1973, 1987) in which a series of perturbed problems are solved in order to guarantee convergence and increase the convergence rate. The cutoff value of 1.25 as well as some of the other implementation details given in the remaining discussion were investigated by Sallas (1990) for their effect on **CPU** times.

In each case, for the first iteration a least-squares solution for the regression coefficients is computed using routine **RGIVN**. If $p = 2$, the computations are finished. Otherwise, the residuals from the k-th iteration,

$$e_i^{(k)} = y_i - \hat{y}_i^{(k)}$$

are used to compute the gradient and Hessian for the Newton step for the (k + 1)-st iteration for minimizing the p-th power of the L_p norm. (The exponent 1/p in the L_p norm can be omitted during the iterations.)

For subsequent iterations, we first discuss the p > 1.25 case. For p > 1.25, the gradient and Hessian at the (k + 1)-st iteration depend upon

$$z_i^{(k+1)} = |e_i^{(k)}|^{p-1} \text{sign}\left(e_i^{(k)}\right)$$

and

$$v_i^{(k+1)} = |e_i^{(k)}|^{p-2}$$

In the case 1.25 < p < 2 and

$$e_i^{(k)} = 0, v_i^{(k+1)}$$

and the Hessian are undefined; and we follow the recommendation of Merle and Spath (1974). Specifically, we modify the definition of

$$v_i^{(k+1)}$$

to the following:

$$v_i^{(k+1)} = \begin{cases} \tau^{p-2} & \text{if } p < 2 \text{ and } |e_i^{(k)}| < \tau \\ |e_i^{(k)}|^{p-2} & \text{otherwise} \end{cases}$$

where τ equals 100 * **AMACH**(4) times the square root of the residual mean square from the least-squares fit. (See routine **AMACH** , which is documented in the section "Machine-Dependent Constants" in the *Reference Material*.)

Let $V^{(k+1)}$ be a diagonal matrix with diagonal entries

$$v_i^{(k+1)}$$

and let $z^{(k+1)}$ be a vector with elements

$$z_i^{(k+1)}$$

In order to compute the step on the (k + 1)-st iteration, the R from the QR decomposition of $[V^{(k+1)}]^{1/2}X$ is computed using fast Givens transformations. Let $R^{(k+1)}$ denote the upper triangular matrix from the QR decomposition. Routine **GIRTS** (see Chapter 20, Mathematical Support) is used to solve the linear system $[R^{(k+1)}]^T R^{(k+1)} d^{(k+1)} = X^T z^{(k+1)}$ for $d^{(k+1)}$ where $R^{(k+1)}$ is from the QR decomposition of $[V^{(k+1)}]^{1/2}X$. The step taken on the (k + 1)-st iteration is

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + \alpha^{(k+1)} \frac{1}{p^{-1}} d^{(k+1)}$$

The first attempted step on the (k + 1)-st iteration is with $\alpha^{(k+1)} = 1$. If all of the

$$e_i^{(k)}$$

are nonzero, this is exactly the Newton step. See Kennedy and Gentle (1980, pages 528-529) for further discussion.

If the first attempted step does not lead to a decrease of at least one-tenth of the predicted decrease in the p-th power of the L_p norm of the residuals, a backtracking linesearch procedure is used. The backtracking procedure uses a one-dimensional quadratic model to estimate the backtrack constant ρ. The value of ρ is constrained to be no less that 0.1. An approximate upper bound for ρ is 0.5. If after 10 successive backtrack attempts, $\alpha^{(k)} = \rho_1\rho_2\ldots\rho_{10}$ does not produce a step with a sufficient decrease, then **RLLP** issues a message with error code 5. For further details on the backtrack line-search procedure, see Dennis and Schnabel (1983, pages 126-127).

Convergence is declared when the maximum relative change in the residuals from one iteration to the next is less than or equal to **EPS**. The relative change

$$\delta_i^{(k+1)}$$

in the i-th residual from iteration k to iteration k + 1 is computed as follows:

$$\delta_i^{(k+1)} = \begin{cases} 0 & \text{if } e_i^{(k+1)} = e_i^{(k)} = 0 \\ \left| e_i^{(k+1)} - e_i^{(k)} \right| / \max\left(\left| e_i^{(k)} \right|, \left| e_i^{(k+1)} \right|, s \right) & \text{otherwise} \end{cases}$$

where s is the square root of the residual mean square from the least-squares fit on the first iteration.

For the case $1 \le p \le 1.25$, we describe the modifications to the previous procedure that incorporate Ekblom's (1973) results. A sequence of perturbed problems are solved with a successively smaller perturbation constant c. On the first iteration, the least-squares problem is solved. This corresponds to an infinite c. For the second problem, c is taken equal to s, the square root of the residual mean square from the least-squares fit. Then, for the $(j + 1)$-st problem, the value of c is computed from the previous value of c according to

$$c_{j+1} = c_j / 10^{5p-4}$$

Each problem is stated as

$$\text{Minimize} \sum_{i=1}^{n} \left(e_i^2 + c^2 \right)^{p/2}$$

For each problem, the gradient and Hessian on the $(k + 1)$-st iteration depend upon

$$z_i^{(k+1)} = e_i^{(k)} r_i^{(k)}$$

and

$$v_i^{(k+1)} = \left[1 + \frac{(p-2)(e_i^{(k)})^2}{(e_i^{(k)})^2 + c^2} \right] r_i^{(k)}$$

where

$$r_i^{(k)} = \left[(e_i^{(k)})^2 + c^2 \right]^{(p-2)/2}$$

The linear system $[R^{(k+1)}]^T R^{(k+1)} d^{(k+1)} = X^T Z^{(k+1)}$ is solved for $d^{(k+1)}$ where $R^{(k+1)}$ is from the QR decomposition of $[V^{(k+1)}]^{1/2} X$. The step taken on the $(k + 1)$-st iteration is

$$\hat{\beta}^{(k+1)} = \hat{\beta}^{(k)} + \alpha^{(k+1)} d^{(k+1)}$$

where the first attempted step is with $\alpha^{(k+1)} = 1$. If necessary, the backtracking line-search procedure discussed earlier is used.

Convergence for each problem is relaxed somewhat by using a convergence epsilon equal to $\max(\text{EPS}, 10^{-j})$ where $j = 1, 2, 3, \ldots$ indexes the problems ($j = 0$ corresponds to the least-squares problem).

After the convergence of a problem for a particular c, Ekblom's (1987) extrapolation technique is used to compute the initial estimate of β for the new problem. Let $R^{(k)}$, $v_i^{(k)}$, $e_i^{(k)}$ and c be from the last iteration of the last problem. Let

$$t_i = \frac{(p-2)v_i^{(k)}}{(e_i^{(k)})^2 + c^2}$$

and let t be the vector with elements t_i. The initial estimate of β for the new problem with perturbation constant $0.01c$ is

$$\hat{\beta}^{(0)} = \hat{\beta}^{(k)} + \Delta c d$$

where $\Delta c = (0.01c - c) = -0.99c$, and where d is the solution of the linear system $[R^{(k)}]^T R^{(k)} d = X^T t$.

Convergence of the sequence of problems is declared when the maximum relative difference in residuals from the solution of successive problems is less than **EPS**.

The preceding discussion was limited to the case for which **IWT** = 0 and **IFRQ** = 0, i.e., the weights and frequencies are all taken equal to one. The necessary modifications to the preceding algorithm to handle weights and frequencies not all equal to one are as follows:

1. Replace

   $$e_i^{(k)} \text{ by } \sqrt{w_i}\, e_i^{(k)}$$

   in the definitions of

   $$z_i^{(k+1)}, v_i^{(k+1)}, \delta_i^{(k+1)}$$

   and t_i.

2. Replace

   $$z_i^{(k+1)} \text{ by } f_i \sqrt{w_i}\, z_i^{(k+1)}, \quad v_i^{(k+1)} \text{ by } f_i w_i v_i^{(k+1)}, \quad \text{and } t_i^{(k+1)} \text{ by } f_i \sqrt{w_i}\, t_i^{(k+1)}$$

These replacements have the same effect as multiplying the i-th row of X and y by

$$\sqrt{w_i}$$

and repeating the row f_i times except for the fact that the residuals returned by **RLLP** are in terms of the original y and X.

Finally, R and an estimate of λ^2 are computed. Actually, R is recomputed because on output it corresponds to the R from the initial QR decomposition for least squares. The formula for the estimate of λ^- depends on p.

For $p = 1$, the estimator for λ^2 is given by (McKean and Schrader 1987)

$$\hat{\lambda}^2 = \left[\frac{\sqrt{\text{DFE}} \left(\tilde{e}_{(\text{DFE}-k+1)} - \tilde{e}_{(k)} \right)}{2z_{0.975}} \right]^2$$

with

$$k = \frac{\text{DFE} + k}{2} - z_{0.975}\sqrt{\frac{\text{DFE}}{4}}$$

where $z_{0.975}$ is the 97.5 percentile of the standard normal distribution, and where

$$\tilde{\varepsilon}_{(m)} \left(m = 1, 2, \ \ldots, \text{DFE} \right)$$

are the ordered residuals where **IRANK** zero residuals are excluded. (Note that

$$\text{DFE} = \sum_{i=1}^{n} f_i - \text{IRANK}$$

For $p = 2$, the estimator of λ^2 is the customary least-squares estimator given by

$$s^2 = \frac{\text{SSE}}{\text{DFE}} = \frac{\sum_{i=1}^{n} f_i w_i (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} f_i - \text{IRANK}}$$

For $1 < p < 2$ and for $p > 2$, the estimator for λ^2 is given by (Gonin and Money 1989)

$$\hat{\omega}_p^2 = \frac{m_{2p-2}}{\left[(p-1) m_{p-2} \right]^2}$$

with

$$m_r = \frac{\sum_{i=1}^{n} f_i \left| \sqrt{w_i} (y_i - \hat{y}_i) \right|^r}{\sum_{i=1}^{n} f_i}$$

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2LP/DR2LP**. The reference is:

   CALL R2LP (**NROW**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IRSP**, **IFRQ**, **IWT**, **P**, **TOL**, **MAXIT**, **EPS**, **B**, **R**, **LDR**, **IRANK**, **DFE**, **E**, **SCALE2**, **ELP**, **ITER**, **NRMISS**, **IWK**, **WK**)

   The additional arguments are as follows:

IWK — Work array of length NOBS + |IIND| + 3.

WK — Work array of length 2 * NOBS + 8 * NCOEF + 4.

2. Informational errors

Type	Code	Description
4	1	A negative weight was encountered.
4	2	A negative frequency was encountered.
4	3	The *p*-th power of the absolute value of one or more of the current residuals will result in overflow or underflow in subsequent computations. A solution cannot be computed because of a serious loss of accuracy. For large *p*, consider the use of IMSL routine RLMV, which uses the L_∞ (minimax) criterion.
3	4	Convergence has not been achieved after MAXIT iterations. MAXIT or EPS may be too small. Try increasing MAXIT or EPS.
3	5	Convergence is not declared. The line-search procedure failed to find an acceptable solution after 10 successive attempts. EPS may be too small. Try increasing its value.

Example

Different straight line fits to a data set are computed under the criterion of minimizing the L_p norm by using p equal to 1, 1.5, 2.0 and 2.5.

```
      USE RLLP_INT
      USE UMACH_INT
      USE WRRRL_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    INTCEP, LDR, LDX, NCOEF, NCOL, NIND, NOBS, J
      PARAMETER  (INTCEP=1, NCOL=2, NIND=1, NOBS=8, LDX=NOBS, &
                 NCOEF=INTCEP+NIND, LDR=NCOEF)
!
      INTEGER    IIND, INDIND(NIND), IRANK, IRSP, ITER, NOUT, NRMISS
      REAL       B(NCOEF), DFE, E(NOBS), ELP, EPS, P, &
                 R(LDR,NCOEF), SCALE2, X(LDX,NCOL)
      CHARACTER  CLABEL(1)*4, RLABEL(1)*12
!
      DATA (X(1,J),J=1,NCOL)/1.0, 1.0/
      DATA (X(2,J),J=1,NCOL)/4.0, 5.0/
      DATA (X(3,J),J=1,NCOL)/2.0, 0.0/
      DATA (X(4,J),J=1,NCOL)/2.0, 2.0/
      DATA (X(5,J),J=1,NCOL)/3.0, 1.5/
      DATA (X(6,J),J=1,NCOL)/3.0, 2.5/
      DATA (X(7,J),J=1,NCOL)/4.0, 2.0/
      DATA (X(8,J),J=1,NCOL)/5.0, 3.0/
!
      CALL UMACH (2, NOUT)
      IIND      = NIND
```

```
      INDIND(1) = 1
      IRSP      = 2
      EPS       = 0.001
!
      DO 10  P=1.0, 2.5, 0.5
         CALL RLLP (X, IIND, INDIND, IRSP, P, B, eps=eps, r=r, &
                    irank=irank, DFE=DFE, e=e, scale2=scale2, elp=elp, &
                    ITER=ITER, nrmiss=nrmiss)
!
         WRITE (NOUT,99997)
         RLABEL(1) = 'Coefficients'
         CLABEL(1) = 'NONE'
         CALL WRRRL ('%/', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(F6.2)')
         RLABEL(1) = 'Residuals'
         CLABEL(1) = 'NONE'
         CALL WRRRL ('%/', E, RLABEL, CLABEL, 1, NOBS, 1, FMT='(F6.2)')
         WRITE (NOUT,*)
         WRITE (NOUT,99998) 'p', P
         WRITE (NOUT,99998) 'Lp norm of the residuals', ELP
         WRITE (NOUT,99999) 'Rank of the matrix of regressors', IRANK
         WRITE (NOUT,99998) 'Degrees of freedom error', DFE
         WRITE (NOUT,99999) 'Number of iterations', ITER
         WRITE (NOUT,99999) 'Number of missing values', NRMISS
         WRITE (NOUT,99998) 'Square of the scale constant', SCALE2
         CALL WRRRN ('R matrix', R)
   10 CONTINUE
99997 FORMAT (/1X, 72('-'))
99998 FORMAT (1X, A, T34, F5.2)
99999 FORMAT (1X, A, T34, I5)
      END
```

Output

```
------------------------------------------------------------------------
Coefficients    0.50    0.50
Residuals    0.00    2.50   -1.50    0.50   -0.50    0.50   -0.50    0.00

p                               1.00
Lp norm of the residuals        6.00
Rank of the matrix of regressors    2
Degrees of freedom error        6.00
Number of iterations               8
Number of missing values           0
Square of the scale constant    6.25

   R matrix
        1       2
1   2.828   8.485
2   0.000   3.464

------------------------------------------------------------------------

Coefficients    0.39    0.55

Residuals    0.06    2.39   -1.50    0.50   -0.55    0.45   -0.61   -0.16

p                               1.50
Lp norm of the residuals        3.71
```

```
Rank of the matrix of regressors    2
Degrees of freedom error         6.00
Number of iterations                6
Number of missing values            0
Square of the scale constant     1.06

   R matrix
        1       2
1   2.828   8.485
2   0.000   3.464

------------------------------------------------------------------------

Coefficients   -0.12    0.75
Residuals    0.38    2.12   -1.38    0.62   -0.62    0.38   -0.88   -0.62

p                               2.00
Lp norm of the residuals        2.94
Rank of the matrix of regressors    2
Degrees of freedom error         6.00
Number of iterations                1
Number of missing values            0
Square of the scale constant     1.44

   R matrix
        1       2
1   2.828   8.485
2   0.000   3.464

------------------------------------------------------------------------

Coefficients   -0.44    0.87
Residuals    0.57    1.96   -1.30    0.70   -0.67    0.33   -1.04   -0.91
p                               2.50
Lp norm of the residuals        2.54
Rank of the matrix of regressors    2
Degrees of freedom error         6.00
Number of iterations                4
Number of missing values            0
Square of the scale constant     0.79

   R matrix
        1       2
1   2.828   8.485
2   0.000   3.464
```

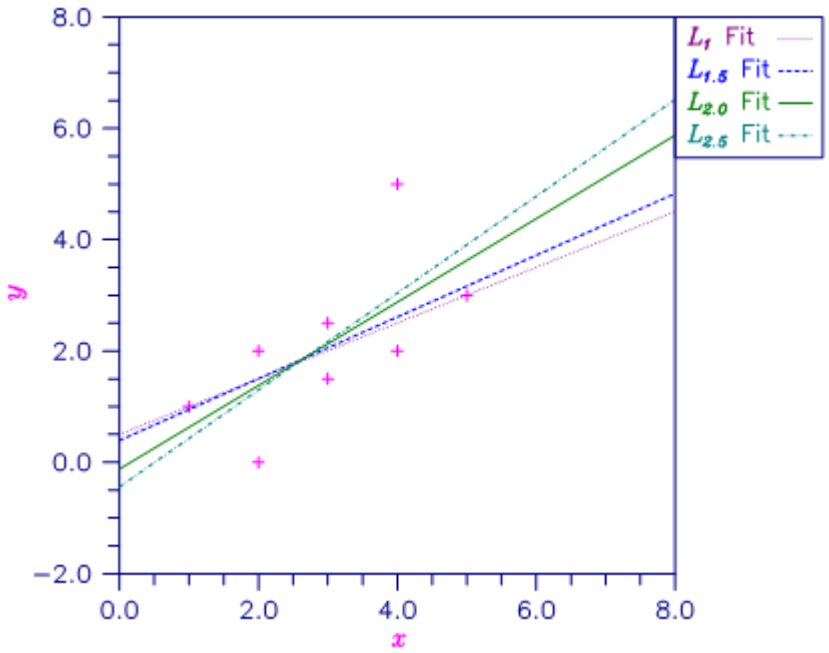


Figure 12, Various Lp Fitted Lines

RLMV

Fits a multiple linear regression model using the minimax criterion.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IIND — Independent variable option. (Input)
The absolute value of IIND is the number of independent (explanatory) variables. The sign of IIND specifies the following options :

IIND	Meaning
< 0	The data for the -IIND independent variables are given in the first -IIND columns of X.
> 0	The data for the IIND independent variables are in the columns of X whose column numbers are given by the elements of INDIND.
= 0	There are no independent variables.

The regressors are the constant regressor (if INTCEP = 1) and the independent variables.

INDIND — Index vector of length IIND containing the column numbers of X that are the independent (explanatory) variables. (Input, if IIND is positive)
If IIND is negative, INDIND is not referenced and can be a vector of length one.

IRSP — Column number IRSP of X contains the data for the response (dependent) variable. (Input)

B — Vector of length INTCEP + |IIND| containing a minimax solution for the regression coefficients. (Output)
If INTCEP = 1, B(1) contains the intercept estimate. B(INTCEP + I) contains the coefficient estimate for the I-th independent variable.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

INTCEP — Intercept option. (Input)
Default: **INTCEP** = 1.

INTCEP	Action
0	An intercept is not in the model.
1	An intercept is in the model.

IRANK — Rank of the matrix of regressors. (Output)
If **IRANK** is less than **INTCEP** + |**IIND**|, linear dependence of the regressors was declared.

AEMAX — Magnitude of the largest residual. (Output)

ITER — Number of iterations performed. (Output)

NRMISS — Number of rows of data containing NaN (not a number) for the dependent or independent variables. (Output)
If a row of data contains NaN for any of these variables, that row is excluded from the computations.

FORTRAN 90 Interface

Generic: **CALL RLMV** (**X**, **IIND**, **INDIND**, **IRSP**, **B** [, ...])
Specific: The specific interface names are **S_RLMV** and **D_RLMV**.

FORTRAN 77 Interface

Single: **CALL RLMV** (**NOBS**, **NCOL**, **X**, **LDX**, **INTCEP**, **IIND**, **INDIND**, **IRSP**, **B**, **IRANK**, **AEMAX**, **ITER**, **NRMISS**)
Double: The double precision name is **DRLMV**.

Description

Routine **RLMV** computes estimates of the regression coefficients in a multiple linear regression model. The criterion satisfied is the minimization of the maximum deviation of the observed response y_i from the fitted response $\hat{y}_i$ for a set on n observations. Under this criterion, known as the minimax or LMV (least maximum value) criterion, the regression coefficient estimates minimize

$$\max_{1 \le i \le n} |y_i - \hat{y}_i|$$

The estimation problem can be posed as a linear programming problem. A dual simplex algorithm is appropriate, however, the special nature of the problem allows for considerable gains in efficiency by modification of the dual simplex iterations so as to move more rapidly toward the optimal solution. The modifications are described in detail by Barrodale and Phillips (1975).

When multiple solutions exist for a given problem, **RLMV** may yield different estimates of the regression coefficients on different computers, however, the largest residual in absolute value should have the same absolute value (within rounding differences). The informational error indicating nonunique solutions may result from rounding accumulation. Conversely, because of rounding, the error may fail to result even when the problem does have multiple solutions.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2MV/DR2MV**. The reference is:

   ```
   CALL R2MV (NOBS, NCOL, X, LDX, INTCEP, IIND, INDIND, IRSP, B, IRANK, AEMAX, ITER,
       NRMISS, WK)
   ```

   The additional argument is:

   WK — Workspace of length **NOBS** * (|**IIND**| + 5) + 2 * |**IIND**| + 3.

2. Informational errors

Type	Code	Description
3	1	The solution may not be unique.
4	1	Calculations terminated prematurely due to rounding.

3. If **X** is not needed, **LDX** = **NOBS**, and **IIND** < 0, then **X** and the first **NOBS** * (**-IIND** + 1) elements of **WK** may occupy the same storage locations. The reference would be:

   ```
   CALL R2MV (NOBS, NCOL, WK, NOBS, INTCEP, IIND, INDIND, IRSP, B, IRANK, AEMAX,
       ITER, NRMISS, WK)
   ```

Example

A straight line fit to a data set is computed under the LMV criterion.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE RLMV_INT
      USE UMACH_INT
      USE WRRRL_INT

      IMPLICIT   NONE
```

```
      INTEGER    LDX, NCOEF, NCOL, NOBS, J
      PARAMETER  (NCOEF=2, NCOL=2, NOBS=7, LDX=NOBS)
!
      INTEGER    IIND, INDIND(1), IRANK, IRSP, ITER, NOUT, &
                 NRMISS
      REAL       B(NCOEF), AEMAX, X(LDX,NCOL)
      CHARACTER  CLABEL(1)*4, RLABEL(1)*4
!
      DATA (X(1,J),J=1,NCOL)/0.0, 0.0/
      DATA (X(2,J),J=1,NCOL)/1.0, 2.5/
      DATA (X(3,J),J=1,NCOL)/2.0, 2.5/
      DATA (X(4,J),J=1,NCOL)/3.0, 4.5/
      DATA (X(5,J),J=1,NCOL)/4.0, 4.5/
      DATA (X(6,J),J=1,NCOL)/4.0, 6.0/
      DATA (X(7,J),J=1,NCOL)/5.0, 5.0/
!
      IIND   = -1
      IRSP   = 2
!
      CALL RLMV (X, IIND, INDIND, IRSP, B, IRANK=IRANK, AEMAX=AEMAX, &
                 iter=iter, nrmiss=nrmiss)
!
      CALL UMACH (2, NOUT)
      RLABEL(1) = 'B ='
      CLABEL(1) = 'NONE'
      CALL WRRRL (' ', B, RLABEL, CLABEL, 1, NCOEF, 1, FMT='(F6.2)')
      WRITE (NOUT,*) 'IRANK = ', IRANK
      WRITE (NOUT,*) 'AEMAX = ', AEMAX
      WRITE (NOUT,*) 'ITER = ', ITER
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
 B =    1.00    1.00
 IRANK =   2
 AEMAX =     1.00000
 ITER =   3
 NRMISS =   0
```

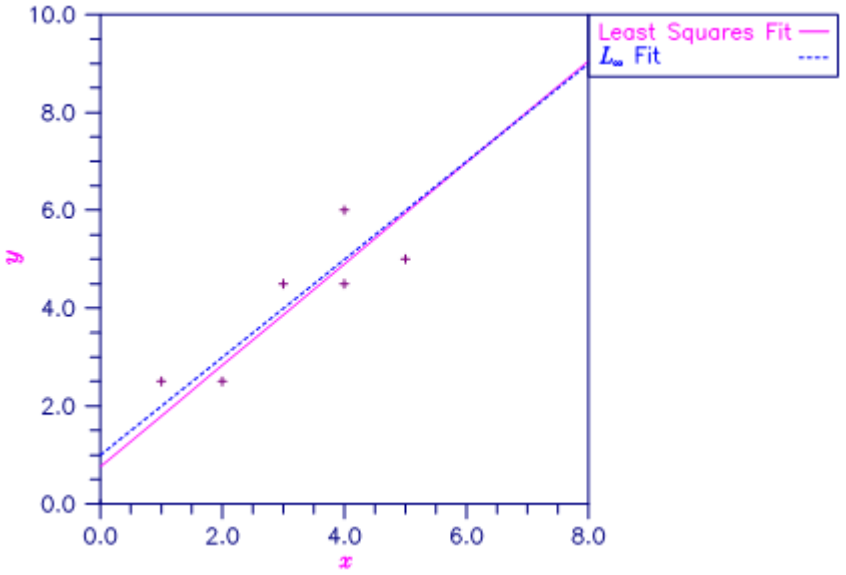


Figure 13, Least Squares and Least Maximum Value Fitted Lines

PLSR

more...

Performs partial least squares regression for one or more response variables and one or more predictor variables.

Required Arguments

Y — Array of size n_y by h containing the values of the responses, where $n_y \geq$ **NOBS** is the number of rows of Y and h is the number of response variables. (Input)

X — Array of size n_x by p containing the values of the predictor variables, where $n_x \geq$ **NOBS** is the number of rows of X and p is the number of predictor variables. (Input)

COEF — Array of size `SIZE(IXIND)` by `SIZE(IYIND)` containing the final PLS regression coefficient estimates. (Output)

Optional Arguments

NOBS — Positive integer specifying the number of observations to be used in the analysis. (Input)
Default : **NOBS** = *min*(size (**Y**,1), size (**X**,1)).

IYIND — Array containing column indices of Y specifying which response variables to use in the analysis.
`MAXVAL(IYIND)` $\leq h$. (Input)
Default: `IYIND` = 1, 2, ..., h.

IXIND — Array containing column indices of X specifying which predictor variables to use in the analysis.
`MAXVAL(IXIND)` $\leq p$. (Input)
Default: `IXIND` = 1, 2,..., p.

NCOMPS — The number of PLS components to fit. **NCOMPS** $\leq$ `SIZE`(`IXIND`). (Input)
Default: **NCOMPS** = size (`IXIND`).

Note: If **CV** = `.TRUE.`, models with 1 up to **NCOMPS** components are tested using cross-validation. The model with the lowest predicted residual sum of squares is reported.

CV — Logical. If **.TRUE.**, the routine performs *K*-fold cross validation to select the number of components. If .FALSE., the routine fits only the model specified by **NCOMPS**. (Input)
Default: **CV** = .TRUE.

K — Integer specifying the number of folds to use in *K*-fold cross validation. **K** must be between 2 and **NOBS**, inclusive. **K** is ignored if **CV** = **.FALSE.** (Input)
Default**:** **K** = 5.

Note: If **NOBS**/**K** ≤ 3, the routine performs leave-one-out cross validation as opposed to *K*-fold cross validation.

SCALE — Logical. If **.TRUE.**, *Y* and *X* are centered and scaled to have mean 0 and standard deviation of 1. If **.FALSE.**, *Y* and *X* are centered only. (Input)
Default: **SCALE** = .FALSE.

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	Prints final results only.
2	Prints final results and intermediate results.

YHAT — Array of size **NOBS** by *h* containing the predicted values for the response variables using the final values of the coefficients. (Output)

RESIDS — Array of size **NOBS** by *h* containing residuals of the final fit for each response variable. (Output)

SE — Array of size *p* by *h* containing the standard errors of the PLS coefficients. (Output)

PRESS — Array of size **NCOMPS** by *h* providing the predicted residual error sum of squares obtained by cross-validation for each model of size *j* = 1, …, **NCOMPS** components. The argument **PRESS** is ignored if **CV** = **.FALSE.**. (Output)

XSCRS — Array of size **NOBS** by **NCOMPS** containing *X*-scores. (Output)

YSCRS — Array of size **NOBS** by **NCOMPS** containing *Y*-scores. (Output)

XLDGS — Array of size *p* by **NCOMPS** containing *X*-loadings. (Output)

YLDGS — Array of size *h* by **NCOMPS** containing *Y*-loadings. (Output)

WTS — Array of size *p* by **NCOMPS** containing the weight vectors. (Output)

FORTRAN

Generic: **CALL PLSR** (**X**, **Y**, **COEF** [, ...])
Specific: The specific interface names are **S_PLSR** and **D_PLSR**.

Description

Routine **PLSR** performs partial least squares regression for a response matrix $Y\left(n_y \times h\right)$, and a set of *p* explanatory variables, $X\left(n_x \times p\right)$. **PLSR** finds linear combinations of the predictor variables that have highest covariance with *Y*. In so doing, **PLSR** produces a predictive model for *Y* using components (linear combinations) of the individual predictors. Other names for these linear combinations are scores, factors, or latent variables. Partial least squares regression is an alternative method to ordinary least squares for problems with many, highly collinear predictor variables. For further discussion see, for example, Abdi (2010), and Frank and Friedman (1993).

In Partial Least Squares (PLS), a score, or component matrix, *T*, is selected to represent both *X* and *Y* as in,

$$X = TP^{\mathrm{T}} + E_{\mathrm{x}}$$

and

$$Y = TQ^{\mathrm{T}} + E_{\mathrm{y}}$$

The matrices *P* and *Q* are the least squares solutions of *X* and *Y* regressed on *T*.

That is,

$$Q^{\mathrm{T}} = (T^{\mathrm{T}}T)^{-1}T^{\mathrm{T}}Y$$

and

$$P^{\mathrm{T}} = (T^{\mathrm{T}}T)^{-1}T^{\mathrm{T}}X$$

The columns of *T* in the above relations are often called *X*-scores, while the columns of *P* are the *X*-loadings. The columns of the matrix *U* in $Y = UQ^{\mathrm{T}} + G$ are the corresponding *Y* scores, where *G* is a residual matrix and *Q* as defined above contains the *Y*-loadings.

Restricting *T* to be linear in *X* , the problem is to find a set of weight vectors (columns of *W*) such that *T* = *XW* predicts both *X* and *Y* reasonably well.

Formally, $W = \left[w_1,...w_{m-1},w_m,..w_M \right]$ where each w_{j} is a column vector of length *p*, $M \le p$ is the number of components, and where the *m*-th partial least squares (PLS) component w_{m} solves:

$$\begin{cases} \max_{\alpha} \mathrm{Corr}^2(\mathrm{Y},\mathrm{X}\alpha)\mathrm{Var}(X\alpha) \\ s.t. \\ ||\alpha|| = 1 \\ \alpha^{\mathrm{T}} S w_l = 0, \quad l = 1,...m-1 \end{cases}$$

where $S = X^T X$ and $||\alpha|| = \sqrt{(\alpha,\alpha)} = \sqrt{\alpha^{\mathrm{T}}\alpha}$ is the Euclidean norm. For further details see Hastie, et. al., pages 80-82 (2001).

That is, w_m is the vector which maximizes the product of the squared correlation between *Y* and $X\boldsymbol{\alpha}$ and the variance of $X\boldsymbol{\alpha}$, subject to being orthogonal to each previous weight vector left multiplied by *S*. The PLS regression coefficients $\hat{\beta}_{PLS}$ arise from

$$Y = X\hat{\beta}_{PLS} + E_y = TQ^T + E_y = XWQ^T + E_y, \text{ or } \hat{\beta}_{PLS} = WQ^T$$

Algorithms to solve the above optimization problem include NIPALS (nonlinear iterative partial least squares) developed by Herman Wold (1966, 1985) and numerous variations, including the SIMPLS algorithm of de Jong (1993). Subroutine **PLSR** implements the SIMPLS method. SIMPLS is appealing because it finds a solution in terms of the original predictor variables, whereas NIPALS reduces the matrices at each step. For univariate *Y* it has been shown that SIMPLS and NIPALS are equivalent (the score, loading, and weights matrices will be proportional between the two methods).

If **CV=.TRUE.**, **PLSR** searches for the best number of PLS components using *K*-fold cross-validation. That is, for each *M* = 1, 2, …, *p*, **PLSR** estimates a PLS model with *M* components using all of the data except a hold-out set of size roughly equal to **NOBS/K**. Using the resulting model estimates, **PLSR** predicts the outcomes in the hold-out set and calculates the predicted residual sum of squares (PRESS). The procedure then selects the next hold-out sample and repeats for a total of *K* times (i.e., *folds*). For further details see Hastie, et. al., pages 241-245 (2001).

For each response variable, **PLSR** returns results for the model with lowest PRESS. The best model (the number of components giving lowest PRESS), generally will be different for different response variables.

When requested via the optional argument **SE**, **PLSR** calculates modifed jackknife estimates of the standard errors as described in Martens and Martens (2000).

Comments

1. **PLSR** defaults to leave-one-out cross-validation when there are too few observations to form *K* folds in the data. The user is cautioned that there may be too few observations to make strong inferences from the results:

2. Informational errors

Type	Code	Description
2	1	For response #, residuals converged in # components, while # is the requested number of components.

3. This implementation of PLSR does not handle missing values. The user should remove missing values in the data. The user should removes missing data or NaN's from the data input.

Examples

Example 1

The following artificial data set is provided in de Jong (1993).

$$X = \begin{bmatrix} -4 & 2 & 1 \\ -4 & -2 & -1 \\ 4 & 2 & -1 \\ 4 & -2 & 1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 430 & -94 \\ -436 & 12 \\ -361 & -22 \\ 367 & 104 \end{bmatrix}$$

The first call to PLSR fixes the number of components to 3 for both response variables, and the second call sets `cv` = `.true.` in order to perform *K*-fold cross validation. Note that because the number of folds is equal to *n*, PLSR performs leave-one-out (LOO) cross–validation.

```
        use plsr_int
        use umach_int
        implicit none

        integer, parameter :: n=4, p=3, h=2
        integer :: ncomps, nobs, iprint, nout
        logical :: cvflag
        real(kind(1e0)) :: x(n,p), y(n,h), yhat(n,h), coef(p,h), se(p,h)

        data x/-4.0, -4.0, 4.0, 4.0, 2.0, -2.0, 2.0, -2.0, 1.0, -1.0, &
              -1.0, 1.0/
        data y/430.0, -436.0, -361.0, 367.0, -94.0, 12.0, -22.0, 104.0/
        !                                Print out informational error.
        call erset(2, 1, 0)
        call umach(2,nout)
        cvflag = .false.
        iprint = 1
        ncomps = 3
        nobs = min(size(y,1), size(x,1))

        write(nout,*) 'Example 1a: no cross-validation, request', &
```

```
         ncomps, 'components.'
      call plsr(y, x, coef, ncomps=ncomps, cv=cvflag, yhat=yhat, &
         iprint=iprint, se=se)

      write(nout,*)
      write(nout,*) 'Example 1b: cross-validation '
      call plsr(y, x, coef, k=nobs, iprint=iprint, yhat=yhat, se=se)
      end
```

Output

```
 Example 1a: no cross-validation, request           3 components.

     PLS Coeff
         1       2
1     0.7    10.3
2    17.2   -29.0
3   398.5     5.0

    Predicted Y
         1       2
1   430.0   -94.0
2  -436.0    12.0
3  -361.0   -22.0
4   367.0   104.0

    Std. Errors
         1       2
1   131.5     5.1
2   263.0    10.3
3   526.0    20.5

 *** ALERT    ERROR 1 from s_plsr.  For response 2, residuals converged in 2
 ***          components, while 3 is the requested number of components.

 Example 1b: cross-validation

Cross-validated results for response  1:

 Comp      PRESS
  1   542903.8
  2   830049.8
  3   830049.8

The best model has  1 component(s).

Cross-validated results for response  2:

 Comp      PRESS
   1   5079.6
   2   1263.4
   3   1263.4

The best model has  2 component(s).

     PLS Coeff
         1       2
1    15.9    12.7
2    49.2   -23.9
```

```
      3   371.1     0.6

         Predicted Y
              1       2
      1   405.8   -97.8
      2  -533.3    -3.5
      3  -208.8     2.2
      4   336.4    99.1

         Std. Errors
              1       2
      1   134.1     7.1
      2   269.9     3.8
      3   478.5    19.5

      *** ALERT    ERROR 1 from s_plsr.  For response 2, residuals converged in 2
      ***          components, while 3 is the requested number of components.
```

Example 2

The data, as appears in S. Wold, et.al. (2001), is a single response variable, the "free energy of the unfolding of a protein", while the predictor variables are 7 different, highly correlated measurements taken on 19 amino acids.

```
          use plsr_int
          use umach_int
          implicit none

          integer, parameter :: n=19, p=7, h=1
          integer :: ncomps, iprint, nout
          logical :: cvflag, scale
          real(kind(1e0)) :: x(n,p), y(n,h), yhat(n,h), coef(p,h), se(p,h)

          data x/0.23, -0.48, -0.61, 0.45, -0.11, -0.51, 0.0, 0.15, 1.2, &
                 1.28, -0.77, 0.9, 1.56, 0.38, 0.0, 0.17, 1.85, 0.89, &
                 0.71, 0.31, -0.6, -0.77, 1.54, -0.22, -0.64, 0.0, 0.13, &
                 1.8, 1.7, -0.99, 1.23, 1.79, 0.49, -0.04, 0.26, 2.25, &
                 0.96, 1.22, -0.55, 0.51, 1.2, -1.4, 0.29, 0.76, 0.0, &
                 -0.25, -2.1, -2.0, 0.78, -1.6, -2.6, -1.5, 0.09, -0.58, &
                 -2.7, -1.7, -1.6, 254.2, 303.6, 287.9, 282.9, 335.0, &
                 311.6, 224.9, 337.2, 322.6, 324.0, 336.6, 336.3, 366.1, &
                 288.5, 266.7, 283.9, 401.8, 377.8, 295.1, 2.126, 2.994, &
                 2.994, 2.933, 3.458, 3.243, 1.662, 3.856, 3.35, 3.518, &
                 2.933, 3.86, 4.638, 2.876, 2.279, 2.743, 5.755, 4.791, &
                 3.054, -0.02, -1.24, -1.08, -0.11, -1.19, -1.43, 0.03, &
                 -1.06, 0.04, 0.12, -2.26, -0.33, -0.05, -0.31, -0.4, &
                 -0.53, -0.31, -0.84, -0.13, 82.2, 112.3, 103.7, 99.1, &
                 127.5, 120.5, 65.0, 140.6, 131.7, 131.5, 144.3, 132.3, &
                 155.8, 106.7, 88.5, 105.3, 185.9, 162.7, 115.6/


          data y/8.5, 8.2, 8.5, 11.0, 6.3, 8.8, 7.1, 10.1, 16.8, 15.0, &
                 7.9, 13.3, 11.2, 8.2, 7.4, 8.8, 9.9, 8.8, 12.0/

          call umach(2, nout)

          cvflag = .false.
          iprint =  2
```

```
      ncomps = 7
      scale = .true.

      write(nout,*) 'Example 2a:  no cross-validation, request', &
                    ncomps,'components.'
      call plsr(y, x, coef, ncomps=ncomps, cv=cvflag, scale=scale, &
                iprint=iprint, yhat=yhat, se=se)

      write(nout,*)
      write(nout,*) 'Example 2b: cross-validation '
      call plsr(y, x, coef, scale=scale, iprint=iprint, &
                yhat=yhat, se=se)
      end
```

Output

```
 Example 2a:  no cross-validation, request 7 components.

  Standard PLS Coefficients
         1  -5.459
         2   1.668
         3   0.625
         4   1.430
         5  -2.550
         6   4.862
         7   4.859

  PLS Coeff
  1  -20.04
  2    4.63
  3    1.42
  4    0.09
  5   -7.27
  6   20.90
  7    0.46

  Predicted Y
   1    9.38
   2    7.30
   3    8.09
   4   12.02
   5    8.79
   6    6.76
   7    7.24
   8   10.44
   9   15.79
  10   14.35
  11    8.41
  12    9.95
  13   11.52
  14    8.64
  15    8.23
  16    8.40
  17   11.12
  18    8.97
  19   12.39

  Std. Errors
   1   3.599
```

```
      2   2.418
      3   0.812
      4   3.214
      5   1.641
      6   3.326
      7   3.529

     Corrected Std. Errors
           1   13.21
           2    6.71
           3    1.84
           4    0.20
           5    4.68
           6   14.30
           7    0.33

     Variance Analysis
     ============================================
     Pctge of Y variance explained
     Component    Cum. Pctge
      1         39.7
      2         42.8
      3         58.3
      4         65.1
      5         68.2
      6         75.1
      7         75.5
     ============================================
     Pctge of X variance explained
     Component    Cum. Pctge
      1         64.1
      2         96.3
      3         97.4
      4         97.9
      5         98.2
      6         98.3
      7         98.4

     Example 2b: cross-validation

    Cross-validated results for response  1:

     Comp      PRESS
        1   0.669
        2   0.675
        3   0.885
        4   1.042
        5   2.249
        6   1.579
        7   1.194

    The best model has  1 component(s).

     Standard PLS Coefficients
           1   0.1486
           2   0.1639
           3  -0.1492
           4   0.0617
           5   0.0669
           6   0.1150
```

```
         7   0.0691

 PLS Coeff
1   0.5453
2   0.4546
3  -0.3384
4   0.0039
5   0.1907
6   0.4942
7   0.0065

Predicted Y
 1    9.18
 2    7.97
 3    7.55
 4   10.48
 5    8.75
 6    7.89
 7    8.44
 8    9.47
 9   11.76
10   11.80
11    7.37
12   11.14
13   12.65
14    9.96
15    8.61
16    9.27
17   13.47
18   11.33
19   10.71

Std. Errors
1   0.06312
2   0.07055
3   0.06272
4   0.03911
5   0.03364
6   0.06386
7   0.03955

Corrected Std. Errors
     1   0.2317
     2   0.1957
     3   0.1423
     4   0.0025
     5   0.0959
     6   0.2745
     7   0.0037

Variance Analysis
============================================
Pctge of Y variance explained
Component    Cum. Pctge
 1         39.7
============================================
Pctge of X variance explained
Component    Cum. Pctge
 1         64.1
```

Correlation

Routines

Usage Notes

This chapter is concerned with measures of correlation for bivariate data. The usual multivariate measures of correlation and covariance for continuous random variables are produced by routine CORVC. For data grouped by some auxiliary variable, routine COVPL can be used to compute the pooled covariance matrix along with the means for each group. Partial correlations or covariances, given the correlation or covariance matrix computed from CORVC or COVPL, are computed by PCORR. Routine RBCOV computes robust estimates of the covariance matrix and mean vector. If data are grouped by some auxiliary variable, routine RBCOV can also be used to estimate the pooled covariance matrix and means for each group. The remaining routines are concerned with rank and/or discrete data. General references for these routines are Conover (1980) or Gibbons (1971).

CTRHO and TETCC produce measures of correlation for contingency tables. In CTRHO, the inverse normal scores obtained from the row and column marginal distributions are assumed known, and the correlation coefficient is estimated by assuming bivariate normality. In TETCC, a 2×2 table is produced from continuous input data using estimates for the sample medians. The correlation coefficient is estimated from the resulting 2×2 table.

If one of the variables is dichotomous while the second variable can be ranked, the routines BSPBS or BSCAT can be used. The difference between these routines is in whether the class values for the ranked variable are given by the user (BSPBS) or are estimated as inverse normal scores from the marginal cumulative distribution (BSCAT). Routine CNCRD computes Kendall's coefficient of concordance, and routine KENDL computes Kendall's rank correlation coefficient τ. Probabilities for τ are computed by routine KENDP.

Other Routines

Other IMSL routines compute measures of correlation or association and may be of interest. Routine CTTWO described in Chapter 5, "Categorical and Discrete Data Analysis," computes measures of association for the 2×2 contingency table. Routine CTCHI in the same chapter computes measures of association for the general $r \times c$ contingency table. Routine CDIST in Chapter 11, "Cluster Analysis," computes measures of similarity and dissimilarity, including the correlation coefficient. Measures of multivariate association or correlation are computed in Chapter 2, "Regression," and in "Independence of Sets of Variables and Canonical Correlation Analysis."

CORVC

Computes the variance-covariance or correlation matrix.

Required Arguments

NVAR — Number of variables. (Input)
The weight or frequency variables, if used, are not counted in NVAR.

X — |NROW| by NVAR + *m* matrix containing the data, where *m* is 0, 1, or 2 depending on whether any column(s) of X correspond to weights and/or frequencies. (Input)

COV — NVAR by NVAR matrix containing either the correlation matrix (possibly with the standard deviations on the diagonal), the variance-covariance matrix, or the corrected sums of squares and crossproducts matrix, as controlled by the COV option, ICOPT. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
The elements of COV correspond to the columns of X, except for the columns of X containing weights or frequencies (see XMEAN).

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of CORVC for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to CORVC will be made. Initialization and updating for the NROW observations are performed. The means (in XMEAN) are output correctly, but the quantities output in COV are intermediate results.
2	This is an intermediate invocation of CORVC, and updating for the NROW observations is performed.
3	This is the final invocation of this routine. If NROW is not zero, updating is performed. The wrap-up computations for COV are performed.

It is possible to call CORVC twice in succession with IDO = 3 in order to first compute covariances (ICOPT = 1) and then compute correlations (ICOPT = 2 or 3). This ability is most important when pairwise deletion of missing values is used (MOPT = 3). The workspace arrays (or the workspace) must not be altered in between calls.

NROW — The absolute value of **NROW** is the number of rows of data currently input in **X**. (Input)
Default: **NROW** = size (**X**,1).
NROW may be positive, zero, or negative. Negative **NROW** means that the **-NROW** rows of data are to be deleted from (most aspects of) the analysis. This should be done only if **IDO** is 2 or 3 and the wrap-up computations for **COV** have not been performed. When a negative value is input for **NROW**, it is assumed that each of the **-NROW** rows of **X** has been input (with positive **NROW**) in previous invocations of **CORVC**. Use of negative values of **NROW** should be made with care since it is possible that a constant variable in the remaining data will not be recognized as such.

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive **IFRQ**, column **IFRQ** of **X** contains the frequencies.
Default: **IFRQ** = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. For positive **IWT**, column **IWT** of **X** contains the weights. Observations with zero weight are counted as observations in the frequencies, but do not contribute to the means, variances, covariances, or correlations. Observations with negative weights are missing.
Default: **IWT** = 0.

MOPT — Missing value option. (Input)
NaN (not a number) is interpreted as the missing value code, and any value in **X** equal to NaN is excluded from the computations. If **MOPT** is positive, various pairwise exclusion methods are used. See routine **AMACH**/**DMACH** in the Reference Material.
Default: **MOPT** = 0.

MOPT	Action
0	The exclusion is listwise. (The entire row of X is excluded if any of the values of the row is equal to the missing value code.)
1	Raw crossproducts are computed from all valid pairs and means, and variances are computed from all valid data on the individual variables. Corrected crossproducts, covariances and correlations are computed using these quantities.
2	Raw crossproducts, means and variances are computed as in the case of MOPT = 1. However, corrected crossproducts and covariances are computed only from the valid pairs of data. Correlations are computed using these covariances and the variances from all valid data.
3	Raw crossproducts, means, variances, and covariances are computed as in the case of MOPT = 2. Correlations are computed using these covariances, but the variances used are computed only from the valid pairs of data.

ICOPT — COV option. (Input)
Default: ICOPT = 0.

ICOPT	Action
0	COV contains the variance-covariance matrix.
1	COV contains the corrected sums of squares and crossproducts matrix.
2	COV contains the correlation matrix.
3	COV contains the correlation matrix, except for the diagonal elements, which are the standard deviations.

XMEAN — Vector of length NVAR containing the variable means. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
The elements of XMEAN correspond to the columns of X, except that if weights and/or frequencies are used, the elements of XMEAN beyond the IWT or IFRQ element are shifted relative to the columns of X.

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

INCD — Incidence matrix. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
If MOPT is zero, INCD is 1 by 1, and contains the number of valid observations. If MOPT is positive, INCD is NVAR by NVAR and contains the numbers of pairs of valid observations that are used in calculating the crossproducts for COV.

LDINCD — Leading dimension of INCD exactly as specified in the dimension statement in the calling program. (Input)
Default: LDINCD = size(INCD,1).

NOBS — Total number of observations (that is, the total of the frequencies). (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
If MOPT = 0, observations with missing values are not included in NOBS. For other values of MOPT, all observations are included except for observations with missing values for the weight or the frequency.

NMISS — Total number of observations that contain any missing values. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

SUMWT — Sum of the weights of all observations that are processed. (Output, if IDO = 0, or 1; input/output, if IDO = 2 or 3)
If MOPT = 0, observations with missing values are not included in SUMWT. For other values of MOPT, all observations are included except for observations with missing values for the weight or the frequency.

FORTRAN 90 Interface

Generic: CALL CORVC (NVAR, X, COV [, ...])

Specific: The specific interface names are S_CORVC and D_CORVC.

FORTRAN 77 Interface

Single: CALL CORVC (IDO, NROW, NVAR, X, LDX, IFRQ, IWT, MOPT, ICOPT, XMEAN, COV, LDCOV, INCD, LDINCD, NOBS, NMISS, SUMWT)

Double: The double precision name is DCORVC.

Description

Routine CORVC computes estimates of correlations, covariances, or sums of squares and crossproducts for a data matrix X. Weights and frequencies are allowed but not required. Also allowed are listwise or pairwise deletion of missing values. Routine CORVC is an "IDO routine," so it may be called with all of the data in one invocation, or it may be called in several invocations with some (or none) of the data input during each call. By setting NROW to a negative integer, observations that have previously been added to the covariance/correlation statistics may be deleted from the statistics. Exercise care with this option, however, since the program may not be able to detect constant variables when negative NROW is used.

The means, (corrected) sums of squares, and (corrected) sums of crossproducts are computed using the method of provisional means. Let

$$\overline{x}_{ki}$$

denote the mean based upon i observations for the k-th variable, f_i denote the frequency of the i-th observation, w_i denote the weight of the i-th observation, and let c_{jki} denote the sum of crossproducts (or sum of squares if $j = k$) based upon i observations. Then, the method of provisional means finds new means and sums of crossproducts as follows:

The means and crossproducts are initialized as:

$$\overline{x}_{k0} = 0.0 \quad k = 1, \ldots, p$$
$$c_{jk0} = 0.0 \;\; j, k = 1, \ldots, p$$

where p denotes the number of variables. Letting $x_{k(i+1)}$ denote the k-th variable on observation $i + 1$, each new observation leads to the following updates for

$$\bar{x}_{ki}$$

and c_{jki} using update constant r_{i+1}:

$$r_{i+1} = \frac{f_{i+1} w_{i+1}}{\sum_{j=1}^{i+1} f_i w_i}$$

$$\bar{x}_{k(i+1)} = \bar{x}_{ki} + (x_{k(i+1)} - \bar{x}_{ki}) r_{i+1}$$

$$c_{jk(i+1)} = c_{jki} + w_{i+1} f_{i+1} (x_{j(i+1)} - \bar{x}_{ji})(x_{k(i+1)} - \bar{x}_{ki})(1 - r_{i+1})$$

If there is no weight variable, weights of 1.0 are used. If there is no frequency column, frequencies of 1.0 are used. Means and variances are computed based upon all of the valid data for each variable or, if required, based upon all of the valid data for each pair of variables.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2RVC/DC2RVC.The reference is:

   ```
   CALL C2RVC (IDO, NROW, NVAR, X, LDX, IFRQ, IWT, MOPT, ICOPT, XMEAN, COV, LDCOV,
      INCD, LDINCD, NOBS, NMISS, SUMWT, WK)
   ```

   The additional argument is:

   WK — Workspace of the length specified in the table below. WK should not be changed between calls to C2RVC.

   The workspace may contain statistics of interest. Let

   $$m = \text{NVAR}$$

   $$k = m(m+1)/2$$

   Statistics that are stored in the workspace that are part of symmetric matrices are stored in symmetric storage mode, i.e., only the lower triangular elements are stored. The workspace utilization is :

MOPT	IWT	Start	Length	Contents
All	All	1	m	Indicators of constant data
All	All	m + 1	m	First nonmissing data
0	All	2m+1	m	Deviation from temporary mean
0	Positive	3m + 1	1	Sum of weights
1, 2	All	2m + 1	m^2	Pairwise means
1, 2	Positive	2m + m^2 + 1	k	Pairwise sums of weights
3	All	2m + 1	m^2	Pairwise means

MOPT	IWT	Start	Length	Contents
3	0	$2m + m^2 + 1$	m^2	Pairwise sums of products
3	Positive	$2m + m^2 + 1$	k	Pairwise sums of weights
3	Positive	$2m + k + m^2 + 1$	m^2	Pairwise sums of products

2. Informational errors

Type	Code	Description
3	12	The sum of the weights is zero. The means, variance and covariances are set to NaN.
3	13	The sum of the weights is zero. The means and correlations are set to NaN.
3	14	Correlations are requested but the observations on a variable are constant. The pertinent correlations are set to NaN.
3	15	Variances and covariances are requested but fewer than two valid observations are present for some variables. The corresponding variances or covariances are set to NaN.
3	16	Pairwise correlations are requested but the observations on a variable are constant. The pertinent correlations are set to NaN.
3	17	Correlations are requested but fewer than two valid observations are present for some variables. The corresponding variances or covariances are set to NaN.
4	10	More observations have been deleted than were originally entered.
4	11	More observations have been deleted from COV(*i*, *j*) than were originally entered. INCD(*i*, *j*) is less than zero.
4	18	Different observations have been deleted from COV(*i*, *j*) than were originally entered. COV(*i*, *j*) is less than zero.

Usage Notes

In **CORVC**, each observation x_{ki} with weight w_i is assumed to have mean μ_k and variance

$$\sigma_k^2 / w_i$$

With these assumptions, **CORVC** uses the following definition of a sample mean:

$$\bar{x}_k = \frac{\sum_{i=1}^{n_r} f_i w_i x_{ki}}{\sum_{i=1}^{n_r} f_i w_i}$$

where n_r is the number of cases. The following formula defines the sample covariance, s_{jk}, between variables j and k:

$$s_{jk} = \frac{\sum_{i=1}^{n} f_i w_i (x_{ji} - \bar{x}_j)(x_{ki} - \bar{x}_k)}{\left(\sum_{i=1}^{n} f_i\right) - 1}$$

The sample correlation between variables j and k, r_{jk}, is defined as:

$$r_{jk} = \frac{s_{jk}}{\sqrt{s_{jj} s_{kk}}}$$

Examples

Example 1

The first example illustrates the use of **CORVC** when inputing all of the data at once. The first 50 observations in the Fisher iris data (see routine **GDATA**, Chapter 19, "Utilities") are used. Note in this example that the first variable is constant over the first 50 observations.

```
          USE GDATA_INT
          USE UMACH_INT
          USE CORVC_INT
          USE WRRRN_INT
          USE WRIRN_INT

          IMPLICIT   NONE
          INTEGER    LDCOV, LDINCD, LDX, NVAR
          PARAMETER  (LDCOV=5, LDINCD=1, LDX=150, NVAR=5)
     !
          INTEGER   INCD(LDINCD,1), NMISS, NOBS, NOUT, NROW, NV
          REAL       COV(LDCOV,NVAR), SUMWT, X(LDX,NVAR), XMEAN(NVAR)
     !
          CALL GDATA (3, X, NOBS, NV)
     !
          CALL UMACH (2, NOUT)
          NROW  = 50
     !
          CALL CORVC (NVAR, X, COV, NROW=NROW, XMEAN=XMEAN, INCD=INCD, &
          NOBS=NOBS, NMISS=NMISS, SUMWT=SUMWT)
     !
```

```
      CALL WRRRN ('XMEAN', XMEAN, 1, NVAR, 1, 0)
      CALL WRRRN ('COV', COV)
      CALL WRIRN ('INCD', INCD)
      WRITE (NOUT,*) ' NOBS = ', NOBS, ' NMISS = ', NMISS, ' SUMWT = ', &
                     SUMWT
      END
```

Output

```
                  XMEAN
      1        2        3        4        5
  1.000    5.006    3.428    1.462    0.246

                           COV
              1        2        3        4        5
      1   0.0000   0.0000   0.0000   0.0000   0.0000
      2   0.0000   0.1242   0.0992   0.0164   0.0103
      3   0.0000   0.0992   0.1437   0.0117   0.0093
      4   0.0000   0.0164   0.0117   0.0302   0.0061
      5   0.0000   0.0103   0.0093   0.0061   0.0111

  INCD
    50
  NOBS =   50 NMISS =   0 SUMWT =     50.0000
```

Example 2

In the second example, the `IDO` option is used. After the initialization step in which `IDO` = 1, the first 53 observations in the Fisher iris data are input, one observation at a time. The last three observations input are then deleted from the covariances by setting `NROW` = `-` 1. Finally, the wrap-up step is accomplished by calling `CORVC` with `IDO` = 3. The output is identical to the output above.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCOV, LDINCD, LDX, LDY, NVAR
      PARAMETER  (LDCOV=5, LDINCD=1, LDX=150, LDY=1, NVAR=5)
!
      INTEGER   I, IDO, INCD(LDINCD,1), NMISS, NOBS, NOUT, NROW, NV
      REAL       COV(LDCOV,NVAR), SUMWT, X(LDX,NVAR), XMEAN(NVAR), &
      Y(LDY,NVAR)
!
      CALL GDATA (3, X, NOBS, NV)
!
      CALL UMACH (2, NOUT)
!
!
      IDO  = 1
      NROW = 0
!                                Initialization
      CALL CORVC (NVAR, Y, COV, IDO=IDO, NROW=NROW, XMEAN=XMEAN, &
      INCD=INCD, NOBS=NOBS, NMISS=NMISS, SUMWT=SUMWT)
!
      IDO  = 2
      NROW = 1
```

```
!                                  Add the observations
      DO 10  I=1, 53
         CALL SCOPY (NVAR, X(I:,1), LDX, Y(1:,1), 1)
         CALL CORVC (NVAR, Y, COV, IDO=IDO, NROW=NROW, XMEAN=XMEAN, &
      INCD=INCD, NOBS=NOBS, NMISS=NMISS, SUMWT=SUMWT)
   10 CONTINUE
!                                  Delete the last 3 added
      NROW = -1
      DO 20  I=51, 53
         CALL SCOPY (NVAR, X(I:,1), LDX, Y(1:,1), 1)
         CALL CORVC (NVAR, Y, COV, IDO=IDO, NROW=NROW, XMEAN=XMEAN, &
      INCD=INCD, NOBS=NOBS, NMISS=NMISS, SUMWT=SUMWT)
   20 CONTINUE
!                                  Wrap-up
      IDO  = 3
      NROW = 0
      CALL CORVC (NVAR, Y, COV, IDO=IDO, NROW=NROW, XMEAN=XMEAN, INCD=INCD,&
      NOBS=NOBS, NMISS=NMISS, SUMWT=SUMWT)
      CALL WRRRN ('XMEAN', XMEAN, 1, NVAR, 1)
      CALL WRRRN ('COV', COV)
      CALL WRIRN ('INCD', INCD)
      WRITE (NOUT,*) ' NOBS = ', NOBS, ' NMISS = ', NMISS, ' SUMWT = ', &
            SUMWT
      END
```

Output

```
                  XMEAN
      1       2       3       4       5
  1.000   5.006   3.428   1.462   0.246

                        COV
          1       2       3       4       5
  1   0.0000  0.0000  0.0000  0.0000  0.0000
  2   0.0000  0.1242  0.0992  0.0164  0.0103
  3   0.0000  0.0992  0.1437  0.0117  0.0093
  4   0.0000  0.0164  0.0117  0.0302  0.0061
  5   0.0000  0.0103  0.0093  0.0061  0.0111

  INCD
    50
  NOBS =   50 NMISS =   0 SUMWT =     50.0000
```

COVPL

Computes a pooled variance-covariance matrix from the observations.

Required Arguments

NROW — The absolute value of NROW is the number of rows of X that contain an observation. (Input)
NROW may be positive, zero, or negative. Negative NROW means that the -NROW rows of data are to be deleted from (most aspects of) the analysis. This should be done only if IDO is 2 or 3 and the wrap-up computations for COV have not been performed. When a negative value is input for NROW, it is assumed that each of the -NROW rows of X has been input (with positive NROW) in previous invocations of CORVC. Use of negative values of NROW should be made with care since it is possible that a constant variable in the remaining data will not be recognized as such.

NVAR — Number of variables to be used in computing the covariance matrix. (Input)
The weight, frequency or group variables, if used, are not counted in NVAR.

X — |NROW| by NVAR + *m* matrix containing the data. (Input)
The number of columns of X that are used is NVAR + *m*, where *m* is 0, 1, 2, or 3 depending upon whether any columns in X contain frequencies, weights or group numbers.

NGROUP — Number of groups in the data. (Input)

COV — NVAR by NVAR matrix of covariances. (Output, if IDO = 0 or 1; input/ output, if IDO = 2 or 3)

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of COVPL and all the data are input at once.
1	This is the first invocation of COVPL with this data, and additional calls will be made. Initialization of program variables and updating for the NROW observations are performed.
2	This is an intermediate invocation of COVPL, and updating for the NROW observations is performed.
3	All statistics are updated for the NROW observations. The covariance matrix is computed.

NCOL — Number of columns in matrix X.
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IND — Vector of length NVAR containing the column numbers in X to be used in computing the covariance matrices. (Input)
By default: IND(I) = I.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. Positive IFRQ indicates that column number IFRQ of X contains the frequencies. All frequencies should be integer values. The NINT (nearest integer) function is used to obtain integer frequencies if this is not the case.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. Positive IWT means that column IWT of X contains the weights. Negative weights are not allowed.
Default: IWT = 0.

IGRP — Column of X giving the group numbers. (Input)
If IGRP = 0, one group is assumed. If IGRP > 0, then column number IGRP of X contains the group number for the observation. Group numbers must be numbered 1, 2, ..., NGROUP. The NINT function is used to get integer values for the group numbers.
Default: IGRP = NVAR + 1.

NI — Vector of length NGROUP containing the numbers of observations in the groups. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
The *i*-th element of NI contains the number of observations in group *i*.

SWT — Vector of length NGROUP containing the sum of the weights times the frequencies in the groups. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

XMEAN — NGROUP by NVAR matrix. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)
The *i*-th row of XMEAN contains the group *i* variable means.

LDXMEA — Leading dimension of XMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXMEAN = size (XMEAN ,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement of the calling program. (Input)
Default: LDCOV = size (COV,1).

NRMISS — Number of rows of data encountered in calls to **COVPL** containing missing values (NaN) for any of the variables used. (Output, if **IDO** = 0 or 1; input/ output, if **IDO** = 2 or 3)

FORTRAN 90 Interface

Generic: **CALL COVPL** (**NROW**, **NVAR**, **X**, **NGROUP**, **COV** [, ...])

Specific: The specific interface names are **S_COVPL** and **D_COVPL**.

FORTRAN 77 Interface

Single: **CALL COVPL** (**IDO**, **NROW**, **NVAR**, **NCOL**, **X**, **LDX**, **IND**, **IFRQ**, **IWT**, **NGROUP**, **IGRP**, **NI**, **SWT**, **XMEAN**, **LDXMEA**, **COV**, **LDCOV**, **NRMISS**)

Double: The double precision name is **DCOVPL**.

Description

Routine **COVPL** computes the pooled variance-covariance matrix from a matrix of observations. The within-groups means are also computed. Listwise deletion of missing values is assumed so that all observations used are "complete"; in any row of **X**, if an element in the "list" **IND**, **IGRP**, **IFRQ** or **IWT** is missing, then the row is not used. Routine **COVPL** should be used whenever one suspects that the data has been sampled from populations with different means but identical variance-covariance matrices. If these assumptions cannot be made, a different variance-covariance matrix should be estimated within each group.

When **IDO** = 0, the same computations occur as if **COVPL** were consecutively called with **IDO** equal to 1, 2, and 3. For brevity, the following discusses the computations with **IDO** > 0.

When **IDO** = 1 variables are initialized, workspace is allocated, and input variables are checked for errors.

If **NROW** ≠ 0 (for any value of **IDO**), the group observation totals, T_i, for $i = 1,\ldots, g$, where g is the number of groups, are updated for the |**NROW**| observations in **X**. The group totals are computed as:

$$T_i = \sum_j \omega_{ij} f_{ij} x_{ij}$$

where ω_{ij} is the observation weight, x_{ij} is the j-th observation in the i-th group, and f_{ij} is the observation frequency.

Modified Givens rotations (see routines **SROTM** and **SROTMG** in the IMSL MATH/LIBRARY) are used in computing the Cholesky decomposition of the pooled sums of squares and crossproducts matrix. The interested reader is referred to Golub and Van Loan (1983) for details.

The group means and the pooled sample covariance matrix *S* are computed from the intermediate results when **IDO** = 3. These quantities are defined by

$$\bar{x}_{i\bullet} = \frac{T_i}{\sum_j \omega_{ij} f_{ij}}$$

$$S = \frac{1}{\sum_{i,j} f_{ij} - g} \sum_{i,j} \omega_{ij} f_{ij} (x_{ij} - \bar{x}_{i\bullet})(x_{ij} - \bar{x}_{i\bullet})^T$$

Occasionally, the Cholesky factorization, such that $S = U^T U$ where *U* is lower triangular of the pooled sample cross-products matrix, may be desired. *U* may be computed from the output array **COV**, and the workspace array **D** returned in calls to **C2VPL**. The Cholesky factor *U* can be computed prior to calling **C2VPL** with **IDO** = 3 by multiplying the elements in the *i*-th row of **COV** by

$$\sqrt{D_i} \Big/ \sqrt{\sum_{ij} f_{ij} - g.}$$

If subsequent calls to **C2VPL** are to be made, **COV** must not be modified in computing *U*.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2VPL/DC2VPL**. The reference is:

   ```
   CALL C2VPL (IDO, NROW, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, NGROUP, IGRP, NI, SWT,
       XMEAN, LDXMEA, COV, LDCOV, NRMISS, D, OB, XVAL, DIF)
   ```

   The additional arguments are as follows:

   D — Real work vector of length **NVAR**.

   OB — Real work vector of length **NVAR**.

   XVAL — Real work vector of length **NVAR** * **NGROUP**.

   DIF — Real work vector of length **NVAR**.

2. Informational error

Type	Code	Description
3	1	The group number is not between 1 and **NGROUP**. The observation is ignored.

Example

The following example computes a pooled variance-covariance matrix for the Fisher iris data (see routine **GDATA**, Chapter 19, "Utilities"). The first column in this data set is the group indicator. To illustrate the use of the **IDO** argument, multiple calls to **COVPL** are made.

```
!                                  Specifications
      USE GDATA_INT
      USE COVPL_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE

      INTEGER    IFRQ, IGRP, IWT, LDCOV, LDX, LDXMEA, NCOL, NGROUP, &
      NROW, NVAR
      PARAMETER  (IFRQ=0, IGRP=1, IWT=0, LDX=150, NCOL=5, NGROUP=3, &
      NROW=1, NVAR=4, LDCOV=NVAR, LDXMEA=NGROUP)
!
      INTEGER    I, IDO, IND(4), NI(NGROUP), NOBS, NOUT, NRMISS, NV
      REAL       COV(LDCOV,LDCOV), SWT(NGROUP), X(LDX,5), XMEAN(LDXMEA,NVAR)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (3, X, NOBS, NV)
!
      IDO = 1
      CALL COVPL (0, NVAR, X, NGROUP, COV, IDO=IDO, IND=IND, IGRP=IGRP, &
      NI=NI, SWT=SWT, XMEAN=XMEAN, NRMISS=NRMISS)
!                                  Add the observations
      IDO = 2
      DO 10  I=1, NOBS
      CALL COVPL (NROW, NVAR, X(I:, 1:NCOL), NGROUP, COV, IDO=IDO, &
      IND=IND, IGRP=IGRP, NI=NI,SWT=SWT, XMEAN=XMEAN, NRMISS=NRMISS)
   10 CONTINUE
!                                  Summarize the statistics
      IDO = 3
      CALL COVPL (0, NVAR, X, NGROUP, COV, IDO=IDO, IND=IND, IGRP=IGRP, &
      NI=NI,SWT=SWT, XMEAN=XMEAN, NRMISS=NRMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' NRMISS = ', NRMISS
      CALL WRRRN ('XMEAN', XMEAN)
      CALL WRRRN ('COV', COV)
      END
```

Output

```
 NRMISS = 0
                XMEAN
          1       2       3       4
1   5.006   3.428   1.462   0.246
2   5.936   2.770   4.260   1.326
3   6.588   2.974   5.552   2.026

                  COV
          1       2       3       4
1   0.2650   0.0927   0.1675   0.0384
2   0.0927   0.1154   0.0552   0.0327
3   0.1675   0.0552   0.1852   0.0427
4   0.0384   0.0327   0.0427   0.0419
```

PCORR

Computes partial correlations or covariances from the covariance or correlation matrix.

Required Arguments

COR — NVAR by NVAR correlation or covariance matrix. (Input)

NIND — Number of "independent" variables to to be used in the partial correlations. (Input)
If NIND is -1, the independent variables are taken to be the NVAR - NDEP variables not in INDDEP. If NIND is zero, no independent variables are used, and *p*-values for the input dependent (see INDDEP) correlations (or covariances) are computed. The partial correlations (covariances) are the correlations (covariances) between the dependent variables after removing the linear effect of the independent variables. NIND and NDEP cannot simultaneously be -1.

IND — Vector of length NIND containing the column (or row) numbers in COR of the independent variables. (Input, if NIND > 0; not referenced otherwise)
If NIND is negative or zero, IND is not used and can be dimensioned of length 1 in the calling program.

NDEP — Number of variables for which partial correlations (covariances) are desired (the number of "dependent" variables). (Input)
If NDEP is -1, the dependent variables are taken as the NVAR - NIND variables not in IND. NIND and NDEP cannot simultaneously be -1.

INDDEP — Vector of length NDEP containing the indices of the dependent variables. (Input, if NDEP > 0; not referenced otherwise)
If NDEP is 1, INDDEP is not used and can be dimensioned of length 1 in the calling program.

PCOR — Matrix of size *m* by *m* containing the partial correlations or partial covariances. (Output)
m = NDEP if NDEP > 0, and *m* = NVAR - NIND otherwise. If NIND = 0, then COR and PCOR can share the same memory location.

Optional Arguments

NVAR — Number of variables in COR. (Input)
Default: NVAR = size (COR,1).

LDCOR — Leading dimension of **COR** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOR** = size (**COR**,1).

NDF — Number of degrees of freedom in **COR**. (Input)
If the number of degrees of freedom in **COR** varies from element to element, then a conservative choice for **NDF** is the minimum degrees of freedom for all elements in **COR**. If **NDF** is not known, then **NDF** ≤ 0 defaults to **NDF** = 100.
Default: **NDF** = 0.

ICOR — Partial correlations/covariances option. (Input)
Default: **ICOR** = 0.

ICOR	Action
1	Partial correlations are desired.
0	Partial covariances are desired.

Partial correlations can be computed when either a correlation or a covariance matrix in input in **COR**. To compute partial covariances, **COR** must contain a covariance matrix.

LDPCOR — Leading dimension of **PCOR** exactly as specified in the dimension statement of the calling program. (Input)
Default: **LDPCOR** = size (**PCOR**,1).

NDFP — Number of degrees of freedom in the test that the partial correlation (covariance) is zero. (Output)
This will usually be **NDF - NIND** but will be greater than this value if the variables in **IND** are computationally linearly related.

PVAL — Matrix of size *m* by *m* (see **PCOR**) containing the *p*-values for testing the null hypothesis that the associated partial correlation (covariance) is zero. (Output)
The *p*-values reported in **PVAL** assume that the observations from which **COR** was computed follow a multivariate normal distribution and that each element in **COR** has **NDF** degrees of freedom.

LDPVAL — Leading dimension of **PVAL** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDPVAL** = size(**PVAL**, 1).

FORTRAN 90 Interface

Generic: **CALL PCORR** (**COR**, **NIND**, **IND**, **NDEP**, **INDDEP**, **PCOR** [, ...])
Specific: The specific interface names are **S_PCORR** and **D_PCORR**.

FORTRAN 77 Interface

Single: CALL PCORR (NVAR, COR, LDCOR, NDF, ICOR, NIND, IND, NDEP, INDDEP, PCOR, LDPCOR, NDFP, PVAL, LDPVAL)

Double: The double precision name is DPCORR.

Description

Routine PCORR computes partial correlations or partial covariances from an input correlation or covariance matrix. If the "independent" variables (the linear "effect" of the independent variables is removed in computing the partial correlations/covariances) are linearly related to one another, PCORR detects the linearity and eliminates one or more of the independent variables from the list of independent variables. The number of variables eliminated, if any, can be determined from argument NDFP.

Given a correlation or covariance matrix Σ partitioned as

$$\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

Routine PCORR computes the partial covariances (of the standardized variables if Σ is a correlation matrix) as

$$\sum_{22|1} = \sum_{22} - \sum_{21} \sum_{11}^{-1} \sum_{12}$$

If partial correlations are desired, these are computed as

$$P_{22|1} = \left[\operatorname{diag}\left(\sum_{22|1} \right) \right]^{-\frac{1}{2}} \sum_{22|1} \left[\operatorname{diag}\left(\sum_{22|1} \right) \right]^{-\frac{1}{2}}$$

where "diag" denotes the matrix containing the diagonal of its argument along its diagonal with zeros off the diagonal. If $\Sigma_{11}$is singular, then as many variables as required are deleted from Σ_{11} (and Σ_{12}) in order to eliminate the linear dependency(ies). The computations then proceed as above.

The *p*-value for a partial correlation (covariance) tests the null hypothesis $H_0 : \rho_{ij|1} = 0$ ($H_0 : \sigma_{ij|1} = 0$), where $\rho_{ij|1}$($\sigma_{ij|1}$) is the (*i*, *j*) element in matrix $P_{22|1}$ ($\Sigma_{22|1}$). The *p*-values are returned in PVAL. If NDF is not known, the *p*-values are computed as if each element in COR had 100 degrees of freedom. When NDF is not known, the resulting *p*-values may be useful for comparison, but they should not be used as an approximation to the actual probabilities.

Comments

1. Workspace may be explicitly provided, if desired, by use of P2ORR/DP2ORR. The reference is:

   ```
   CALL P2ORR (NVAR, COR, LDCOR, NDF, ICOR, NIND, IND, NDEP, INDDEP, PCOR, LDPCOR,
       NDFP, PVAL, LDPVAL, SXY, SXX, LDSXX, IY, IX)
   ```

   The additional arguments are as follows:

   SXY — Work vector of length *m* * *n*.

   SXX — Work vector of length n^2.

   LDSXX — The value of *n*.

   IY — Work vector of length NVAR.

   IX — Work vector of length NVAR.

2. Informational errors

Type	Code	Description
4	1	COR is incorrectly specified for two independent variables.
4	2	COR is incorrectly specified for an independent variable and a dependent variable.
4	3	COR is incorrectly specified for two dependent variables.
4	4	A computed partial correlation is greater than one.

Example

The following example computes partial correlations from a 9 variable correlation matrix originally given by Emmett (1949). The partial correlations between the remaining variables, after adjusting for variables 1, 3, and 9, are computed. Note in the output that the row and column labels are column numbers, not variable numbers. The corresponding variable numbers would be 2, 4, 5, 6, 7, and 8, respectively.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE PCORR_INT
      USE UMACH_INT
      USE WRRRN_INT
      IMPLICIT NONE

      INTEGER    ICOR, LDCOR, LDP, LDPCOR, NDEP, NDF, NIND, NVAR
      PARAMETER  (ICOR=1, LDCOR=9, LDP=6, LDPCOR=6, NDEP=-1, NDF=30, &
                 NIND=3, NVAR=9)
!
      INTEGER    IND(NIND), INDDEP(1), NDFP, NOUT
      REAL       COR(LDCOR,NVAR), P(LDP,LDP), PCOR(LDPCOR,LDPCOR)
!
      DATA IND/1, 3, 9/
!
      DATA COR/1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, &
          0.639, 0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, &
          0.283, 0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, &
```

```
            0.452, 0.219, 0.504, 0.471, 0.506, 0.355, 1.000, 0.691, &
            0.791, 0.443, 0.285, 0.505, 0.346, 0.418, 0.270, 0.691, &
            1.000, 0.679, 0.383, 0.149, 0.409, 0.426, 0.462, 0.254, &
            0.791, 0.679, 1.000, 0.372, 0.314, 0.472, 0.576, 0.547, &
            0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, 0.434, &
            0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
            0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
            1.000/
!
      CALL PCORR (COR, NIND, IND, NDEP, INDDEP, PCOR, NDF=NDF, &
                  ICOR=ICOR, NDFP=NDFP, PVAL=P)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'The degrees of freedom are ', NDFP
      CALL WRRRN ('PCOR', PCOR)
      CALL WRRRN ('P', P)
!
      END
```

Output

```
 The degrees of freedom are   27

                        PCOR
         1       2       3       4       5       6
1    1.000   0.224   0.194   0.211   0.125  -0.061
2    0.224   1.000   0.605   0.720   0.092   0.025
3    0.194   0.605   1.000   0.598   0.123  -0.077
4    0.211   0.720   0.598   1.000   0.035   0.086
5    0.125   0.092   0.123   0.035   1.000   0.062
6   -0.061   0.025  -0.077   0.086   0.062   1.000

                          P
          1       2       3       4       5       6
1    0.0000  0.2525  0.3232  0.2801  0.5249  0.7576
2    0.2525  0.0000  0.0006  0.0000  0.6417  0.9000
3    0.3232  0.0006  0.0000  0.0007  0.5328  0.6982
4    0.2801  0.0000  0.0007  0.0000  0.8602  0.6650
5    0.5249  0.6417  0.5328  0.8602  0.0000  0.7532
6    0.7576  0.9000  0.6982  0.6650  0.7532  0.0000
```

RBCOV

Computes a robust estimate of a covariance matrix and mean vector.

Required Arguments

WGHTS — User-supplied SUBROUTINE to compute observation weights. The form is
CALL WGHTS (R, NVAR, PERCNT, UU, WW, UP), where

R – Distance of observation from the mean vector at which weights are to be computed. (Input)
UU, WW, and UP are to be computed at distance R.

NVAR – Number of variables. (Input)

PERCNT – Percentage of outliers expected. (Input)

UU – Value of covariance matrix weighting function at distance R. (Output)

WW – Value of mean vector weighting function at distance R. (Output)

UP – Value of first derivative of UU with respect to R. (Output)

WGHTS must be declared EXTERNAL in the calling program. A standard weighting subroutine is provided as routine R5COV/DR5COV. See the Description section for further description of the subroutine WGHTS.

X — NOBS by NVAR + *m* matrix containing the data. (Input)
m is 0, 1, 2, or 3 depending upon whether any columns in X contain frequencies, weights or group numbers.

IND — Vector of length NVAR containing the column numbers in X for which covariances are desired. (Input)

XMEAN — NGROUP by NVAR matrix containing the estimates of the location parameters in each group. (Output, if INIT ≠ 2; input/output, otherwise)
Row *i* of XMEAN contains the location estimates for the variables in group *i*. The columns of XMEAN are in the order specified by IND.

COV — NVAR by NVAR matrix of estimated covariances. (Output, if INIT ≠ 2; input/output, otherwise)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NVAR — Number of variables in the covariance matrix. (Input)
Default: NVAR = size (IND,1).

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. Positive IFRQ indicates that column number IFRQ of X contains the frequencies. All frequencies should be positive integer values. The NINT (nearest integer) function is used to obtain integer frequencies from X.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. Positive IWT means that column IWT of X contains the positive weights. Negative weights are not allowed. Note that weights in column IWT are the proportionality constants used in computing a covariance matrix from observations with proportional covariance matrices. The weights used for robust estimation are computed in the estimation procedure.
Default: IWT = 0.

NGROUP — Number of groups (populations) in the data. (Input)
If the data comes from a single population, NGROUP = 1.
Default: NGROUP = 1.

IGRP — Column of X giving the group numbers. (Input)
If IGRP = 0, one group is assumed. If IGRP > 0, then column number IGRP of X contains the group number for the observation. Group numbers must be
1, 2, …, NGROUP. The NINT intrinsic function is used to obtain integer group numbers
Default: IGRP = 0.

INIT — Estimate initialization option. (Input)
Default: INIT = 0.

INIT	Method
0	Initial estimates are obtained as the usual estimate of a mean vector and of a covariance matrix.
1	Initial estimates based upon the median and interquartile range are used.
2	User input initial estimates are used.

IMTH — Option parameter giving the algorithm to be used in computing the estimates. (Input)
Default: IMTH = 0.

IMTH	Method
0	Huber's conjugate-gradient algorithm is used.
1	Stahel's algorithm is used.

PERCNT — Percentage of gross errors expected in the data. (Input)
PERCNT is in the range from zero to 100 and contains the percentage of outliers expected in the data. PERCNT is usually only used if IMSL supplied weighting subroutine R5COV/DR5COV is used as the subroutine WGHTS.
Default: PERCNT = 0.e0.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
When the maximum absolute change in a location or covariance estimate is less than EPS, convergence is assumed.
Default: EPS = 1.e-4.

NI — Vector of length NGROUP containing the number of observations in each group. (Output)

SWT — Vector of length NGROUP containing the sum of the weights times the frequencies for the observations in each group. (Output)

LDXMEA — Leading dimension of XMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXMEA = size (XMEAN,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement of the calling program. (Input)
Default: LDCOV = size (COV,1).

CNST — Vector of length 4 containing some constants computed by RBCOV. (Output)
CNST(1) contains the constant beta (see the Description section) used to ensure that the estimated covariance matrix has unbiased expectation (for given mean vector) for a multivariate normal density. CNST(2), CNST(3), and CNST(4) are the parameters *a*, *b*, and *c*, respectively, in IMSL-supplied subroutine R5COV/DR5COV. They are set to NaN (not a number) if R5COV is not used.

NRMISS — Number of rows of data in X containing any missing values (NaN, not a number) in the columns IND, IWT, IFRQ, or IGRP. (Output)
Rows of X contributing to NRMISS are ignored in all other computations.

FORTRAN 90 Interface

Generic: CALL RBCOV (WGHTS, X, IND, XMEAN, COV [, ...])

Specific: The specific interface names are S_RBCOV and D_RBCOV.

FORTRAN 77 Interface

Single: CALL RBCOV (WGHTS, NOBS, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, NGROUP, IGRP, INIT, IMTH, PERCNT, MAXIT, EPS, NI, SWT, XMEAN, LDXMEA, COV, LDCOV, CNST, NRMISS)

Double: The double precision name is DRBCOV.

Description

Routine RBCOV computes robust M-estimates of the mean and covariance matrix from a matrix of observations. A pooled estimate of the covariance matrix is computed when multiple groups are present in the input data. M-estimate weights are obtained from a user specified weighting subroutine. In addition, user specified observation weights and frequencies may be given for each row in X. Listwise deletion of missing values is assumed so that all observations used are "complete." In any row of X, if any column in the list determined by IND, IFRQ, IWT, or IGRP is missing, the row is not used.

Let $f(x; \mu_i, \Sigma)$ denote the density of an observation p-vector x in population (group) i with mean vector μ_i, for groups $i = 1, \ldots, \tau$. Let the covariance matrix Σ be such that $\Sigma = R^T R$. If

$$y = \left(R^T\right)^{-1}\left(x - \mu_i\right)$$

then

$$g(y) = |\Sigma|^{1/2} f\left(R^T y + \mu_i; \mu_i, \Sigma\right)$$

It is assumed that $g(y)$ is a spherically symmetric density in p-dimensions.

In RBCOV, Σ and μ_i are estimated as the solutions

$$(\hat{\Sigma}, \hat{\mu}_i)$$

of the estimation equations

$$\frac{1}{n}\sum_{j=1}^{n_i} f_{ij}\omega_{ij}w\left(r_{ij}\right)y_{ij} = 0$$

and

$$\frac{1}{n}\sum_{i=1}^{\tau}\sum_{j=1}^{n_i} f_{ij}\omega_{ij}\left[\left(u\left(r_{ij}\right)y_{ij}y_{ij}^{T} - \beta I_p\right]\right] = 0$$

where i indexes the τ groups, n_i is the number of observations in group i, f_{ij} is the frequency for the j-th observation in group i, ω_{ij} is the observation weight specified in column **IWT** of **X**, I_p is a $p \times p$ identity matrix,

$$r_{ij} = \sqrt{y_{ij}^{T}y_{ij}}$$

$w(r)$ and $u(r)$ are weighting functions specified by the user through subroutine **WGHTS**, and where β is a constant computed by the program to make the expected weighted Mahalanobis distance ($y^{T}y$) equal the expected Mahalanobis distance from a multivariate normal distribution (see Marazzi 1985). The constant β is described more fully below.

Routine **RBCOV** uses one of two algorithms for solving the estimation equations. The first algorithm is discussed in detail in Huber (1981) and is a variant of the conjugate gradient method. The second algorithm is due to Stahel (1981) and is discussed in detail by Marazzi (1985). In both algorithms, correction vectors T_{ki} for the group i means and correction matrix $W_k = I_p + U_k$ for the Cholesky factorization of Σ are found such that the updated mean vectors are given by

$$\hat{\mu}_{i,k+1} = \hat{\mu}_{i,k} + T_{ki}$$

and the updated matrix R is given

$$\hat{R}_{k+1} = W_K\hat{R}_k$$

where k is the iteration number and

$$\hat{\Sigma}_k = R_k^{T}R_k$$

When all elements of U_k and T_{ki} are less that ε = **EPS**, convergence is assumed.

Three methods for obtaining initial estimates are allowed. In the first method, the sample weighted estimate of Σ is computed (using routine **COVPL**). In the second method, estimates based upon the median and the interquartile range are used. Finally, in the last method, the user inputs initial estimates.

Routine **RBCOV** computes estimates for any weighting functions *u* and *w*. The constant **β** is chosen such that $E(u(r)r^2) = p\beta$ where the expectation is with respect to a standard *p*-variate multivariate normal distribution. This yields estimates with the correct expectation for the multivariate normal distribution (for given mean vector). The expectation is computed via integration of estimated spline functions. 200 knots are used on an equally spaced grid from 0.0 to the 99.999 percentile of a

$$\chi^2_p$$

distribution. An error estimate is computed based upon 100 of these knots. If the estimated relative error is greater than 0.001, a warning message is issued. If **β** is not computed accurately (*i*.e., if the warning message is issued), the computed estimates are still optimal, but the scale of the estimated covariance matrix may need to be multiplied by a constant in order for

$$\hat{\Sigma}$$

to have the correct multivariate normal covariance expectation.

The Weighting Subroutine

The name of the weighting subroutine (**WGHTS**) is input into **RBCOV**. User-supplied weights may be used. Alternatively, the user may input the name of the IMSL-supplied subroutine, **S_R5COV** in single precision, or **D_R5COV** in double precision. The weights computed by this subroutine are the "minimax" weights of Huber (1981, pages 231 - 235), with **PERCNT** expected gross errors. Huber's (1981) weighting equations are given by:

$$u(r) = \begin{cases} \frac{a^2}{r^2} & r < a \\ 1 & a \le r \le b \\ \frac{b^2}{r^2} & r > b \end{cases}$$

$$w(r) = \min(1, \frac{c}{r})$$

The constants *a*, *b*, and *c* depend upon the number of variables *p* and upon the expected percentage of gross errors. They are computed by **R5COV** as the zeroes of equations given by Huber and are returned in the array **CNST** from **RBCOV**.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2COV**/**DR2COV**. The reference is:

```
CALL R2COV (WGHTS, NOBS, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, NGROUP, IGRP, INIT,
    IMTH, PERCNT, MAXIT, EPS, NI, SWT, XMEAN, LDXMEA, COV, LDCOV, CNST, NRMISS, D,
    U, GXB, OB, OB1, OB2, SWW, WK, IRN, ISF)
```

The additional arguments are as follows:

D — Work vector of length NVAR.

U — Work vector of length max(*m* * NVAR, NGROUP) * NVAR; where *m* = 2 if IMTH = 0, and *m* = 1 otherwise.

GXB — Work vector of length NVAR * NGROUP.

OB — Work vector of length NVAR.

OB1 — Work vector of length NVAR.

OB2 — Work vector of length NVAR.

SWW — Work vector of length NGROUP.

WK — Work vector of length NOBS.

IRN — Work vector of length NOBS.

ISF — Work vector of length NGROUP.

2. Informational errors

Type	Code	Description
4	1	The derivative of UU with respect R is not correctly specified.

Example

The following example computes estimates of the mean vectors and the pooled covariance matrix for the Fisher iris data (routine GDATA provides these data with the group indicator in the first column.). For comparison, these estimates are first computed via routine COVPL. Routine RBCOV with PERCNT = 0.02 is then used to compute the robust estimates. As can be seen from the output, the resulting estimates are quite similar.

To study the behavior of RBCOV, three observations are made into outliers, and, again, both COVPL and RBCOV are used to compute estimates. When outliers are present, COVPL gives estimates that have clearly been adversely affected, while the estimates produced by RBCOV are close to the estimates produced when no outliers are present.

In both calls to RBCOV, the usual pooled estimates were used for the initial estimates, and IMSL supplied routine R5COV with argument PERCNT = 0.02 was used. Because neither NOBS or PERCNT changed in the two calls, the values returned in CNST are identical. If the percentage of gross errors expected in the data, PERCNT, is not known, a reasonable strategy is to use a value of PERCNT that is such that larger values do not result in significant changes in the estimates.

```
        USE IMSL_LIBRARIES
```

```
      IMPLICIT  NONE
      INTEGER   IGRP, LDCOV, LDX, LDXMEA, MAXIT, NCOL, NGROUP, NOBS, &
      NV, NVAR
      REAL      PERCNT
      PARAMETER  (IGRP=1, NCOL=5, NGROUP=3, NOBS=150, NV=5, NVAR=4, &
      PERCNT=2.0,  LDCOV=NVAR, LDX=NOBS, LDXMEA=NGROUP)
!
      INTEGER    IND(NVAR), NI(NGROUP), NOB1, NOUT, NRMISS, NV1
      REAL       CNST(4), COV(LDCOV,NVAR), SWT(NGROUP), X(LDX,NCOL), &
      XMEAN(NGROUP,NVAR)
      EXTERNAL S_R5COV
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (3, X, NOB1, NV1)
!
      CALL COVPL (NOBS, NVAR, X, NGROUP, COV, IND=IND, IGRP=IGRP, &
      XMEAN=XMEAN)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'COVPL estimates with no outliers'
      CALL WRRRN ('XMEAN', XMEAN)
      CALL WRRRN ('COV', COV, ITRING=1)
!
      CALL RBCOV (S_R5COV, X, IND, XMEAN, COV, NGROUP=NGROUP, &
      IGRP=IGRP, PERCNT=PERCNT, NI=NI, SWT=SWT, CNST=CNST)
!
      WRITE (NOUT,*) 'RBCOV estimates with no outliers'
      CALL WRRRN ('XMEAN', XMEAN)
      CALL WRRRN ('COV', COV, ITRING=1)
      CALL WRRRN ('SWT', SWT, 1, NGROUP, 1)
      CALL WRIRN ('NI', NI, 1, NGROUP, 1)
      CALL WRRRN ('CNST', CNST, 1, 4, 1)
!
      X(1,2)   = 100.0
      X(5,5)   = 100.0
      X(100,3) = -100.0
!
      CALL COVPL (NOBS, NVAR, X, NGROUP, COV, IND=IND, IGRP=IGRP, &
      XMEAN=XMEAN)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'COVPL estimates with three outliers'
      CALL WRRRN ('XMEAN', XMEAN)
      CALL WRRRN ('COV', COV, ITRING=1)
!
      CALL RBCOV (S_R5COV, X, IND, XMEAN, COV, NGROUP=NGROUP, IGRP=IGRP, &
      PERCNT=PERCNT, NI=NI, SWT=SWT, CNST=CNST)
!
      WRITE (NOUT,*) 'RBCOV estimates with three outliers'
      CALL WRRRN ('XMEAN', XMEAN)
      CALL WRRRN ('COV', COV, ITRING=1)
      CALL WRRRN ('SWT', SWT, 1, NGROUP, 1)
      CALL WRIRN ('NI', NI, 1, NGROUP, 1)
      CALL WRRRN ('CNST', CNST, 1, 4,1)
!
      END
```

Output

```
COVPL estimates with no outliers

                XMEAN
          1       2       3       4
 1   5.006   3.428   1.462   0.246
 2   5.936   2.770   4.260   1.326
 3   6.588   2.974   5.552   2.026

                  COV
          1       2       3       4
 1   0.2650   0.0927   0.1675   0.0384
 2            0.1154   0.0552   0.0327
 3                     0.1852   0.0427
 4                              0.0419

RBCOV estimates with no outliers

             XMEAN
       1       2       3       4
 1   4.989   3.411   1.465   0.244
 2   5.951   2.784   4.265   1.324
 3   6.529   2.970   5.489   2.026

                  COV
          1       2       3       4
 1   0.2474   0.0872   0.1535   0.0360
 2            0.1073   0.0538   0.0322
 3                     0.1705   0.0412
 4                              0.0401

          SWT
     1       2       3
50.00   50.00   50.00

       NI
  1    2    3
 50   50   50

               CNST
     1       2       3       4
0.972   0.000   3.093   1.717

COVPL estimates with three outliers

                XMEAN
          1       2       3       4
1   6.904   3.428   1.462   2.242
2   5.936   0.714   4.260   1.326
3   6.588   2.974   5.552   2.026

                  COV
          1       2       3       4
1   60.43    0.30    0.13   -1.28
2           70.53    0.17    0.17
3                    0.19    0.00
4                           66.38
```

```
 RBCOV estimates with three outliers

                XMEAN
         1       2       3       4
1   4.999   3.405   1.468   0.253
2   5.959   2.772   4.271   1.324
3   6.528   2.970   5.489   2.026

                 COV
          1       2       3       4
1   0.2567   0.0885   0.1553   0.0361
2            0.1133   0.0546   0.0324
3                     0.1723   0.0412
4                              0.0424

           SWT
     1       2       3
50.00   50.00   50.00

    NI
 1    2    3
50   50   50

             CNST
     1       2       3       4
0.972   0.000   3.093   1.717
```

CTRHO

Estimates the bivariate normal correlation coefficient using a contingency table.

Required Arguments

TABLE — NROW by NCOL contingency table containing the observed counts. (Input)

RHO — Maximum likelihood estimate of the correlation coefficient. (Output)

VAR — Estimated asymptotic variance of RHO. (Output)

PLTMY — Vector of length NROW + NCOL - 2 containing the points of polytomy of the marginal rows and columns of TABLE. (Output)
The first NROW - 1 elements of PLTMY are the points of polytomy for the rows while the last NCOL - elements are the points of polytomy for the columns.

PROB — NROW by NCOL matrix containing the bivariate normal probabilities corresponding to RHO and PLTMY. (Output)

DRIV — NROW by NCOL matrix containing the partial derivatives of the bivariate normal probability with respect to RHO. (Output)

Optional Arguments

NROW — Number of rows in the table. (Input)
Default: NROW = size (TABLE,1).

NCOL — Number of columns in the table. (Input)
Default: NCOL = size (TABLE,2).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement of the calling program. (Input)
Default: LDTABL = size (TABLE,1).

EPS — Convergence criterion in the iterative estimation. (Input)
RHO will be within EPS of the maximum likelihood estimate unless roundoff errors prevent this precision. EPS must be less than 2. EPS less than or equal to zero defaults to 0.00001.
Default: EPS = .00001.

LDPROB — Leading dimension of **PROB** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDPROB** = size (**PROB**,1).

LDDRIV — Leading dimension of **DRIV** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDDRIV** = size (**DRIV**,1).

FORTRAN 90 Interface

Generic: **CALL CTRHO** (**TABLE**, **RHO**, **VAR**, **PLTMY**, **PROB**, **DRIV** [, ...])
Specific: The specific interface names are **S_CTRHO** and **D_CTRHO**.

FORTRAN 77 Interface

Single: **CALL CTRHO** (**NROW**, **NCOL**, **TABLE**, **LDTABL**, **EPS**, **RHO**, **VAR**, **PLTMY**, **PROB**, **LDPROB**, **DRIV**, **LDDRIV**)
Double: The double precision name is **DCTRHO**.

Description

Routine **CTRHO** computes the maximum likelihood estimate and the asymptotic variance for the correlation coefficient of a bivariate normal population from a two-way contingency table. The maximum likelihood estimates are conditional upon the points of polytomy in the marginal distribution. The resulting estimate for the correlation coefficient should be very close to the unconditional estimate (see Martinson and Hamdan 1972).

The points of polytomy for the row and column marginal probabilities are first computed. If the *i*-th cumulative column marginal is denoted by p_{ci}, then the point of polytomy x_i is given as

$$\Phi^{-1}\left(p_{ci}\right),$$

where Φ denotes the cumulative normal distribution. Let α_i, $i = 0, \ldots, r$ denote these points for the row marginal cumulative probabilities where r = **NROW**, $\alpha_0 = -\infty$, and $\alpha_r = \infty$. Similarly, let β_j, $j = 0, \ldots, c$ denote the points of polytomy for the columns where c = **NCOL**. Then, the probability of the (i, j) cell in the table, p_{ij}, is defined as

$$p_{ij} = \Pr\left(\alpha_{i-1} < X < \alpha_i, \beta_{j-1} < Y < \beta_j\right)$$

where X and Y are the bivariate random variables. Maximum likelihood estimates for the correlation coefficient are computed based upon the bivariate normal density. The likelihood is specified by the multinomial distribution of the table using probabilities p_{ij}.

Routine **CTRHO** assumes that the row random variable decreases with increasing row number while the column variable increases with the column number. If this is not the case, the sign of the estimated correlation coefficient may need to be changed.

Example

The data are taken from Martinson and Hamdan (1972), who attribute it to Karl Pearson. The row variable is head breadth (in millimeters) for a human male while the column variable is the head breadth of his sister. Head breadth increases across the columns and decreases down the row. The row and column variables have been categorized into one of three intervals. The original table is as follows:

1.0	36.5	77.5
52.5	340.5	143.5
40.5	58.0	9.0

Note that routine **CTRHO** can accept other than integer counts. It is not clear from Martinson and Hamdan (1972) how the non-integral counts arise in the table here. The correlation is estimated to be 0.5502.

```
      USE CTRHO_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDDRIV, LDPROB, LDTABL, NCOL, NROW
      PARAMETER  (LDDRIV=3, LDPROB=3, LDTABL=3, NCOL=3, NROW=3)
!
      INTEGER    NOUT
      REAL       DRIV(LDDRIV, NCOL), PLTMY(NROW+NCOL-2),PROB(LDPROB,NCOL), &
      RHO, TABLE(LDTABL, NCOL), VAR
!
      DATA TABLE/1.0, 52.5, 40.5, 36.5, 340.5, 58.0, 77.5, 143.5, 9.0/
!
!
      CALL CTRHO (TABLE, RHO, VAR, PLTMY, PROB, DRIV)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'RHO =', RHO, '    VAR =', VAR
      CALL WRRRN ('PLTMY', PLTMY, 1, NROW+NCOL-2, 1, 0)
      CALL WRRRN ('PROB', PROB)
      CALL WRRRN ('DRIV', DRIV)
      END
```

Output

```
 RHO =   0.549125    VAR =    1.33199E-03
               PLTMY
     1       2       3       4
-1.073   1.030  -1.156   0.516
```

```
                PROB
            1        2        3
1   0.0015   0.0517   0.0983
2   0.0700   0.4398   0.1970
3   0.0523   0.0816   0.0077

                DRIV
            1        2        3
1  -0.0134  -0.0984   0.1118
2  -0.0717   0.1388  -0.0672
3   0.0851  -0.0404  -0.0447
```

TETCC

Categorizes bivariate data and compute the tetrachoric correlation coefficient.

Required Arguments

NROW — The absolute value of NROW is the number of observations currently in X and Y. (Input)
NROW may be positive, zero, or negative. Negative NROW means delete the -NROW observations in X and Y from the analysis. In the usual case, in which all of the data have already been categorized into counts in ICOUNT, NROW should be set to 0 and IDO set to 3.

X — Vector of length |NROW| containing the observations on one variable. (Input)

Y — Vector of length |NROW| containing the observations on the second variable. (Input)

HX — Constant used to categorize values of X. (Input)
See description of ICOUNT.

HY — Constant used to categorize values of Y. (Input)
See description of ICOUNT.

ICOUNT — 2 by 2 matrix containing counts. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
The elements of ICOUNT are the numbers of observations satisfying the following relations:
ICOUNT(1, 1) : X(*i*) < HX and Y(*i*) < HY
ICOUNT(1, 2) : X(*i*) < HX and Y(*i*) ≥ HY
ICOUNT(2, 1) : X(*i*) ≥ HX and Y(*i*) < HY
ICOUNT(2, 2) : X(*i*) ≥ HX and Y(*i*) ≥ HY

NR — Number of real roots in the interval (-1.0, 1.0) of the seventh-degree polynomial used to estimate the correlation coefficient. (Output)

R — Vector of length 7 containing in the first NR positions estimates of the correlation coefficient. (Output)

RS — Estimate of the standard error of the estimates of the correlation coefficient(s). (Output)

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of TETCC, and all the data are input at once in X and Y.
1	This is the first invocation of TETCC with this data, and additional calls will be made. Initialization and updating for the data in X and Y are performed.
2	This is an intermediate invocation of TETCC, and updating for the observations in X and Y is performed.
3	Updating for the observations in X and Y is performed, and the tetrachoric correlation coefficient is computed using the values in ICOUNT.

LDICOU — Leading dimension of **ICOUNT** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDICOU** = size (**ICOUNT**,1).

FORTRAN 90 Interface

Generic: **CALL TETCC** (**NROW**, **X**, **Y**, **HX**, **HY**, **ICOUNT**, **NR**, **R**, **RS** [, ...])
Specific: The specific interface names are **S_TETCC** and **D_TETCC**.

FORTRAN 77 Interface

Single: **CALL TETCC** (**IDO**, **NROW**, **X**, **Y**, **HX**, **HY**, **ICOUNT**, **LDICOU**, **NR**, **R**, **RS**)
Double: The double precision name is **DTETCC**.

Description

Routine **TETCC** computes the tetrachoric correlation coefficient for a bivariate sample, using either the sample itself or a two by two table of counts of the data. The tetrachoric correlation coefficient is taken as the solution to the seventh-degree polynomial obtained from the first seven terms of the expansion given by Kendall and Stuart (1979, page 326).

The standard error estimate results from an approximate, asymptotic expression derived under the assumption of bivariate normality with zero correlation. The zero correlation assumption is not overly restrictive since most uses of this standard error would be in tests of zero correlation.

If all of the data is available, the Pearson product-moment correlation coefficient (which can be computed using routine **CORVC**) is a much better estimate for the population correlation coefficient than is the tetrachoric correlation coefficient. If the counts in **ICOUNT** are all that is available, call **TETCC** with **IDO** = 3 and **NROW** = 0.

Comments

1. Informational errors

Type	Code	Description
3	1	Fewer than 200 observations are used.
3	2	The polynomial used to estimate the correlation coefficient has more than one root in the interval (-1.0, 1.0). It is probable that the numerical precision is not good enough to obtain an estimate.
4	4	The proportion of counts in a row or column is so close to one that the inverse normal cdf cannot be computed.
4	6	The polynomial used to estimate the correlation coefficient has no roots in the interval (-1.0, 1.0). It is probable that the numerical precision is not good enough to obtain an estimate.

2. If data for **X** and **Y** are available, it is better to use the Pearson product moment correlation coefficient (as computed by routine **CORVC**, for example) than to use the tetrachoric correlation coefficient.

3. The tetrachoric correlation coefficient should be considered somewhat questionable if the sample size is less than 200, if the cutpoints **HX** and **HY** are not close to the medians, or if there are multiple roots of the estimating equation in the interval (-1.0, 1.0). Also, the tetrachoric correlation coefficient is a better estimate of the true correlation coefficient if the true coefficient is large in absolute value.

Examples

Example 1

In the first example, the data are counts. The 374 in **ICOUNT**(1, 1) indicates that in the raw data there were 374 pairs having both values less than some cutoff point. The 186 in **ICOUNT**(1, 2) indicates that there were 186 pairs in the raw data for which the first value was less than its cutoff value and the second value was greater than or equal to its cutoff value.

```
      USE UMACH_INT
      USE TETCC_INT

      IMPLICIT   NONE
      INTEGER    I, ICOUNT(2,2), IDO, NOUT, NR, NROW
      REAL       HX, HY, R(7), RS, X(1), Y(1)
!
      CALL UMACH (2, NOUT)
      ICOUNT(1,1) = 374
      ICOUNT(1,2) = 186
      ICOUNT(2,1) = 167
      ICOUNT(2,2) = 203
      IDO         = 3
      NROW        = 0
```

```
      CALL TETCC (NROW, X, Y, HX, HY, ICOUNT, NR, R, RS, IDO=IDO)
      WRITE (NOUT,99998) NR, (R(I),I=1,NR)
99998 FORMAT (' Number of roots (estimates) is ', I1, /, ' ', &
      'Estimate(s) = ',7F10.5)
      WRITE (NOUT,99999) RS
99999 FORMAT (' The estimated standard error is ', F10.5)
      END
```

Output

```
 Number of roots (estimates) is 1
 Estimate(s) =    0.33511
 The estimated standard error is    0.05255
```

Example 2

In this example, some artificial bivariate normal data are generated using IMSL routine RNMVN, and then, the tetrachoric correlation coefficient is computed. Since the mean (and median) of each variable is 0.0, the cutpoints HX and HY are set to 0.0.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    I, ICOUNT(2,2), IRANK, NOUT, NR, NROW
      REAL       COV(2,2), HX, HY, R(7), RS, RSIG(2,2), X(1000), &
                 XY(1000,2), Y(1000)
!
      EQUIVALENCE (X, XY), (Y, XY(1,2))
!
      CALL UMACH (2, NOUT)
!                                Generate random sample from
!                                bivariate normal with correlation
!                                of 0.5.
      COV(1,1) = 1.0
      COV(1,2) = 0.5
      COV(2,1) = 0.5
      COV(2,2) = 1.0
!                                Obtain the Cholesky factorization.
      CALL CHFAC (COV, IRANK, RSIG)
!                                Initialize seed of random number
!                                generator.
      CALL RNSET (123457)
      CALL RNMVN (RSIG, XY)
!
      NROW   = 1000
      HX     = 0.0
      HY     = 0.0
      CALL TETCC (NROW, X, Y, HX, HY, ICOUNT, NR, R, RS)
      WRITE (NOUT,99997) ICOUNT
99997 FORMAT (' ICOUNT = ', 4I4)
      WRITE (NOUT,99998) NR, (R(I),I=1,NR)
99998 FORMAT (' Number of roots (estimates) is ', I1, /, ' ', &
              'Estimate(s) = ',7F10.5)
      WRITE (NOUT,99999) RS
99999 FORMAT (' The estimated standard error is ',F10.5)
      END
```

Output

```
    ICOUNT =  326 163 171 340
    Number of roots (estimates) is 1
    Estimate(s) =    0.49824
    The estimated standard error is    0.04968
```

BSPBS

Computes the biserial and point-biserial correlation coefficients for a dichotomous variable and a numerically measurable classification variable.

Required Arguments

A — 3 by **K** matrix containing the frequencies and the class marks of the measured classification variable. (Input)
The first row of **A** contains frequencies for the classification variable when the dichotomous variable takes on one of its values, and the second row of **A** contains the frequencies when the dichotomous variable takes on its other value. The third row of **A** contains the values (class marks) of the classification variable. The elements of the first two rows of **A** must be nonnegative.

STAT — Vector of length 11 containing various statistics. (Output)

I	**STAT(I)**
1	Total count of the first value of the dichotomous variable (the sum of the first row of A)
2	Total count for the second value
3	Total count (sum of STAT(1) and STAT(2))
4	Mean of the measured variable
5	Mean of the measured variable in the first class of the dichotomy
6	Mean of the measured variable in the second class of the dichotomy
7	Standard deviation of the measured variable
8	Biserial correlation coefficient estimate
9	Standard deviation estimate for the biserial correlation coefficient estimate
10	Asymptotic significance level of the biserial correlation coefficient, that is, the probability of a more extreme value
11	Point-biserial correlation coefficient estimate

Optional Arguments

K — Number of classes for the measured classification variable. (Input)
Default: **K** = size (**A**,2).

LDA — Leading dimension of **A** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDA** = size (**A**,1).

FORTRAN 90 Interface

Generic: **CALL BSPBS** (**A**, **STAT** [, ...])
Specific: The specific interface names are **S_BSPBS** and **D_BSPBS**.

FORTRAN 77 Interface

Single: **CALL BSPBS** (**K**, **A**, **LDA**, **STAT**)
Double: The double precision name is **DBSPBS**.

Description

Routine **BSPBS** computes the biserial and point-biserial correlation coefficient for a dichotomous variable and a numerically measurable (classification) variable. Input to **BSPBS** is a $3 \times K$ array, **A**. The first two rows of **A** contain the frequencies for the dichotomous variable as measured at each level of the classification variable. The third row contains the values (class marks) to be used for the classification variable.

The biserial correlation coefficient should be used in situations where the dichotomous variable and the classification variable are assumed to come from a bivariate normal distribution. If this is not the case (i.e., if the bivariate normal assumption cannot be made), then the point-biserial correlation should be used (see Kendall and Stuart 1979, page 331).

Let $a_{\bullet 1}$ and $a_{\bullet 2}$ denote the total count in rows one and two of **A**, respectively, and let $n = a_{\bullet 1} + a_{\bullet 2}$. Let Φ denote the cumulative normal distribution; let a_{ij}, $i = 1, 2$, $j = 1, \ldots, K$, denote the counts in rows 1 and 2 of **A**, and let x_j denote the values in row 3 of **A**. The biserial correlation coefficient r_b is computed as follows:

$$p = \frac{a_{\bullet 1}}{n}$$

$$z_p = \Phi^{-1}(p)$$

$$\overline{x}_i = \frac{\sum_{j=1}^{k} a_{ij} x_j}{\sum_{j=1}^{k} a_{ij}}$$

$$\overline{x} = \frac{\sum_{j=1}^{k} \left(a_{1j} + a_{2j} \right)}{n}$$

$$s_x^2 = \frac{\sum_{j=1}^{k} \left(a_{1j} + a_{2j} - \overline{x} \right)^2}{n-1}$$

$$r_b = \frac{\overline{x}_1 - \overline{x}_2}{s_x} \frac{p(1-p)}{z_p}$$

$$\text{var}(r_b) \approx \left(\frac{\sqrt{p(1-p)}}{z_p} - r_b^2 \right) \frac{1}{n}$$

Let

$$z = |r_b| / \sqrt{\text{var}\left(r_b \right)}$$

If the underlying distributions are normal with zero correlation, then *z* is asymptotically a standard normal deviate that may be used to test that the correlation is zero. The *p*-value for *z* is reported in **STAT**(10).

The point-biserial correlation coefficient is computed as

$$r_p = \frac{z_p r_b}{\sqrt{p(1-p)}}$$

Example

The example is taken from Kendall and Stuart (1979, page 327). The data involve the classification of criminals as alcoholic (first row) or nonalcoholic for each level of a crimetype classification. The severity of the crime decreases with increasing column number. In the example, the column number is used for the column score. The biserial correlation of -0.17 indicates that more criminals responsible for the most serious crimes tend to be alcoholic.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    K, LDA
          PARAMETER  (K=6, LDA=3)
    !
```

```
      REAL       A(LDA,K), STAT(11)
      CHARACTER  CLABEL(2)*10, RLABEL(11)*10
!
      DATA A/50, 43, 1, 88, 62, 2, 155, 110, 3, 379, 300, 4, &
          18, 14, 5, 63, 144, 6/
      DATA RLABEL/'Count-1', 'Count-2', 'Count', 'Mean(X)', &
          'Mean(X-1)', 'Mean(X-2)', 'S-X', 'r-b', 'std(r-b)', &
          'p-value', 'r-p'/
      DATA CLABEL/'Statistic', '    '/
!
      CALL WRRRN('A', A)
!
      CALL BSPBS (A, STAT)
!
      CALL WRRRL ('    ', STAT, RLABEL, CLABEL, FMT='(W12.8)')
      END
```

Output

```
                          A
          1       2       3       4       5       6
1    50.0    88.0   155.0   379.0    18.0    63.0
2    43.0    62.0   110.0   300.0    14.0   144.0
3     1.0     2.0     3.0     4.0     5.0     6.0
Statistic
Count-1          753.00
Count-2          673.00
Count           1426.00
Mean(X)            3.72
Mean(X-1)          3.55
Mean(X-2)          3.91
S-X                1.31
r-b               -0.17
std(r-b)           0.03
p-value            0.00
r-p               -0.14
```

BSCAT

Computes the biserial correlation coefficient for a dichotomous variable and a classification variable.

Required Arguments

A — 2 by K matrix containing the frequencies. (Input)
The first row of A contains frequencies for the classification variable when the dichotomous variable takes on one of its values, and the second row of A contains the frequencies when the dichotomous variable takes on its other value. No ordering is assumed for the values of the classification variable. The elements of A must be nonnegative.

STAT — Vector of length 5 containing various statistics. (Output)

I	STAT(I)
1	Total count of the first value of the dichotomous variable (the sum of the first row of A)
2	Total count for the second value
3	Total count (sum of STAT(1) and STAT(2))
4	Absolute value of the biserial correlation coefficient
5	Square of the biserial correlation coefficient

Optional Arguments

K — Number of classes for the classification variable. (Input)
Default: K = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

FORTRAN 90 Interface

Generic: CALL BSCAT (A, STAT [, ...])
Specific: The specific interface names are S_BSCAT and D_BSCAT.

FORTRAN 77 Interface

Single: CALL BSCAT (K, A, LDA, STAT)

Double: The double precision name is DBSCAT.

Description

Routine **BSCAT** computes the biserial correlation coefficient for a dichotomous variable and a classification variable. The data are input in a $2 \times k$ array, **A**, where the row indicates the value of the dichotomous variable, and the column indicates the value of the classification variable. In **BSCAT**, column scores are computed as $x_i = \phi^{-1}(a_{1i}/(a_{1i} + a_{2i}))$, and the row score is computed as $y = \phi^{-1}(a_{\bullet 1}/(a_{\bullet 1} + a_{\bullet 2}))$, where $a_{\bullet 1}$ is the sum of the counts in row 1, $a_{\bullet 2}$ is the sum of the counts for row 2, and ϕ denotes the cumulative normal distribution. Let N denote the total number of observations (the sum of the elements of **A**). Then, the biserial correlation is computed as

$$r_b^2 = \frac{\sum_{i=1}^{k} \left(a_{1i} + a_{2i}\right) x_i^2 - Ny^2}{N + \sum_{i=1}^{K} \left(a_{1i} + a_{2i}\right) x_i^2}$$

An underlying bivariate normal distribution is assumed. The validity of the estimate depends heavily upon this assumption.

Example

The example is taken from Kendall and Stuart (1979, page 327). The data involve the classification of criminals as alcoholic (first row) or nonalcoholic for each level of a crimetype classification. The severity of the crime decreases with increasing column number. The absolute value of the biserial correlation is 0.23.

```
      USE WRRRN_INT
      USE BSCAT_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    K, LDA
      PARAMETER  (K=6, LDA=2)
!
      REAL       A(LDA,K), STAT(5)
      CHARACTER  CLABEL(2)*10, RLABEL(5)*10
!
      DATA A/50, 43, 88, 62, 155, 110, 379, 300, 18, 14, 63, 144/
      DATA RLABEL/'Count-1', 'Count-2', 'Count', 'r-b', '(r-b)**2'/
      DATA CLABEL/'Statistic', '    '/
!
      CALL WRRRN ('A', A)
```

```
!
      CALL BSCAT (A, STAT)
!
      CALL WRRRL ('    ', STAT, RLABEL, CLABEL, FMT='(W12.6)')
      END
```

Output

```
                          A
          1       2       3       4       5       6
1    50.0    88.0   155.0   379.0    18.0    63.0
2    43.0    62.0   110.0   300.0    14.0   144.0
Statistic
Count-1          753.00
Count-2          673.00
Count           1426.00
r-b                0.23
(r-b)**2           0.05
```

CNCRD

Calculates and test the significance of the Kendall coefficient of concordance.

Required Arguments

X — NOBS by K matrix containing the data. (Input)
Each column of X is a set of observations (which can be converted to ranks) or a set of ranks.

FUZZ — Value to be used for determining ties. (Input)
If within a column of X, the difference between two elements is less than or equal to FUZZ in absolute value, then the elements are said to be tied.

SUMS — Vector of length NOBS containing the sums of the K ranks in the corresponding row of X. (Output)

STAT — Vector of length 4 containing the output statistics. (Output)

i	STAT(i)
1	*W*, the coefficient of concordance
2	Chi-squared statistic corresponding to *W* with NOBS -1 degrees of freedom
3	Asymptotic probability of exceeding STAT(2) under the null hypothesis of independence
4	Kendall *S* statistic. This is the sum of the squared deviations from the expected sum of the ranks

Optional Arguments

NOBS — Number of observations per set of rankings. (Input)
Default: NOBS = size (SUMS,1).

K — Number of sets of rankings. (Input)
K must be greater than or equal to two.
Default: K = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

FORTRAN 90 Interface

Generic: CALL CNCRD (X, FUZZ, SUMS, STAT [, ...])

Specific: The specific interface names are S_CNCRD and D_CNCRD.

FORTRAN 77 Interface

Single: CALL CNCRD (NOBS, K, X, LDX, FUZZ, SUMS, STAT)

Double: The double precision name is DCNCRD.

Description

Routine CNCRD computes and tests the significance of the Kendall coefficient of concordance.

The coefficient of concordance is computed as follows: Within each of the k sets the n = NOBS observations are ranked. Tied ranks are used for tied observations where two observations are tied if they are within FUZZ of each other. Let x_i denote the sum of the ranks for the i-th observation over the k sets. The mean of the x_i is

$$\overline{x} = k\left(n+1\right)/2$$

Using this mean, compute the sums of squares of the x_i about their mean as

$$S = \sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2.$$

This is the Kendall S statistic (STAT(4)). If there are tied ranks within a set i, compute the adjustment

$$T_i^* = \frac{\sum_j \left(t_j^3 - t_j\right)}{12}$$

where t_j is the number of ties in the j-th group of ties, and the summation is over all tie groups for the set. Kendall's coefficient of concordance, W, is computed as

$$W = \frac{12S}{k^2\left(n^3 - n\right) - k\sum_{i=1}^{k} T_i^*}$$

Kendall's coefficient of concordance is related to the Friedman one-way analysis of variance on ranks chi-squared test statistic T (see IMSL routine FRDMN as

$$W = \frac{T}{n(k-1)}$$

When *n* or *k* is small, tables of the exact distribution of *W* exist. See Owen (1962, pages 396 - 397). The probability reported in STAT(3) is asymptotic. It is only approximate when *k* and *n* are small.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2CRD/DC2CRD. The reference is:

   ```
   CALL C2CRD (NOBS, K, X, LDX, FUZZ, SUMS, STAT, IWK, XWK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length NOBS.

   XWK — Work vector of length NOBS * K.

2. Informational errors

Type	Code	Description
3	6	Within each of the K sets of rankings all observations are tied. STAT(1) - STAT(3) cannot be computed and are set to NaN (not a number).
3	7	The chi-squared degrees of freedom is less than 7. STAT(3) should be regarded with suspicion.

Example

The example is taken from Kendall (1962, pages 97 - 98). It involves ten observations in three sets. The resulting coefficient of concordance, 0.828, is quite large, indicating a strong relationship.

```
      USE WRRRN_INT
      USE CNCRD_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    K, LDX, NOBS
      REAL       FUZZ
      PARAMETER  (FUZZ=0.0001, K=3, LDX=10, NOBS=10)
!
      REAL       STAT(4), SUMS(NOBS), X(LDX,K)
      CHARACTER  CLABEL(2)*11, RLABEL(4)*11
!
      DATA RLABEL/'W', 'Chi-squared', 'p-value', 'S'/
      DATA CLABEL/'Statistic', '  '/
      DATA X/1, 4.5, 2, 4.5, 3, 7.5, 6, 9, 7.5, 10, 2.5, 1, 2.5, 4.5, &
          4.5, 8, 9, 6.5, 10, 6.5, 2, 1, 4.5, 4.5, 4.5, 4.5, 8, 8, 8, &
          10/
!
```

```
      CALL WRRRN ('X', X)
!
      CALL CNCRD (X, FUZZ, SUMS, STAT)
!
      CALL WRRRN ('SUMS', SUMS, 1, NOBS, 1, 0)
      CALL WRRRL ('  %/%/', STAT, RLABEL, CLABEL, FMT='(W10.6)')
      END
```

Output

```
                X
          1       2       3
 1     1.00    2.50    2.00
 2     4.50    1.00    1.00
 3     2.00    2.50    4.50
 4     4.50    4.50    4.50
 5     3.00    4.50    4.50
 6     7.50    8.00    4.50
 7     6.00    9.00    8.00
 8     9.00    6.50    8.00
 9     7.50   10.00    8.00
10    10.00    6.50   10.00
                                    SUMS
   1      2       3       4       5       6       7       8       9      10
5.50   6.50    9.00   13.50   12.00   20.00   23.00   23.50   25.50   26.50

Statistic
W                0.828
Chi-squared     22.349
p-value          0.008
S              591.000
```

KENDL

Computes and test Kendall's rank correlation coefficient.

Required Arguments

X — Vector of length NOBS containing the observations for the first variable. (Input)

Y — Vector of length NOBS containing the observations for the second variable. (Input)

FUZZ — Value used to determine ties in X or Y. (Input)
Two observations are said to be tied if the absolute value of their difference is less than or equal to FUZZ.

STAT — Vector of length 9 containing some output statistics. (Output)
See the "Description" section for full definitions. The output statistics are;

i	STAT(i)
1	Kendall τ_a (assumes no ties)
2	Kendall τ_b (corrects for ties)
3	Ties statistic for variable X
4	Ties statistic for variable Y
5	Statistic *S* corresponding to Kendall's τ
6	Exact probability of achieving a score at least as large as *S*. *S* is not calculated if NOBS is too large (34 on many computers) or there are ties. In either case, STAT(6) is set to NaN (not a number).
7	The same probability as STAT(6) but using a normal approximation. (Set to NaN if NOBS is less than 8.
8	The same probability as STAT(6) but using a continuity correction with a normal approximation. (Set to NaN if NOBS is less than 8.)
9	Index in FRQ corresponding to the frequency of the observed *S* statistic. STAT(9) is not computed when there are ties.

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be 3 or more.
Default: NOBS = size (X,1).

FRQ — Vector of length NOBS * (NOBS - 1)/2 + 1 containing the frequencies of occurrence of the possible values of the statistic *S*, STAT(5), under the null hypothesis of no relationship. (Output) FRQ is not calculated if there are ties or if NOBS is too large (34 on many computers).

FORTRAN 90 Interface

Generic: CALL KENDL (X, Y, FUZZ, STAT [, ...])
Specific: The specific interface names are S_KENDL and D_KENDL.

FORTRAN 77 Interface

Single: CALL KENDL (NOBS, X, Y, FUZZ, STAT, FRQ)
Double: The double precision name is DKENDL.

Description

Routine KENDL performs Kendall's test of the hypothesis of no correlation (independence) by calculating τ_a and τ_b (τ_b handles ties), the Kendall sum *S*, and associated probabilities. The frequencies of occurrence of *S* are also computed if the sample size (NOBS) is not too large.

Kendall's (1962) method is used in computing the τ statistics. Each pair (x_i, y_i) is compared with every other pair (x_j, y_j). The Kendall *S* statistic is incremented if the two pairs are concordant (($x_i > x_j$ and $y_i > y_j$) or ($x_i < x_j$ and $y_i < y_j$)) and decremented if the pairs are discordant (($x_i > x_j$ and $y_i < y_j$) or ($x_i < x_j$ and $y_i > y_j$)). Ties ($x_i = x_j$ or $y_i = y_j$) are not counted. Generally, when ties exist, τ_b is a better measure of correlation than is τ_a. The untied form of the denominator is used to calculate τ_a. That is,

$$\tau_a = \frac{S}{n(n-1)/2}$$

where n = NOBS. Ties enter into the denominator of τ_b as follows:

$$\tau_b = \frac{S}{\sqrt{(D - T_x)(D - T_y)}}$$

where $D = n(n - 1)/2$ and

$$T_x = \sum_i t_i\left(t_i - 1\right)/2$$

where t_i is the number of ties in the *x* variable with the *i*-th tie value. T_y is calculated in a similar manner.

For **NOBS** less than 34 (on many machines other values on machines with a different value for the largest real number that can be represented), the array **FRQ** is computed. **FRQ** contains the frequency distribution of *S* under the null hypothesis of independence. The probability distribution of *S* can be obtained directly from these frequencies by dividing each frequency by the sum of the frequencies. See routine **KENDP** for further discussion on the use of the **FRQ** array.

For a two-sided test, if the appropriate probability *p* of achieving or exceeding *S* is small (less than **α**/2, where **α** is the significance level of the test) or if 1 **-** *p* is small (less than **α**/2), then the two-sided hypothesis of no correlation can be rejected. Alternatively, for small *p* or 1 **-** *p*, the appropriate one-sided hypothesis can be rejected.

For $n > 7$, asymptotic normal probabilities are determined using the fact that

$$z = \frac{S}{\sqrt{\mathrm{var}\left(S\right)}}$$

is approximately standard normal for large *n*. Here,

$$\mathrm{var}(S) = \frac{n(n-1)(2n+5) - \sum_x t_i(t_i-1)(2t_i+5) - \sum_y t_i(t_i-1)(2t_i+5)}{18} +$$
$$\frac{\left[\sum_x t_i(t_i-1)(t_i-2)\right]\left[\sum_y t_i(t_i-1)(t_i-2)\right]}{9n(n-1)(n-2)} +$$
$$\frac{\left[\sum_x t_i(t_i-1)\right]\left[\sum_y t_i\left(t_i-1\right)\right]}{2n(n-1)}$$

where t_i is the number of observations in the *i*-th tie group for the *x* (or *y*) summation variable.

STAT(7) contains the probability associated with the *z* statistic while **STAT**(8) contains the same probability but with the value of *S* reduced by 1. This reduction is for "continuity correction." For *n* less than 25, these probabilities are conservative at the 1% level of significance.

Comments

1. Workspace may be explicitly provided, if desired, by use of **K2NDL/DK2NDL**. The reference is:

   ```
   CALL K2NDL (NOBS, X, Y, FUZZ, STAT, FRQ, IWK, WK, XRNK, YRNK)
   ```

   The additional arguments are as follows:

   IWK — Work vector of length **NOBS**.

WK — Work vector of length (NOBS - 1) * (NOBS - 2)/2 + 1.

XRNK — Work vector of length NOBS.

YRNK — Work vector of length NOBS.

2. Informational errors

Type	Code	Description
3	4	Ties are detected in the two samples. STAT(6) is set to NaN (not a number) and FREQ is not calculated.
3	5	NOBS is less than 8 so the asymptotic normal probabilities are not determined. STAT(7) and STAT(8) are set to NaN (not a number).
3	6	NOBS is too large (34 on many computers). STAT(6) is set to NaN (not a number) and FREQ is not calculated.
4	2	All the elements of X are tied. The output statistics are not defined.
4	3	All the elements of Y are tied. The output statistics are not defined.

Example

In this example, the Kendall test is performed on a sample of size 8. The test fails to reject the null hypothesis of no correlation.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE KENDL_INT
      USE WRRRL_INT
      USE WRRRN_INT

      IMPLICIT   NONE

      REAL       FUZZ
      PARAMETER  (FUZZ=0.0001)
!
      REAL       FRQ(29), STAT(9), X(8), Y(8)
      CHARACTER  CLABEL(2)*10, RLABEL(9)*10
!
      DATA RLABEL/'tau(a)', 'tau(b)', 'ties(X)', 'ties(Y)', &
          'S', 'Pr(S)', 'Pr(S)-n', 'Pr(S)-na', 'IFRQ'/
!
      DATA CLABEL/'Statistic', '    '/
!
      DATA X/6, 4, 7, 3, 8, 1, 5, 2/
      DATA Y/7, 1, 5, 8, 6, 4, 2, 3/
!
      CALL KENDL (X, Y, FUZZ, STAT, FRQ=FRQ)
!
      CALL WRRRL ('STAT', STAT, RLABEL, CLABEL, FMT='(W10.6)')
      CALL WRRRN ('FRQ', FRQ, 1, 29, 1, 0)
      END
```

Output

STAT

Statistic	
tau(a)	0.1429
tau(b)	0.1429
ties(X)	0.0000
ties(Y)	0.0000
S	4.0000
Pr(S)	0.3598
Pr(S)-n	0.3103
Pr(S)-na	0.3553
IFREQ	17.0000

FRQ

1	2	3	4	5	6	7	8
1.0	7.0	27.0	76.0	174.0	343.0	602.0	961.0

9	10	11	12	13	14	15	16
1415.0	1940.0	2493.0	3017.0	3450.0	3736.0	3836.0	3736.0

17	18	19	20	21	22	23	24
3450.0	3017.0	2493.0	1940.0	1415.0	961.0	602.0	343.0

25	26	27	28	29
174.0	76.0	27.0	7.0	1.0

KENDP

Computes the frequency distribution of the total score in Kendall's rank correlation coefficient.

Required Arguments

NOBS — Sample size. (Input)
Must be greater than 1 and less than 34 (56 on some computers).

K — Score for which the probability is to be calculated. (Input)
K must be in the range from minus to plus NOBS * (NOBS - 1)/2, inclusive.

FRQ — Vector of length NOBS * (NOBS - 1)/2 + 1 containing the frequency distribution of possible values of K. (Output)
K will range from minus to plus NOBS * (NOBS - 1)/2, inclusive, in increments of 2, with frequency FRQ(*i*), for a possible K = 2 * (*i* - 1) - NOBS * (NOBS - 1)/2, where *i* = 1, 2, ..., NOBS * (NOBS - 1)/2 + 1.

PROB — Probability of equaling or exceeding K if the samples on which K is based are uncorrelated. (Output)

FORTRAN 90 Interface

Generic: CALL KENDP (NOBS, K, FRQ, PROB)
Specific: The specific interface names are S_KENDP and D_KENDP.

FORTRAN 77 Interface

Single: CALL KENDP (NOBS, K, FRQ, PROB)
Double: The double precision name is DKENDP.

Description

Routine KENDP computes the frequency distribution of the Kendall *S* statistic and the probability that *S* equals or exceeds a given value *K*. Routine KENDP requires the sample size, *n* = NOBS, on input. The frequencies reported in position *i* of FRQ correspond to

$$S = 2(i - 1) - n(n - 1)/2$$

To obtain the probability distribution of *S*, divide each frequency by the sum of the frequencies in FRQ.

The upper bound on **NOBS** that can be handled by **KENDP** depends upon the largest real number that can be represented in the computer being used (**AMACH**(2)). If this value is 1.0E+46 or less, **NOBS** cannot be greater than 33.

Comments

Workspace may be explicitly provided, if desired, by use of **K2NDP**/**DK2NDP**.

The reference is:

```
CALL K2NDP (NOBS, K, FRQ, PROB, FWK)
```

The additional argument is:

FWK — Work vector of length (**NOBS** - 1) * (**NOBS** - 2)/2 + 1.

Example

The frequency distribution *S* for **NOBS** of 4 is computed. The probability is computed for *S* = 4.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    K, NOBS
      PARAMETER  (K=4, NOBS=4)
!
      INTEGER    I, M, NOUT
      REAL       FRQ(NOBS*(NOBS-1)/2+1,3), PROB, SUM
      CHARACTER  CLABEL(4)*10, RLABEL(1)*10
!
      DATA RLABEL/'NONE'/
      DATA CLABEL/'  ', 'S', 'FRQ', 'pf'/
!
      M = NOBS*(NOBS-1)/2 + 1
      DO 10  I=1, M
         FRQ(I,1) = 2*(I-1) - NOBS*(NOBS-1)/2
   10 CONTINUE
!
      CALL KENDP (NOBS, K, FRQ(1:,2), PROB)
!                                Compute the probabilities
      SUM = SSUM(M,FRQ(1:,2),1)
      CALL SCOPY (M, FRQ(1:,2), 1, FRQ(1:,3), 1)
      CALL SSCAL (M, 1.0/SUM, FRQ(1:,3), 1)
!                                Print results
      CALL UMACH (2, NOUT)
      CALL WRRRL (' ', FRQ, RLABEL, CLABEL, FMT='(W10.4)')
      WRITE (NOUT,*) 'PROB = ', PROB
      END
```

Output

```
            S        FRQ           pf
```

```
          -6.000       1.000       0.042
          -4.000       3.000       0.125
          -2.000       5.000       0.208
           0.000       6.000       0.250
           2.000       5.000       0.208
           4.000       3.000       0.125
           6.000       1.000       0.042
           PROB =    0.16666667
```

Analysis of Variance

Routines

Usage Notes

The routines described in this chapter are for commonly-used experimental designs. Typically, responses are stored in the input vector **Y** in a pattern that takes advantage of the balanced design structure. Consequently, the full set of model subscripts is not needed to identify each response. The routines assume the usual pattern, which requires that the last model subscript change most rapidly, the next to last model subscript change next most rapidly, and so forth, with the first subscript changing the slowest. This pattern is referred to as lexicographical ordering.

Routines **AONEW**, **AONEC**, and **ANEST** allow missing responses. NaN (not a number) is the missing value code used by these routines. Use routine **AMACH** to retrieve NaN. Any element of **Y** that is missing must be set to **AMACH**(6). For a description of **AMACH**, see the section Machine-Dependent Constants in the Reference Material Other routines described in this chapter do not allow missing responses because they generally deal with balanced designs.

As a diagnostic tool for determination of the validity of a model, routines in this chapter typically perform a test for lack of fit when *n* (*n* > 1) responses are available in each cell of the experimental design. Routines in Chapter 2, "Regression," are useful for analysis of generalizations of many of the models treated in this chapter. In particular, Chapter 2 provides routines for the general linear model.

AONEW

Analyzes a one-way classification model.

Required Arguments

NI — Vector of length **NGROUP** containing the number of responses for each group. (Input)

Y — Vector of length **NI**(1) + **NI**(2) + + **NI**(**NGROUP**) containing the responses for each group. (Input)

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	**AOV(I)**
1	Degrees of freedom for among groups
2	Degrees of freedom for within groups
3	Total (corrected) degrees of freedom
4	Sum of squares for among groups
5	Sum of squares for within groups
6	Total (corrected) sum of squares
7	Among-groups mean square
8	Within-groups mean square
9	*F* -statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the error within groups
14	Overall mean of Y
15	Coefficient of variation (in percent)

Optional Arguments

NGROUP — Number of groups. (Input)
Default: **NGROUP** = size (**NI**,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	AOV is printed only.
2	STAT is printed only.
3	All printing is performed.

STAT — NGROUP by 4 matrix containing information concerning the groups. (Output)
Row I contains information pertaining to the I-th group. The information in the columns is as follows:

	Description
1	Group number
2	Number of nonmissing observations
3	Group mean
4	Group standard deviation

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSTAT= size (STAT , 1)

NMISS — Number of missing values. (Output)
Elements of Y containing NaN (not a number) are omitted from the computations.

FORTRAN 90 Interface

Generic: CALL AONEW (NI, Y, AOV [, ...])
Specific: The specific interface names are S_AONEW and D_AONEW.

FORTRAN 77 Interface

Single: CALL AONEW (NGROUP, NI, Y, IPRINT, AOV, STAT, LDSTAT, NMISS)
Double: The double precision name is DAONEW.

Description

Routine AONEW performs an analysis of variance of responses from a one-way classification design. The model is

$$y_{ij} = \mu_i + \varepsilon_{ij} \quad i = 1, 2, \ldots, k; j = 1, 2, \ldots, n_i$$

where the observed value of y_{ij} constitutes the j-th response in the i-th group, μ_i denotes the population mean for the i-th group, and the ε_{ij}'s are errors that are identically and independently distributed normal with mean zero and variance σ^2. **AONEW** requires the y_{ij}'s as input into a single vector **Y** with responses in each group occupying contiguous locations. The analysis of variance table is computed along with the group sample means and standard deviations. A discussion of formulas and interpretations for the one-way analysis of variance problem appears in most elementary statistics texts, e.g., Snedecor and Cochran (1967, Chapter 10).

Example

This example computes a one-way analysis of variance for data discussed by Searle (1971, Table 5.1, pages 165-179). The responses are plant weights for 6 plants of 3 different types-3 normal, 2 off-types, and 1 aberrant. The responses are given by type of plant in the following table:

Normal	Off-Type	Aberrant
101	84	32
105	88	
94		

Note that for the group with only one response, the standard deviation is undefined and is set to NaN (not a number).

```
      USE AONEW_INT

      IMPLICIT   NONE
      INTEGER    NGROUP, NOBS
      PARAMETER  (NGROUP=3, NOBS=6)
!
      INTEGER    IPRINT, NI(NGROUP)
      REAL       AOV(15), Y(NOBS)
!
      DATA NI/3, 2, 1/
      DATA Y/101.0, 105.0, 94.0, 84.0, 88.0, 32.0/
!
      IPRINT = 3
      CALL AONEW (NI, Y, AOV, IPRINT=IPRINT)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
 Y             98.028     96.714            4.83          84           5.751
```

```
                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
Source                    DF     Squares      Square  Overall F  Larger F
Among Groups               2        3480      1740.0     74.571    0.0028
Within Groups              3          70        23.3
Corrected Total            5        3550

              Group Statistics
                                          Standard
 Group           N        Mean   Deviation
     1           3         100       5.568
     2           2          86       2.828
     3           1          32         NaN
```

AONEC

Analyzes a one-way classification model with covariates.

Required Arguments

NI — Vector of length NGROUP containing the number of responses for each group. (Input)

XY — (NI(1) + NI(2) + … + NI(NGROUP)) by (NCOV + 1) matrix containing the data for each covariate and the response variable. (Input)
Data for each group must appear in contiguous rows of XY, and the responses must appear in the last column.

AOV — Vector of length 15 that contains statistics relating to the analysis of variance for the model assuming parallelism. (Output)

I	AOV(I)
1	Degrees of freedom for model (groups + covariates)
2	Degrees of freedom for error
3	Total (corrected) degrees of freedom
4	Sum of squares for model
5	Sum of squares for error
6	Total (corrected) sum of squares
7	Model mean square
8	Error mean square
9	*F* -statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimate of the error standard deviation
14	Overall response mean
15	Coefficient of variation (in percent)

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (NI,1).

NCOV — Number of covariates. (Input)
Default: NCOV = size (XY,2) – 1.

LDXY — Leading dimension of XY exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXY = size (XY,1).

ITEST — Indicator for test for parallelism (equal covariate coefficients across groups). (Input)
Default: ITEST = 0.

ITEST	Action
0	Test for parallelism is not performed.
1	Test for parallelism is performed.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing for model assuming parallelism is performed.
2	Printing for separate regression models for each group is performed as well as for the model assuming parallelism.

COEF — NGROUP + NCOV by 4 matrix containing statistics relating to the regression coefficients for the model assuming parallelism. (Output)
Each row corresponds to a coefficient in the model. For I = 1, 2, …, NGROUP, row I is for the Y intercept for the I-th group. The remaining NCOV rows are for the covariate coefficients. The statistics in the columns are as follows:

Col.	Description
1	Coefficient estimate
2	Estimated standard error of the estimate
3	*t*-statistic
4	*p*-value

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

R — NGROUP + NCOV by NGROUP + NCOV upper triangular matrix containing the *R* matrix from the *QR* decomposition. (Output)
The *R* matrix is from the regression assuming parallelism.

LDR — Leading dimension of *R* exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

PTSS — Vector of length 8 containing statistics relating to the partial sums of squares for groups and for covariates in the model assuming parallelism. (Output)

I	PTSS(I)
1	Degrees of freedom for groups after covariates
2	Degrees of freedom for covariates after groups
3	Sum of squares for groups after covariates
4	Sum of squares for covariates after groups
5	*F* -statistic for groups
6	*F* -statistic for covariates
7	*p*-value for groups
8	*p*-value for covariates

TESTPL — Vector of length 10 containing statistics relating to the test for parallelism. (Output if ITEST = 1)
If ITEST = 0, TESTPL is not referenced and can be a vector of length one.

I	TESTPL(I)
1	Extra degrees of freedom for model not assuming parallelism
2	Degrees of freedom for error for model not assuming parallelism
3	Degrees of freedom for error for model assuming parallelism
4	Extra sum of squares for model not assuming parallelism
5	Sum of squares for error for model not assuming parallelism
6	Sum of squares for error for model assuming parallelism
7	Mean square for TESTPL(1)
8	Mean square for TESTPL(2)

I	TESTPL(I)
9	*F* -statistic
10	*p*-value

XYMEAN — NGROUP + 1 by NCOV + 3 matrix containing means. (Output)
Each row for I = 1, 2, ..., NGROUP corresponds to a group. Row NGROUP + 1 contains overall statistics. The statistics in the columns are as follows:

Column	Description
1	Number of nonmissing cases.
2 thru NCOV+1	Covariate means.
NCOV + 2	Response mean.
NCOV + 3	Adjusted mean assuming parallelism

LDXYME — Leading dimension of XYMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXYME = size (XYMEAN,1).

COVM — NGROUP by NGROUP matrix containing the estimated variance-covariance matrix of the adjusted group means in the model assuming parallelism. (Output)

LDCOVM — Leading dimension of COVM exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVM = size (COVM,1).

COVB — NGROUP + NCOV by NGROUP + NCOV matrix containing the estimated variance-covariance matrix of the estimated coefficients in the model assuming parallelism. (Output)
If R is not needed, R and COVB can occupy the same storage locations.

LDCOVB — Leading dimension of COVB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOVB = size (COVB,1).

NRMISS — Number of rows of XY that contain any missing values. (Output)
Rows of XY containing NaN (not a number) are omitted from computations.

FORTRAN 90 Interface

Generic: CALL AONEC (NI, XY, AOV [, ...])
Specific: The specific interface names are S_AONEC and D_AONEC.

FORTRAN 77 Interface

Single: CALL AONEC (NGROUP, NI, NCOV, XY, LDXY, ITEST, IPRINT, COEF, LDCOEF, R, LDR, AOV, PTSS, TESTPL, XYMEAN, LDXYME, COVM, LDCOVM, COVB, LDCOVB, NRMISS)

Double: The double precision name is DAONEC.

Description

Routine AONEC performs analyses for models that combine the features of a one-way analysis of variance model with that of a multiple linear regression model. The basic one-way analysis of covariance model is

$$y_{ij} = \beta_{0i} + \beta_1 x_{ij1} + \beta_2 x_{ij2} + \ \ldots \ + \beta_m x_{ijm} + \varepsilon_{ij} \quad i = 1, 2, \ \ldots, k; \quad j = 1, 2, \ \ldots, n_i$$

where the observed value of y_{ij} constitutes the j-th response in the i-th group, $\beta_{0\,i}$ denotes the y intercept for the regression function for the i-th group, $\beta_1, \beta_2, \ldots, \beta_m$ are the regression coefficients for the covariates, and the ε_{ij}'s are independently distributed normal errors with mean zero and variance σ^2. This model allows the regression function for each group to have different intercepts. However, the remaining m regression coefficients are the same for each group, i.e., the regression functions are parallel. Often in practice, the regression functions are not parallel. In addition to estimates for the model assuming parallelism, AONEC computes estimates and summary statistics for the separate regressions for each group. With IPRINT = 2, the estimates and summary statistics for each group are printed. If ITEST = 1, a test for parallelism is performed.

AONEC requires ($x_{ij1}, x_{ij2}, \ldots, x_{ijk}, y_{ij}$) as input into a single data matrix XY with the data for each group occupying contiguous rows of XY.

Estimates for the β_{0i}'s and $\beta_1, \beta_2, \ldots, \beta_m$ in the model assuming parallelism are computed and stored in COEF. Summary statistics are also computed for this model. The adjusted group means (stored in column m + 3 of XYMEAN) are given by

$$\hat{\beta}_{oi} + \hat{\beta}_1 \overline{x}_1 + \hat{\beta}_2 \overline{x}_2 + \ \ldots \ + \hat{\beta}_m \overline{x}_m$$

The estimated covariance between the $i1$-th and $i2$-th adjusted group mean is given by

$$v_{i_1 i_2} + \sum_{r=1}^{m}\sum_{s=1}^{m} \overline{x}_r v_{k+r,k+s} \overline{x}_s + \sum_{r=1}^{m} \overline{x}_r v_{i_1,k+r} + \sum_{r=1}^{m} \overline{x}_r v_{i_2,k+r}$$

where v_{pq} is the pq-th entry in COVB and is the estimated covariance between the p-th and q-th estimated coefficients in the regression function.

The design of **AONEC** can be used with routines described in Chapter 2, "Regression." For example, confidence intervals and diagnostics for the individual cases can be computed by using the output matrices **R** and **COEF** as input into regression routines for case analysis.

A discussion of formulas and interpretations for the one-way analysis of covariance problem appears in most elementary statistics texts, e.g., Snedecor and Cochran (1967, Chapter 14).

Comments

Workspace may be explicitly provided, if desired, by use of **A2NEC**/**DA2NEC**. The reference is:

```
CALL A2NEC (NGROUP, NI, NCOV, XY, LDXY, ITEST, IPRINT, COEF, LDCOEF, R, LDR, AOV,
   PTSS, TESTPL, XYMEAN, LDXYME, COVM, LDCOVM, COVB, LDCOVB, NRMISS, WK)
```

The additional argument is:

WK — Work vector of length 4 * (**NGROUP** + **NCOV** + 1).

Examples

Example 1

This example fits a one-way analysis of covariance model assuming parallelism using data discussed by Snedecor and Cochran (Table 14.6.1, pages 432-436). The responses are concentrations of cholesterol (in mg/100 ml) in the blood of two groups of women: women from Iowa and women from Nebraska. Age of a woman is the single covariate. The cholesterol concentrations and ages of the women according to state are shown in the following table. (There are 11 Iowa women and 19 Nebraska women in the study. Only the first 5 women from each state are shown here.)

Age	**Cholesterol**	**Age**	**Cholesterol**
46	181	18	137
52	228	44	173
39	182	33	177
65	249	78	241
54	259	51	225

There is no evidence from the data to indicate that the regression lines for cholesterol concentration as a function of age are not parallel for Iowa and Nebraska women (*p*-value is 0.5425). The parallel line model suggests that Nebraska women may have higher cholesterol concentrations than Iowa women. The cholesterol concentrations (adjusted for age) are 195.5 for Iowa women versus 224.2 for Nebraska women. The difference is 28.7 with an estimated standard error of

$$\sqrt{170.4 + 97.4 - 2\left(2.9\right)} = 16.1$$

```
      USE AONEC_INT

      IMPLICIT   NONE
      INTEGER    LDXY, NCOV,  NGROUP, NOBS
      PARAMETER  (NCOV=1, NGROUP=2, NOBS=30, LDXY=NOBS)
!
      INTEGER    IPRINT, ITEST, NI(NGROUP)
      REAL       AOV(15), XY(LDXY,NCOV+1)
!
      DATA NI/11, 19/
      DATA XY/46.0, 52.0, 39.0, 65.0, 54.0, 33.0, 49.0, 76.0, 71.0, &
          41.0, 58.0, 18.0, 44.0, 33.0, 78.0, 51.0, 43.0, 44.0, 58.0, &
          63.0, 19.0, 42.0, 30.0, 47.0, 58.0, 70.0, 67.0, 31.0, 21.0, &
          56.0, 181.0, 228.0, 182.0, 249.0, 259.0, 201.0, 121.0, &
          339.0, 224.0, 112.0, 189.0, 137.0, 173.0, 177.0, 241.0, &
          225.0, 223.0, 190.0, 257.0, 337.0, 189.0, 214.0, 140.0, &
          196.0, 262.0, 261.0, 356.0, 159.0, 191.0, 197.0/
!
      ITEST  = 1
      IPRINT = 2
      CALL AONEC (NI, XY, AOV, ITEST=ITEST, IPRINT=IPRINT)
!
      END
```

Output

```
 SEPARATE REGRESSION FOR GROUP  1

 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
 Y             47.120     41.245            48.9      207.7           23.54
```

```
                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
Source               DF     Squares       Square  Overall F  Larger F
Model                 1     19177.2     19177.2      8.020    0.0197
Error                 9     21521.0      2391.2
Corrected Total      10     40698.2

                Inference on Coefficients
                    Standard                 Prob. of
Coef.    Estimate       Error  t-statistic  Larger |t|
    1       35.81       62.47        0.573      0.5805
    2        3.24        1.14        2.832      0.0197

SEPARATE REGRESSION FOR GROUP  2
Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             56.812     54.272           39.76       217.1           18.31



                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
Source               DF     Squares       Square  Overall F  Larger F
Model                 1     35351.9     35351.9     22.363    0.0002
Error                17     26873.9      1580.8
Corrected Total      18     62225.8

                Inference on Coefficients
                    Standard                 Prob. of
Coef.    Estimate       Error  t-statistic  Larger |t|
    1       101.3       26.13        3.876      0.0012
    2         2.5        0.53        4.729      0.0002

SAME REGRESSION FOR ALL GROUPS

Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             47.303     45.421           44.14       213.7           20.66

                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
Source               DF     Squares       Square  Overall F  Larger F
Model                 1     48976.3     48976.3     25.134    0.0000
Error                28     54560.4      1948.6
Corrected Total      29    103536.7

                Inference on Coefficients
                    Standard                 Prob. of
Coef.    Estimate       Error  t-statistic  Larger |t|
    1       91.57       25.65        3.570      0.0013
    2        2.51        0.50        5.013      0.0000

REGRESSION ASSUMING PARALLELISM

Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             52.573     49.060           42.65       213.7           19.96

                    * * * Analysis of Variance * * *
                                  Sum of        Mean             Prob. of
```

Source	DF	Squares	Square	Overall F	Larger F
Model	2	54432.8	27216.4	14.965	0.0000
Error	27	49103.9	1818.7		
Corrected Total	29	103536.7			

Partial Sums of Squares

Source	DF	Sum of Squares	F	Prob. of Larger F
Groups after Covariates	1	5456.5	3.000	0.0947
Covariates after Groups	1	53820.1	29.593	0.0000

R Matrix

	1	2	3
1	3.3	0.0	176.1
2		4.4	200.3
3			86.0

Inference on Coefficients

Coef.	Estimate	Standard Error	t-statistic	Prob. of Larger |t|
1	64.49	29.3	2.201	0.0365
2	93.14	24.8	3.756	0.0008
3	2.70	0.5	5.440	0.0000

Test for Parallelism

Source	DF	Sum of Squares	Mean Square	F	Prob. of Larger F
Extra due to nonparallelism	1	709.0	709.0	0.381	0.5425
Error assuming nonparallelism	26	48394.9	1861.3		
Error assuming parallelism	27	49103.9			

XYMEAN

	1	2	3	4
1	11	53.09	207.7	195.5
2	19	45.95	217.1	224.2
3	30	48.57	213.7	213.7

Variance-Covariance Matrix of the Adjusted Group Means

	1	2
1	170.4	-2.9
2		97.4

Variance-Covariance Matrix of the Estimated Coefficients

	1	2	3
1	858.6	600.0	-13.1
2		615.0	-11.3
3			0.2

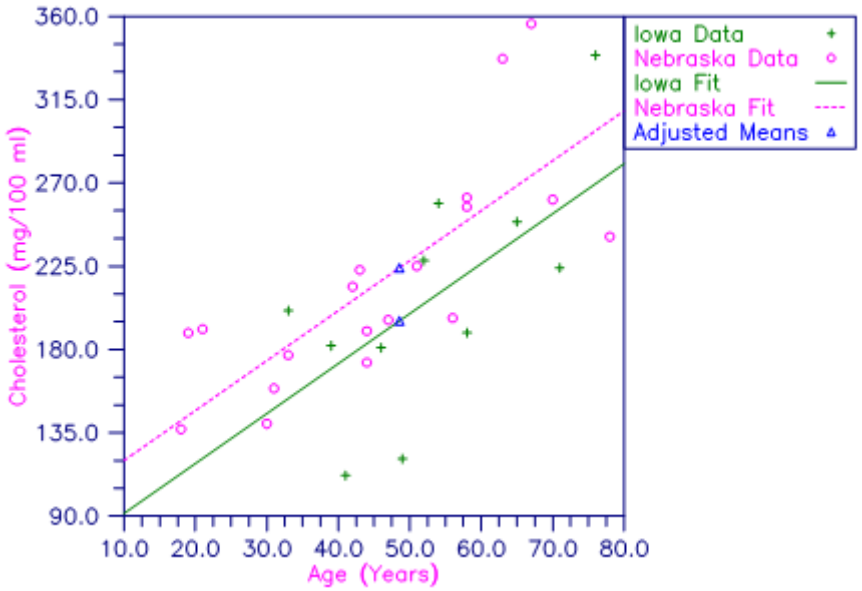


Figure 14, Plot of Cholesterol Concentrations and Fitted Parallel Lines by State

Example 2

This example fits a one-way analysis of covariance model and performs a test for parallelism using data discussed by Snedecor and Cochran (1967, Table 14.8.1, pages 438-443). The responses are weight gains (in pounds per day) of 40 pigs for 4 groups of pigs under varying treatments. Two covariates-initial age (in days) and initial weight (in pounds)-are used. For each treatment, there are 10 pigs. Only the first 5 pigs from each treatment are shown here.

Age	**Wt.**	**Gain**	**Age**	**Wt.**	**Gain**	**Age**	**Wt.**	**Gain**	**Age**	**Wt.**	**Gain**
78	61	1.40	78	74	1.61	78	80	1.67	77	62	1.40
90	59	1.79	99	75	1.31	83	61	1.41	71	55	1.47
94	76	1.72	80	64	1.12	79	62	1.73	78	62	1.37
71	50	1.47	75	48	1.35	70	47	1.23	70	43	1.15
99	61	1.26	94	62	1.29	85	e59	1.49	95	57	1.22

```
USE AONEC_INT

IMPLICIT   NONE
INTEGER    LDXY, NCOV, NGROUP, NOBS
```

```
      PARAMETER  (NCOV=2, NGROUP=4, NOBS=40, LDXY=NOBS)
!
      INTEGER    IPRINT, ITEST, NI(NGROUP)
      REAL       AOV(15), XY(LDXY,NCOV+1)
!
      DATA NI/10, 10, 10, 10/
      DATA XY/78.0, 90.0, 94.0, 71.0, 99.0, 80.0, 83.0, 75.0, 62.0, &
          67.0, 78.0, 99.0, 80.0, 75.0, 94.0, 91.0, 75.0, 63.0, 62.0, &
          67.0, 78.0, 83.0, 79.0, 70.0, 85.0, 83.0, 71.0, 66.0, 67.0, &
          67.0, 77.0, 71.0, 78.0, 70.0, 95.0, 96.0, 71.0, 63.0, 62.0, &
          67.0, 61.0, 59.0, 76.0, 50.0, 61.0, 54.0, 57.0, 45.0, 41.0, &
          40.0, 74.0, 75.0, 64.0, 48.0, 62.0, 42.0, 52.0, 43.0, 50.0, &
          40.0, 80.0, 61.0, 62.0, 47.0, 59.0, 42.0, 47.0, 42.0, 40.0, &
          40.0, 62.0, 55.0, 62.0, 43.0, 57.0, 51.0, 41.0, 40.0, 45.0, &
          39.0, 1.40, 1.79, 1.72, 1.47, 1.26, 1.28, 1.34, 1.55, 1.57, &
          1.26, 1.61, 1.31, 1.12, 1.35, 1.29, 1.24, 1.29, 1.43, 1.29, &
          1.26, 1.67, 1.41, 1.73, 1.23, 1.49, 1.22, 1.39, 1.39, 1.56, &
          1.36, 1.40, 1.47, 1.37, 1.15, 1.22, 1.48, 1.31, 1.27, 1.22, &
          1.36/
!
      ITEST  = 1
      IPRINT = 2
      CALL AONEC (NI, XY, AOV, ITEST=ITEST, IPRINT=IPRINT)
!
      END
```

Output

```
 SEPARATE REGRESSION FOR GROUP  1

 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
 Y            13.271      0.000          0.2013      1.464           13.75

                   * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
  Source               DF     Squares      Square  Overall F  Larger F
  Model                 2      0.0434     0.02170      0.536    0.6075
  Error                 7      0.2836     0.04052
  Corrected Total       9      0.3270

                Inference on Coefficients
                     Standard                 Prob. of
  Coef.    Estimate       Error  t-statistic  Larger |t|
      1       1.357      0.4639        2.925      0.0222
      2      -0.006      0.0105       -0.572      0.5849
      3       0.011      0.0114        0.948      0.3749

 SEPARATE REGRESSION FOR GROUP  2

 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
 Y            21.989      0.000          0.1292      1.319           9.799

                   * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
  Source               DF     Squares      Square  Overall F  Larger F
  Model                 2      0.0330     0.01648      0.987    0.4193
  Error                 7      0.1169     0.01670
```

```
 Corrected Total        9      0.1499

                Inference on Coefficients
                      Standard                 Prob. of
 Coef.    Estimate       Error  t-statistic  Larger |t|
     1       1.401      0.2694        5.199      0.0013
     2      -0.005      0.0040       -1.164      0.2825
     3       0.005      0.0040        1.301      0.2343

SEPARATE REGRESSION FOR GROUP  3

Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             49.246     34.745          0.1369       1.445           9.473

                   * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Model                 2      0.1273     0.06364      3.396    0.0931
 Error                 7      0.1312     0.01874
 Corrected Total       9      0.2584

                Inference on Coefficients
                      Standard                 Prob. of
 Coef.    Estimate       Error  t-statistic  Larger |t|
     1       1.452      0.4709        3.082      0.0178
     2      -0.008      0.0075       -1.017      0.3429
     3       0.011      0.0043        2.544      0.0384

SEPARATE REGRESSION FOR GROUP  4

Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             17.076      0.000          0.1141       1.325           8.609
                   * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Model                 2      0.0188     0.00938      0.721    0.5193
 Error                 7      0.0911     0.01301
 Corrected Total       9      0.1098

                Inference on Coefficients
                      Standard                 Prob. of
 Coef.    Estimate       Error  t-statistic  Larger |t|
     1       1.044      0.2574        4.055      0.0048
     2       0.001      0.0038        0.251      0.8094
     3       0.004      0.0051        0.833      0.4324

SAME REGRESSION FOR ALL GROUPS

Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             17.724     13.277          0.1508       1.388           10.86

                   * * * Analysis of Variance * * *
                              Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Model                 2       0.181     0.09064      3.985    0.0271
 Error                37       0.842     0.02274
 Corrected Total      39       1.023
```

Inference on Coefficients

Coef.	Estimate	Standard Error	t-statistic	Prob. of Larger |t|
1	1.251	0.1708	7.327	0.0000
2	-0.003	0.0028	-1.178	0.2464
3	0.007	0.0027	2.743	0.0093

REGRESSION ASSUMING PARALLELISM

Dependent Variable	R-squared (percent)	Adjusted R-squared	Est. Std. Dev. of Model Error	Mean	Coefficient of Var. (percent)
Y	34.467	24.829	0.1404	1.388	10.11

* * * Analysis of Variance * * *

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
Model	5	0.353	0.07050	3.576	0.0105
Error	34	0.670	0.01971		
Corrected Total	39	1.023			

Partial Sums of Squares

Source	DF	Sum of Squares	F	Prob. of Larger F
Groups after Covariates	3	0.1712	2.895	0.0493
Covariates after Groups	2	0.1750	4.438	0.0194

R Matrix

	1	2	3	4	5	6
1	3.2	0.0	0.0	0.0	252.7	172.0
2		3.2	0.0	0.0	247.9	173.9
3			3.2	0.0	236.9	164.4
4				3.2	237.2	156.5
5					67.4	42.7
6						55.3

Inference on Coefficients

Coef.	Estimate	Standard Error	t-statistic	Prob. of Larger |t|
1	1.337	0.1724	7.751	0.0000
2	1.182	0.1697	6.965	0.0000
3	1.318	0.1626	8.109	0.0000
4	1.217	0.1624	7.493	0.0000
5	-0.003	0.0026	-1.314	0.1978
6	0.007	0.0025	2.919	0.0062

Test for Parallelism

Source	DF	Sum of Squares	Mean Square	F	Prob. of Larger F
Extra due to nonparallelism	6	0.0474	0.00790	0.355	0.9007
Error assuming nonparallelism	28	0.6228	0.02224		
Error assuming parallelism	34	0.6703			

XYMEAN

	1	2	3	4	5
1	10	79.90	54.40	1.464	1.461

2	10	78.40	55.00	1.319	1.307
3	10	74.90	52.00	1.445	1.443
4	10	75.00	49.50	1.325	1.342
5	40	77.05	52.72	1.388	1.388

Variance-Covariance Matrix of the Adjusted Group Means

	1	2	3	4
1	0.002007	0.000016	-0.000027	-0.000024
2		0.001992	-0.000007	-0.000030
3			0.001994	0.000011
4				0.002014

Variance-Covariance Matrix of the Estimated Coefficients

	1	2	3	4	5	6
1	0.02974	0.02729	0.02605	0.02602	-0.00033	-0.00002
2		0.02880	0.02561	0.02556	-0.00032	-0.00003
3			0.02642	0.02441	-0.00031	-0.00003
4				0.02638	-0.00032	-0.00001
5					0.00001	0.00000
6						0.00001

ATWOB

Analyzes a randomized block design or a two-way balanced design.

Required Arguments

NBLK — Number of blocks. (Input)

NTRT — Number of treatments. (Input)

NRESP — Number of repeated responses within each block-treatment combination. (Input)

Y — Vector of length NBLK * NTRT * NRESP containing the responses. (Input)
The first NRESP elements of Y contain the responses for block one, treatment one, the second NRESP elements of Y contain the responses for block one, treatment two; ...; the last NRESP elements of Y contain the responses for block NBLK, treatment NTRT.

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for the model (blocks and treatments)
2	Degrees of freedom for error (interaction is pooled with the within-cell error)
3	Total (corrected) degrees of freedom
4	Sum of squares for the model (blocks and treatments)
5	Sum of squares for error (interaction is pooled with the within-cell error)
6	Total (corrected) sum of squares
7	Model mean square
8	Error mean square
9	F-statistic
10	p-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Overall mean of Y
15	Coefficient of variation (in percent)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	AOV(I)
0	No printing is performed.
1	Print AOV, EFSS, and TESTLF (if NRESP > 1).
2	Print YMEANS only.
3	All printing is performed.

EFSS — Vector of length 8 containing statistics relating to the sums of squares for the effects in the model. (Output)
Elements of EFSS are described as follows:

	Description
1, 2	Degrees of freedom for blocks and treatments, respectively
3, 4	Sum of squares for blocks and treatments, respectively
5, 6	*F*-statistics for blocks and treatments, respectively. *F*-statistics are computed using AOV(8) as the estimated error variance.
7, 8	*p*-values associated with the *F* -statistics

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the two-way model without interaction. (Output if NRESP > 1)
If NRESP = 1, TESTLF is not referenced and can be a vector of length one. Elements of TESTLF are described as follows:

	Description
1	Degrees of freedom for interaction
2	Degrees of freedom for within-cell error
3	Degrees of freedom for error (TESTLF(1) + TESTLF(2))
4	Sum of squares for interaction
5	Sum of squares for interaction
6	Sum of squares for within-cell error
7	Mean square for interaction
8	Mean square for within-cell error
9	*F*-statistic
10	*p*-value

YMEANS — Vector of length NBLK + NTRT + NBLK * NTRT containing the block means, treatment means and block-by-treatment means, respectively. (Output)

FORTRAN 90 Interface

Generic: CALL ATWOB (NBLK, NTRT, NRESP, Y, AOV [, ...])
Specific: The specific interface names are S_ATWOB and D_ATWOB.

FORTRAN 77 Interface

Single: CALL ATWOB (NBLK, NTRT, NRESP, Y, IPRINT, AOV, EFSS, TESTLF, YMEANS)
Double: The double precision name is DATWOB.

Description

Routine **ATWOB** performs an analysis for a two-way classification design with balanced data. For balanced data, there must be an equal number of responses in each cell of the two-way layout. The basic model is the same as for the randomized block design. The block and treatment effects are additive, i.e., there are no interactions. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} \qquad i = 1, 2, \ldots, n_1; j = 1, 2, \ldots, n_2; k = 1, 2, \ldots, n_3$$

where the observed value of y_{ijk} constitutes the k-th response in the ij-th cell of the two-way layout, $\mu + \alpha_i + \beta_j$ is the population mean for the ij-th cell, and the ε_{ijk}'s are identically and independently distributed normal errors with mean zero and variance σ^2. This model assumes that the effects for the two factors are additive. Often in practice, there are interactions between the two factors. For this reason, in addition to summary statistics for the additive model, **ATWOB** computes a test for nonadditivity (lack of fit). The test used here requires at least two responses in each cell. Tests for nonadditivity with one response per cell are given by Tukey (1949) and Mandel (1961). Tukey's test is discussed by Snedecor and Cochran (1967, pages 331-334).

The routine **ATWOB** requires y_{ijk}'s as input into a single vector **Y** with the data for each cell occupying contiguous elements. The cells must be in standard order, i.e., (1, 1), (1, 2), …, (1, n_2), (2, 1), (2, 2), …, (2, n_2), …, (n_1, 1), (n_1, 2), …, (n_1, n_2):

Examples

Example 1

This example performs an analysis for a randomized block design using data discussed by Neter and Wasserman (1974,Table 23.2, pages 725-730). Fifteen businessmen were shown one of three methods for quantifying the maximum risk premium they would be willing to pay to avoid uncertainty. The responses are a stated degree of confidence, on a scale of 0 (no confidence) to 20 (highest confidence). The fifteen businessmen were grouped into five blocks by age. The three businessmen in each block were randomly assigned to a rating method. The data are given in the following table:

	Confidence Rating		
Block	Method 1	Method 2	Method 3
1	1	5	8
2	2	8	14
3	7	9	16
4	6	13	18
5	12	14	17

```
      USE ATWOB_INT

      IMPLICIT   NONE
      INTEGER    NBLK, NRESP, NTRT
      PARAMETER  (NBLK=5, NRESP=1, NTRT=3)
!
      INTEGER    IPRINT
      REAL       AOV(15), Y(NBLK*NTRT*NRESP)
!
      DATA Y/1.0, 5.0, 8.0, 2.0, 8.0, 14.0, 7.0, 9.0, 16.0, 6.0, 13.0, &
          18.0, 12.0, 14.0, 17.0/
!
      IPRINT = 3
      CALL ATWOB (NBLK, NTRT, NRESP, Y, AOV, IPRINT=IPRINT)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
 Y            94.003     89.506          1.727         10           17.27

                   * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
  Source              DF      Squares      Square  Overall F  Larger F
  Model                6        374.1       62.36     20.901    0.0002
  Error                8         23.9        2.98
  Corrected Total     14        398.0

  * * * Decomposition of Variation Attributable to the Model * * *
                                Sum of             Prob. of
        Source          DF      Squares          F  Larger F
        Blocks           4        171.3     14.358    0.0010
        Treatment        2        202.8     33.989    0.0001

  * * * Block Means * * *
     Block  Mean (N=3)
         1      4.6667
         2      8.0000
         3     10.6667
         4     12.3333
         5     14.3333
```

```
   * * * Treatment Means * * *
      Treatment  Mean (N=5)
              1      5.6000
              2      9.8000
              3     14.6000

      * * * Cell Means * * *
   Block  Treatment  Mean (N=1)
       1          1      1.0000
       1          2      5.0000
       1          3      8.0000
       2          1      2.0000
       2          2      8.0000
       2          3     14.0000
       3          1      7.0000
       3          2      9.0000
       3          3     16.0000
       4          1      6.0000
       4          2     13.0000
       4          3     18.0000
       5          1     12.0000
       5          2     14.0000
       5          3     17.0000
```

Example 2

This example fits an additive two-way analysis of variance model and performs a test for nonadditivity (lack of fit) using data discussed by Kirk (1982,Table 8.3-1, pages 354-359). The data for the two-way layout is given in the following table:

TREATMENT	**1**	**2**	**3**
1	24, 33, 37, 29, 42	44, 36, 25, 27, 43	38, 29, 28, 47, 48
2	30, 21, 39, 26, 34	35, 40, 27, 31, 22	26, 27, 36, 46, 45
3	21, 18, 10, 31, 20	41, 39, 50, 36, 34	42, 52, 53, 49, 64

```
      USE ATWOB_INT

      IMPLICIT   NONE
      INTEGER    NBLK, NRESP, NTRT
      PARAMETER  (NBLK=3, NRESP=5, NTRT=3)
!
      INTEGER    IPRINT
      REAL       AOV(15), Y(NBLK*NTRT*NRESP)
```

```
!
      DATA Y/24.0, 33.0, 37.0, 29.0, 42.0, 30.0, 21.0, 39.0, 26.0, &
          34.0, 21.0, 18.0, 10.0, 31.0, 20.0, 44.0, 36.0, 25.0, 27.0, &
          43.0, 35.0, 40.0, 27.0, 31.0, 22.0, 41.0, 39.0, 50.0, 36.0, &
          34.0, 38.0, 29.0, 28.0, 47.0, 48.0, 26.0, 27.0, 36.0, 46.0, &
          45.0, 42.0, 52.0, 53.0, 49.0, 64.0/
!
      IPRINT = 3
      CALL ATWOB (NBLK, NTRT, NRESP, Y, AOV, IPRINT=IPRINT)
      END
```

Output

```
Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
Y             33.206     26.526           9.336          35           26.68

                    * * * Analysis of Variance * * *
                                 Sum of        Mean             Prob. of
 Source                  DF     Squares      Square  Overall F  Larger F
 Model                    4      1733.3       433.3      4.971    0.0024
 Error                   40      3486.7        87.2
 Corrected Total         44      5220.0

 * * * Decomposition of Variation Attributable to the Model * * *
                                  Sum of             Prob. of
        Source           DF     Squares          F  Larger F
        Blocks            2      1543.3      8.853    0.0007
        Treatment         2       190.0      1.090    0.3460

                   * * * Test for Lack of Fit * * *
                              Sum of        Mean             Prob. of
 Source             DF     Squares      Square          F   Larger F
 Interaction         4      1236.7       309.2      4.947     0.0028
 Within cell        36      2250.0        62.5
 Error              40      3486.7

 * * * Block Means * * *
    Block  Mean (N=3)
        1     27.6667
        2     35.3333
        3     42.0000

 * * * Treatment Means * * *
    Treatment  Mean (N=3)
            1     35.3333
            2     32.3333
            3     37.3333

    * * * Cell Means * * *
 Block  Treatment  Mean (N=5)
     1          1     33.0000
     1          2     30.0000
     1          3     20.0000
     2          1     35.0000
     2          2     31.0000
     2          3     40.0000
     3          1     38.0000
     3          2     36.0000
```

```
          3           3      52.0000
```

ABIBD

Analyzes a balanced incomplete block design or a balanced lattice design.

Required Arguments

NTRT — Number of treatments. (Input)

NREP — Number of replications. (Input)

NBLK — Number of blocks. (Input)

NTBLK — Number of treatments within each block. (Input)

NRESP — Number of responses within each treatment-block combination. (Input)

Y — Vector of length NBLK * NTBLK * NRESP containing the responses. (Input)
The first NRESP elements of Y contain the responses for the first treatment in the first block in the first replicate. The second NRESP elements of Y contain the responses for the second treatment in the first block in the first replicate. The NTBLK-th NRESP elements of Y contain the responses for the NTBLK-th treatment in the first block in the first replicate. The last NRESP elements of Y contain the responses for the NTBLK-th treatment in the NBLK-th block in the NREP-th replicate.

ITRT — Vector of length NBLK * NTBLK containing the treatment numbers for the responses in Y. (Input)
The treatment numbers must be from the set 1, 2, ..., NTRT. For
I = 1, 2, ..., NBLK * NTBLK, element numbers (I - 1) * NRESP + 1 thru
(I - 1) * NRESP + NRESP of Y correspond to treatment number ITRT(I).

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square

9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

Optional Arguments

INTER — Interblock analysis option. (Input)
Default: `INTER` = 0.

`INTER`	**Means**
0	Intrablock analysis is requested. (Blocks are fixed effects.)
1	Interblock analysis is requested. (Blocks are random effects.)

IPRINT — Printing option. (Input)
Default: `IPRINT` = 0.

`INTER`	**Means**
0	No printing is performed.
1	Print `AOV`, `SQSS`, and `TESTLF` (if `NRESP` > 1).
2	Print `YMEANS` only.
3	All printing is performed.

SQSS — Vector of length 12 containing statistics relating to the sequential sum of squares for the model. (Output)

	Description
1, 2, 3	Degrees of freedom for replicates, blocks within replicates, and treatments (adjusted), respectively
4, 5, 6	Sum of squares for replicates, blocks within replicates, and treatments (adjusted), respectively
7, 8, 9	*F* -statistics for replicates, blocks, and treatments, respectively, computed using A0V(8) as the estimated error variance
10-12	*p*-values associated with the *F* -statistics

SSALT — Vector of length 2 containing an alternative partitioning of the model sum of squares. (Output)
SSALT(1) is the treatment sum of squares (unadjusted) and SSALT(2) is the block sum of squares (adjusted).

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the model. (Output, if NRESP > 1)
If NRESP = 1, TESTLF is not referenced and can be a vector of length one. Elements of TESTLF are described as follows:

	Description
1	Degrees of freedom for experimental error
2	Degrees of freedom for within-cell error
3	Degrees of freedom for error (TESTLF(1) + TESTLF(2))
4	Sum of squares for experimental error
5	Sum of squares for within-cell error
6	Sum of squares for error
7	Mean square for experimental error
8	Mean square for within-cell error
9	*F*-statistic
10	*p*-value

YMEANS — Vector of length NREP + NBLK + NTRT + NTBLK * NBLK containing the replicate means, block by replicate means, treatment means (adjusted), and treatment by block means, respectively. (Output)
The treatment means (adjusted) in YMEANS are used for estimating treatment differences.

SETRTD — Estimated standard error of a treatment difference. (Output)

EFNCY — Estimated efficiency of this design relative to a randomized complete block design. (Output)
The randomized complete block design has NBLK * NTBLK/NTRT complete blocks.

FORTRAN 90 Interface

Generic: CALL ABIBD (NTRT, NREP, NBLK, NTBLK, NRESP, Y, ITRT, AOV [, ...])

Specific: The specific interface names are S_ABIBD and D_ABIBD.

FORTRAN 77 Interface

Single: CALL ABIBD (NTRT, NREP, NBLK, NTBLK, NRESP, Y, ITRT, INTER, IPRINT, AOV, SQSS, SSALT, TESTLF, YMEANS, SETRTD, EFNCY)

Double: The double precision name is DABIBD.

Description

Routine ABIBD performs analyses for balanced incomplete block designs. The basic model used is the randomized block design with the source of variation for "blocks" subdivided into replications and blocks within replications. For INTER = 0, the model is

$$y_{ijtm} = \mu + \alpha_i + \beta_{jj} + \delta_t + \varepsilon_{ijkm} \qquad i = 1, \ldots, r; j = 1, \ldots, k; t = 1, \ldots, p; m = 1, \ldots, n$$

where the observed value of y_{ijtm} constitutes the m-th response with treatment t in block j within the i replicate, $\mu + \alpha_i + \beta_{ij} + \delta_t$ is the population mean for the response, and the ε_{ijtm}'s are independently distributed normal errors with mean zero and variance σ^2. This model assumes the block effects and treatment effects are additive. Often in practice, there are interactions between the blocks and treatments. For this reason, ABIBD computes a test for nonadditivity (lack of fit), in addition to summary statistics for the additive model. This test requires at least two responses in each cell.

The analysis performed with the β_{ij}'s regarded as fixed effects in the model (INTER = 0) is called an "intrablock analysis." For INTER = 1, the β_{ij}'s are assumed to be random effects in the model, the analysis performed for this mixed model is called an "interblock analysis."

Routine ABIBD requires the y_{ijtm}'s to be entered in a single vector Y ordered lexicographically, so that the i subscript varies least rapidly, the j subscript the next most rapidly, and so forth. Formulas and interpretations for the analysis of balanced incomplete block designs are discussed by Anderson and Bancroft (1952, Chapters 19 and 24) and Kempthorne (1975, pages 532-539).

Comments

Workspace may be explicitly provided, if desired, by use of A2IBD/DA2IBD. The reference is:

```
CALL A2IBD (NTRT, NREP, NBLK, NTBLK, NRESP, Y, ITRT, INTER, IPRINT, AOV, SQSS,
    SSALT, TESTLF, YMEANS, SETRTD, EFNCY, WK)
```

The additional argument is:

WK — Work vector of length NTRT or 2 * NTRT.

Example

This example performs an intrablock analysis for a balanced incomplete block design using data discussed by Anderson and Bancroft (1952, pages 254-256). The responses are weight gains of rats fed $p = 9$ different rations. There are four replications with $k = 3$ blocks within each replicate. (Since $p = k^2$, this balanced incomplete block design is a balanced lattice design.) The data with the treatment numbers in parentheses are given in the following table:

1	1	(1): 20	(4): 15	(7): 11
1	2	(3): 8	(6): 18	(9): 26
1	3	(2): 18	(5): 16	(8): 2
2	1	(7): 8	(8): 12	(9): 16
2	2	(1): 20	(2): 2	(3): 2
2	3	(4): 20	(5): 6	(6): 2
3	1	(1): 13	(9): 19	(5): 14
3	2	(8): 14	(4): 34	(3): 2
3	3	(6): 14	(2): 20	(7): 14
4	1	(5): 19	(7): 23	(3): 6
4	2	(1): 22	(6): 12	(8): 2
4	3	(9): 27	(2): 7	(4): 20

```
      USE ABIBD_INT

      IMPLICIT   NONE
      INTEGER    NBLK, NREP, NRESP, NTBLK, NTRT
      PARAMETER  (NBLK=12, NREP=4, NRESP=1, NTBLK=3, NTRT=9)
!
      INTEGER    IPRINT, ITRT(NBLK*NTBLK)
      REAL       AOV(15), Y(NBLK*NTBLK*NRESP)
!
      DATA Y/20.0, 15.0, 11.0, 8.0, 18.0, 26.0, 18.0, 16.0, 2.0, 8.0, &
          12.0, 16.0, 20.0, 2.0, 2.0, 20.0, 6.0, 2.0, 13.0, 19.0, &
          14.0, 14.0, 34.0, 2.0, 14.0, 20.0, 14.0, 19.0, 23.0, 6.0, &
          22.0, 12.0, 2.0, 27.0, 7.0, 20.0/
      DATA ITRT/1, 4, 7, 3, 6, 9, 2, 5, 8, 7, 8, 9, 1, 2, 3, 4, 5, 6, &
          1, 9, 5, 8, 4, 3, 6, 2, 7, 5, 7, 3, 1, 6, 8, 9, 2, 4/
!
```

```
      IPRINT = 3
      CALL ABIBD (NTRT, NREP, NBLK, NTBLK, NRESP, Y, ITRT, &
                  AOV, IPRINT=IPRINT)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
 Y             79.771     55.748           5.345          14           38.18

                    * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
  Source               DF     Squares      Square  Overall F  Larger F
  Model                19      1802.8       94.88      3.321    0.0095
  Error                16       457.2       28.57
  Corrected Total      35      2260.0

  * * * Decomposition of Variation Attributable to the Model * * *
                                   Sum of             Prob. of
       Source              DF     Squares          F  Larger F
       Replicates           3       219.6      2.561    0.0913
       Blocks within
          Replicates        8       127.1      0.556    0.7980
       Treatments
          (adjusted)        8      1456.1      6.370    0.0009

  * * * Replicate means * * *
     Replicate  Mean (N=4)
             1     14.8889
             2      9.7778
             3     16.0000
             4     15.3333

  * * * Block by Replicate Means * * *
    Replicate        Block  Mean (N=3)
            1            1     15.3333
            1            2     17.3333
            1            3     12.0000
            2            1     12.0000
            2            2      8.0000
            2            3      9.3333
            3            1     15.3333
            3            2     16.6667
            3            3     16.0000
            4            1     16.0000
            4            2     12.0000
            4            3     18.0000

  * * * Adjusted Treatment Means * * *
          Treatment  Mean (N=1)
                  1       22.11
                  2       11.67
                  3        0.67
                  4       23.89
                  5       14.78
                  6       11.11
                  7       12.89
```

8	6.44
9	22.44

* * * Treatment by Block Means * * *

Replicate	Block	Treatment	Mean (N=1)
1	1	1	20.0000
1	1	4	15.0000
1	1	7	11.0000
1	2	3	8.0000
1	2	6	18.0000
1	2	9	26.0000
1	3	2	18.0000
1	3	5	16.0000
1	3	8	2.0000
2	1	7	8.0000
2	1	8	12.0000
2	1	9	16.0000
2	2	1	20.0000
2	2	2	2.0000
2	2	3	2.0000
2	3	4	20.0000
2	3	5	6.0000
2	3	6	2.0000
3	1	1	13.0000
3	1	9	19.0000
3	1	5	14.0000
3	2	8	14.0000
3	2	4	34.0000
3	2	3	2.0000
3	3	6	14.0000
3	3	2	20.0000
3	3	7	14.0000
4	1	5	19.0000
4	1	7	23.0000
4	1	3	6.0000
4	2	1	22.0000
4	2	6	12.0000
4	2	8	2.0000
4	3	9	27.0000
4	3	2	7.0000
4	3	4	20.0000

ALATN

Analyzes a Latin square design.

Required Arguments

NTRT — Number of treatments. (Input)
NTRT must also be the number of rows and the number of columns.

NRESP — Number of repeated responses within each row-column position. (Input)

Y — Vector of length NTRT * NTRT * NRESP containing the responses. (Input)
The first NRESP elements of Y contain the responses for row 1, column 1; the second NRESP elements of Y contain the responses for row 1, column 2. The last NRESP elements of Y contain the responses for row NTRT, column NTRT.

ITRT — Vector of length NTRT * NTRT containing the treatment numbers for the responses in Y. (Input)
The treatment numbers must be from the set 1, 2, ..., NTRT. For I = 1, 2, ..., NTRT**2, element numbers (I - 1) * NRESP + 1 through (I - 1) * NRESP + NRESP of Y correspond to treatment number ITRT(I).

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for the model (blocks and treatments)
2	Degrees of freedom for error (interaction is pooled with the within-cell error)
3	Total (corrected) degrees of freedom
4	Sum of squares for the model (blocks and treatments)
5	Sum of squares for error (experimental error pooled with the within-cell error)
6	Total (corrected) sum of squares
7	Model mean square
8	Error mean square
9	*F* -statistic
10	*p*-value
11	R^2 (in percent)

I	AOV(I)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Overall mean of Y
15	Coefficient of variation (in percent)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Print AOV, EFSS, and TESTLF (if NRESP > 1) only.
2	Print YMEANS only.
3	All print is performed.

EFSS — Vector of length 12 containing statistics relating to the sums of squares for the effects in the model. (Output)
Elements of EFSS are described as follows:

Elem	Description
1, 2, 3	Degrees of freedom for rows, columns, and treatments, respectively.
4, 5, 6	Sum of squares for rows, columns, and treatments, respectively.
7, 8, 9	*F*-statistics for rows, columns, and treatments, respectively. *F*-statistics are computed using AOV(8) as the estimated error variance.
10-12	*p*-values associated with the *F*-statistics.

TESTLF — Vector of length 10 containing statistics relating to the test for lack of fit of the model.(Output if NRESP > 1)
If NRESP = 1, TESTLF is not referenced and can be a vector of length one. Elements of TESTLF are described as follows:

	Description
1	Degrees of freedom for experimental error
2	Degrees of freedom for within-cell error
3	Degrees of freedom for error (TESTLF(1) + TESTLF(2))
4	Sum of squares for experimental error
5	Sum of squares for within-cell error
6	Sum of squares for error
7	Mean square for experimental error
8	Mean square for within-cell error
9	*F* -statistic
10	*p*-value

YMEANS — Vector of length 3 * NTRT + NTRT * NTRT containing the row means, column means, treatment means, and the row-column means, respectively. (Output)

FORTRAN 90 Interface

Generic: CALL ALATN (NTRT, NRESP, Y, ITRT, AOV [, ...])
Specific: The specific interface names are S_ALATN and D_ALATN.

FORTRAN 77 Interface

Single: CALL ALATN (NTRT, NRESP, Y, ITRT, IPRINT, AOV, EFSS, TESTLF, YMEANS)
Double: The double precision name is DALATN.

Description

Routine ALATN performs an analysis for a Latin square design. The model is

$$y_{ijkm} = \mu + \alpha_i + \beta_j + \delta_k + \varepsilon_{ijkm} \qquad i, j, k = 1, 2, \ldots, p; m = 1, 2, \ldots, n$$

where the observed value of y_{ijkm} constitutes the m-th response on the k-th treatment in row i column j of the Latin square design; $\mu + \alpha_i + \beta_j + \delta_k$ is the population mean for the response, and the ε_{ijkm}'s are identically and independently distributed normal errors with mean zero and variance σ^2. This model assumes the row effects (α_i), column effects (β_j), and treatment effects (δ_k) are additive. Often in practice, there are interactions between two or more of these factors. For this reason, ALATN computes a test for nonadditivity (lack of fit), in addition to

summary statistics for the additive model. This test requires at least two responses in each cell. A test for nonadditivity with one response per cell in a Latin square design is discussed by Snedecor and Cochran (1967, pages 334-337).

Routine **ALATN** requires y_{ijk}'s to be entered in single vector **Y** with the data for each cell occupying contiguous elements. The cells must be in standard order, i.e., (1, 1), (1, 2), …, (1, *p*), (2, 1), (2, 2), …, (2, *p*), …, (*p*, 1), (*p*, 2), …, (*p*, *p*). A discussion of formulas and interpretations for the analysis of a Latin square design appears in many elementary statistics texts, e.g., Snedecor and Cochran (1967, pages 312-317).

Example

This example performs an analysis for a Latin square design using data discussed by Kirk (1982, Table 7.3-2, pages 312-317). The responses are thickness of tread remaining on each of 32 tires after 10,000 miles of driving. The tires are divided equally among four different types, labeled A, *B*, *C*, and D. Four cars are used in the study. The experiment is performed twice, sixteen tires are used in each experiment. Each of the sixteen tires occupies one of the four wheel positions on one of the cars. The data are given in the following table:

Right Front	A: 1, 2	B: 2, 3	C: 5, 6	D: 9, 8
Left Front	B: 3, 4	C: 8, 6	D: 9, 8	A: 2, 3
Right Rear	C: 5, 7	D: 10, 11	A: 3, 2	B: 5, 4
Left Rear	D: 7, 10	A: 6, 3	B: 3, 4	C: 6, 6

```
        USE ALATN_INT

        IMPLICIT   NONE
        INTEGER    NRESP, NTRT
        PARAMETER  (NRESP=2, NTRT=4)
!
        INTEGER    IPRINT, ITRT(NTRT*NTRT)
        REAL       AOV(15), Y(NTRT*NTRT*NRESP)
!
        DATA Y/1.0, 2.0, 2.0, 3.0, 5.0, 6.0, 9.0, 8.0, 3.0, 4.0, 8.0, &
            6.0, 9.0, 8.0, 2.0, 3.0, 5.0, 7.0, 10.0, 11.0, 3.0, 2.0, &
            5.0, 4.0, 7.0, 10.0, 6.0, 3.0, 3.0, 4.0, 6.0, 7.0/
        DATA ITRT/1, 2, 3, 4, 2, 3, 4, 1, 3, 4, 1, 2, 4, 1, 2, 3/
        DATA IPRINT/3/
!
        CALL ALATN (NTRT, NRESP, Y, ITRT, AOV, IPRINT=IPRINT)
        END
```

Output

```
   Dependent  R-squared   Adjusted  Est. Std. Dev.             Coefficient of
```

Variable	(percent)	R-squared	of Model Error	Mean	Var. (percent)
Y	89.809	85.640	1.044	5.375	19.43

* * * Analysis of Variance * * *

Source	DF	Sum of Squares	Mean Square	Overall F	Prob. of Larger F
Model	9	211.5	23.50	21.542	0.0000
Error	22	24.0	1.09		
Corrected Total	31	235.5			

* * * Decomposition of Variation Attributable to the Model * * *

Source	DF	Sum of Squares	F	Prob. of Larger F
Row	3	9.2	2.826	0.0622
Column	3	7.8	2.368	0.0983
Treatment	3	194.5	59.431	0.0000

Test for Lack of Fit

Source	DF	Sum of Squares	Mean Square	F	Prob. of Larger F
Experimental Error	6	5	0.833	0.702	0.6525
Within Cell	16	19	1.188		
Error	22	24			

* * * Row Means * * *

Row	Mean (N=4)
1	4.500
2	5.375
3	5.875
4	5.750

* * * Column Means * * *

Column	Mean (N=4)
1	4.875
2	6.125
3	5.000
4	5.500

* * * Treatment Means * * *

Treatment	Mean (N=4)
1	2.8
2	3.5
3	6.2
4	9.0

* * * Cell Means * * *

Row	Column	Mean (N=2)
1	1	1.500
1	2	2.500
1	3	5.500
1	4	8.500
2	1	3.500
2	2	7.000
2	3	8.500
2	4	2.500
3	1	6.000
3	2	10.500
3	3	2.500
3	4	4.500
4	1	8.500

4	2	4.500
4	3	3.500
4	4	6.500

ANWAY

Analyzes a balanced *n*-way classification model with fixed effects.

Required Arguments

NF — Number of factors (number of subscripts) in the model including error. (Input)

NL — Vector of length NF containing the number of levels for each of the factors. (Input)

Y — Vector of length NL(1) * NL(2) * ... * NL(NF) containing the responses. (Input)
Y must not contain NaN (not a number) for any of its elements, i.e., missing values are not allowed.

INTERA — Interaction option. (Input)
The absolute value of INTERA is the number of factors to be included in the highest-way interaction in the model. The sign of INTERA indicates if factor NF is error.

INTERA	Meaning
< 0	Factor NF is not error. Only (-INTERA + 1)-way and higher-way interactions are included in error.
> 0	Factor NF is error. Its main effect and all its interaction effects are pooled into the error with the other (INTERA + 1)-way and higher-way

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic
10	*p*-value

I	AOV(I)
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	Printing is not performed.
1	AOV and EFSS are printed.
2, -2	Only marginal means are printed. If IPRINT = 2, then all of YMEANS is printed. If IPRINT = -2, then marginal means higher than (|INTERA|) -way are not printed.
3, -3	AOV, EFSS, and all or some of YMEANS is printed. If IPRINT = 3, then all of YMEANS is printed. If IPRINT = -3, then marginal means higher than(|INTERA|) -way are not printed.

EFSS — NEF by 4 matrix containing statistics relating to the sums of squares for the effects in the model. (Output)
Here, NEF= BINOM(*n*, 1) + BINOM(*n*, 2) + … + BINOM(*n*, |INTERA|) where the IMSL subroutine BINOM (IMSL MATH/LIBRARY Special Functions) returns the binomial coefficient, and *n* is given by

$$n = \begin{cases} \text{NF} & \text{if INTERA is negative} \\ \text{NF} - 1 & \text{if INTERA is positive} \end{cases}$$

Suppose the factors are *A*, *B*, *C*, and error. With INTERA = 3, rows 1 through NEF would correspond to *A*, *B*, *C*, *AB*, *AC*, *BC*, and *ABC*, respectively. The columns of EFSS are as follows:

	Description
1	Degrees of freedom
2	Sum of squares
3	*F* -statistic
4	*p*-value

LDEFSS — Leading dimension of **EFSS** exactly as specified in the dimension statement in the calling program. (Input)
Deafult: **LDEFSS** = size (**EFSS** , 1)

YMEANS — Vector of length (**NL**(1) + 1) * (**NL**(2) + 1) * … * (**NL**(*n*) + 1) containing subgroup means. (Output)
See argument EFSS for a definition of *n*. Suppose that the factors are *A*, *B*, *C*, and error. The ordering of the means is grand mean, *A* means, *B* means, *C* means, *AB* means, *AC* means, *BC* means, and *ABC* means.

FORTRAN 90 Interface

Generic: **CALL ANWAY** (**NF**, **NL**, **Y**, **INTERA**, **AOV** [, ...])
Specific: The specific interface names are **S_ANWAY** and **D_ANWAY**.

FORTRAN 77 Interface

Single: **CALL ANWAY** (**NF**, **NL**, **Y**, **INTERA**, **IPRINT**, **AOV**, **EFSS**, **LDEFSS**, **YMEANS**)
Double: The double precision name is **DANWAY**.

Description

Routine **ANWAY** performs an analysis for an *n*-way classification design with balanced data. For balanced data, there must be an equal number of responses in each cell of the *n*-way layout. The effects are assumed to be fixed effects. The model is an extension of the twoway model to include *n* factors. The interactions (two-way, three-way, up to *n*-way) can be included in the model, or some of the higher-way interactions can be pooled into error. The argument **INTERA** specifies which interactions are to be pooled into error. For example, if three-way and higher-way interactions are to be pooled into error, set **INTERA** = **-** 2 or **INTERA** = 2. A positive **INTERA** indicates there are repeated responses within the *n*-way cells, while a negative **INTERA** indicates otherwise.

Routine **ANWAY** requires the responses as input into a single vector **Y** in lexicographical order so that the response subscript associated with the first factor varies least rapidly, the subscript associated with the second factor varies next most rapidly, and so forth. Hemmerle (1967, Chapter 5) discusses the computational method.

Comments

Workspace may be explicitly provided, if desired, by use of **A2WAY**/**DA2WAY**. The reference is:

```
CALL A2WAY (NF, NL, Y, INTERA, IPRINT, AOV, EFSS, LDEFSS, YMEANS, WK, IWK)
```

The additional arguments are as follows:

WK — Work vector of length 5 * 2^{n} + **NMEANS** + 4.

IWK — Work vector of length (**NF** + 2) * $2\mathbf{NF}^{-1}$ + (*n* + 2) * 2^{n-1} + *n* - 2.

Examples

Example 1

A two-way analysis of variance is performed with balanced data discussed by Snedecor and Cochran (1967, Table 12.5.1, page 347). The responses are the weight gains (in grams) of rats fed diets varying in two components—source of protein (*A*) and level of protein (*B*). Here, **INTERA** = 2 is used. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \quad i = 1, 2; j = 1, 2, 3; k = 1, 2, \ldots, 10$$

where

$$\sum_{i=1}^{2} \alpha_i = 0; \sum_{j=1}^{3} \beta_j = 0; \sum_{i=1}^{2} \gamma_{ij} = 0$$

for *j* = 1, 2, 3; and

$$\sum_{j=1}^{3} \gamma_{ij} = 0$$

for *i* = 1, 2.

The first responses in each cell in the two-way layout are given in the following table:

High	73, 102, 118, 104, 81, 107, 100, 87, 117, 111	98, 74, 56, 111, 95, 88, 82, 77, 86, 92	94, 79, 96, 98, 102, 102, 108, 91, 120, 105
Low	90, 76, 90, 64, 86, 51, 72, 90, 95, 78	107, 95, 97, 80, 98, 74, 74, 67, 89, 58	49, 82, 73, 86, 81, 97, 106, 70, 61, 82

```
      USE ANWAY_INT

      IMPLICIT   NONE
      INTEGER    NF, NOBS
      PARAMETER  (NF=3, NOBS=60)
!
      INTEGER    INTERA, IPRINT, NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA Y/73.0, 102.0, 118.0, 104.0, 81.0, 107.0, 100.0, 87.0, &
          117.0, 111.0, 90.0, 76.0, 90.0, 64.0, 86.0, 51.0, 72.0, &
          90.0, 95.0, 78.0, 98.0, 74.0, 56.0, 111.0, 95.0, 88.0, &
          82.0, 77.0, 86.0, 92.0, 107.0, 95.0, 97.0, 80.0, 98.0, &
          74.0, 74.0, 67.0, 89.0, 58.0, 94.0, 79.0, 96.0, 98.0, &
          102.0, 102.0, 108.0, 91.0, 120.0, 105.0, 49.0, 82.0, 73.0, &
          86.0, 81.0, 97.0, 106.0, 70.0, 61.0, 82.0/
      DATA NL/3, 2, 10/
!
      INTERA = 2
      IPRINT = 3
      CALL ANWAY (NF, NL, Y, INTERA, AOV, IPRINT=IPRINT)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean   Var. (percent)
 Y             28.477     21.854          14.65      87.87           16.67

                    * * * Analysis of Variance * * *
                                 Sum of        Mean             Prob. of
  Source               DF     Squares      Square  Overall F  Larger F
  Model                 5      4612.9       922.6      4.300    0.0023
  Error                54     11586.0       214.6
  Corrected Total      59     16198.9

          * * * Variation Due to the Model * * *
                            Sum of             Prob. of
  Source             DF     Squares          F  Larger F
  A                   2      266.53      0.621    0.5411
  B                   1     3168.27     14.767    0.0003
  A*B                 2     1178.13      2.746    0.0732
```

```
 * * * Subgroup Means * * *
  A Means (N=20)
  1       89.6000
  2       84.9000
  3       89.1000
  B Means (N=30)
  1       95.1333
  2       80.6000
   A*B Means (N=10)
  1   1      100.0000
  1   2       79.2000
  2   1       85.9000
  2   2       83.9000
  3   1       99.5000
  3   2       78.7000
```

Example 2

This example performs a three-way analysis of variance using data discussed by John (1971, pages 91-92). The responses are weights (in grams) of roots of carrots grown with varying amounts of applied nitrogen (*A*), potassium (*B*), and phosphorus (*C*). There is one response within each cell of the three-way layout. **INTERA** is set to -2 in order to pool the *ABC* three-factor interaction into error. (Note that the *ABC* interaction sum of squares, which is 186, is given incorrectly by John [1971, Table 5.2].) **IPRINT** is set to -3 so that the *ABC* means will not be printed (since |**INTERA**| is equal to 2). The three-way layout is given in the following table:

C_0	88.76	91.41	97.85	94.83	100.49	99.75	99.90	100.23	104.51
C_1	87.45	98.27	95.85	84.57	97.20	112.30	92.98	107.77	110.94
C_2	86.01	104.20	90.09	81.06	120.80	108.77	94.72	118.39	102.87

```
      USE ANWAY_INT

      IMPLICIT   NONE
      INTEGER    NF, NOBS
      PARAMETER  (NF=3, NOBS=27)
!
      INTEGER    INTERA, IPRINT, NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA Y/88.76, 87.45, 86.01, 91.41, 98.27, 104.20, 97.85, 95.85, &
          90.09, 94.83, 84.57, 81.06, 100.49, 97.20, 120.8, 99.75, &
          112.30, 108.77, 99.9, 92.98, 94.72, 100.23, 107.77, 118.39, &
          104.51, 110.94, 102.87/
      DATA NL/3, 3, 3/
!
      INTERA = -2
      IPRINT = -3
      CALL ANWAY (NF, NL, Y, INTERA, AOV, IPRINT=IPRINT)
```

```
END
```

Output

```
Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error       Mean   Var. (percent)
Y             92.804     76.612           4.819      98.96           4.869


                    * * * Analysis of Variance * * *
                                 Sum of        Mean             Prob. of
 Source               DF     Squares      Square  Overall F  Larger F
 Model                18      2395.7       133.1      5.731    0.0083
 Error                 8       185.8        23.2
 Corrected Total      26      2581.5

         * * * Variation Due to the Model * * *
                             Sum of             Prob. of
 Source             DF     Squares          F  Larger F
 A                   2      488.37     10.515    0.0058
 B                   2     1090.66     23.483    0.0004
 C                   2       49.15      1.058    0.3911
 A*B                 4      142.59      1.535    0.2804
 A*C                 4       32.35      0.348    0.8383
 B*C                 4      592.62      6.380    0.0131

 * * * Subgroup Means * * *
   A Means (N=9)
 1       93.3211
 2       99.9744
 3      103.5900

   B Means (N=9)
 1       90.0311
 2      104.3067
 3      102.5478
   C Means (N=9)
 1       97.5256
 2       98.5922
 3      100.7678
     A*B Means (N=3)
 1   1       87.4067
 1   2       97.9600
 1   3       94.5967
 2   1       86.8200
 2   2      106.1633
 2   3      106.9400
 3   1       95.8667
 3   2      108.7967
 3   3      106.1067
     A*C Means (N=3)
 1   1       92.6733
 1   2       93.8567
 1   3       93.4333
 2   1       98.3567
 2   2       98.0233
 2   3      103.5433
 3   1      101.5467
 3   2      103.8967
```

```
      3   3      105.3267
        B*C Means (N=3)
      1   1       94.4967
      1   2       88.3333
      1   3       87.2633
      2   1       97.3767
      2   2      101.0800
      2   3      114.4633
      3   1      100.7033
      3   2      106.3633
      3   3      100.5767
```

ABALD

Analyzes a balanced complete experimental design for a fixed, random, or mixed model.

Required Arguments

NL — Vector of length NF containing the number of levels for each of the factors. (Input)

Y — Vector of length NL(1) * NL(2) *...* NL(NF) containing the responses. (Input)
Y must not contain NaN (not a number) for any of its elements, i.e., missing values are not allowed.

NRF — For positive NRF, NRF is the number of random factors. (Input)
For negative NRF, -NRF is the number of random effects (sources of variation).

INDRF — Index vector of length |NRF| containing either the factor numbers to be considered random (for NRF positive) or containing the effect numbers to be considered random (for NRF negative). (Input)
If NRF = 0, INDRF is not referenced and can be a vector of length one.

NFEF — Vector of length NEF containing the number of factors associated with each effect in the model. (Input)

INDEF — Index vector of length NFEF(1) + NFEF(2) + ... + NFEF(NEF). (Input)
The first NFEF(1) elements give the factor numbers in the first effect. The next NFEF(2) elements give the the factor numbers in the second effect. The last NFEF(NEF) elements give the factor numbers in the last effect. Main effects must appear before their interactions. In general, an effect *E* cannot appear after an effect *F* if all of the indices for *E* appear also in *F* .

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square

I	AOV(I)
9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

Optional Arguments

NF — Number of factors (number of subscripts) in the model, including error. (Input)
Default: NF = size (NL,1).

NEF — Number of effects (sources of variation) due to the model excluding the overall mean and error. (Input)
Default: NEF = size (NFEF,1).

CONPER — Confidence level for two-sided interval estimates on the variance components, in percent. (Input)
CONPER percent confidence intervals are computed, hence, CONPER must be in the interval [0.0, 100.0). CONPER often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, ONECL in the interval [50.0, 100.0), set
CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	All is performed.
-*k*	Printing restricted to exclude marginal means higher than *k* ways. For example, only one-way and two-way marginal means will be printed if IPRINT = -2.

Let

$$n = \begin{cases} \text{NF} & \text{if INDEF contains one or more elements equal to NF} \\ \text{NF} - 1 & \text{otherwise} \end{cases}$$

The value of **IPRINT** must be between **-***n* and 1, inclusively.

MODEL — Model Option. (Input)
Default: **MODEL** = 0.

MODEL	Meaning
0	Searle model
1	Scheffe model

For the Scheffe model, effects corresponding to interactions of fixed and random factors have their sum over the subscripts corresponding to fixed factors equal to zero. Also, the variance of a random interaction effect involving some fixed factors has a multiplier for the associated variance component that involves the number of levels in the fixed factors. The Searle model has no summation restrictions on the random interaction effects and has a multiplier of one for each variance component.

EMS — Vector of length (**NEF** + 1) * (**NEF** + 2)/2 containing expected mean square coefficients. (Output)
Suppose the effects are *A*, *B*, and *AB*. The ordering of the coefficients in EMS is as follows:

	Error	***AB***	***B***	***A***
A	EMS(1)	EMS(2)	EMS(3)	EMS(4)
B	EMS(5)	EMS(6)	EMS(7)	
AB	EMS(8)	EMS(9)		
Error	EMS(10)			

VC — **NEF** + 1 by 9 matrix containing statistics relating to the particular variance components or effects in the model and the error. (Output)
Rows of **VC** correspond to the **NEF** effects plus error. Columns of **VC** are as follows:

Column	Description
1	Degrees of freedom
2	Sum of squares
3	Mean squares
4	*F* -statistic
5	*p*-value for *F* test
6	Variance component estimate
7	Percent of variance of *y* explained by random effect
8	Lower endpoint for a confidence interval on the variance component
9	Upper endpoint for a confidence interval on the variance component

Columns 6 through 9 contain NaN (not a number) if the effect is fixed, i.e., if there is no variance component to be estimated. If the variance component estimate is negative, columns 8 and 9 contain NaN.

LDVC — Leading dimension of VC exactly as specified in the dimension statement of the calling program. (Input)
Deafult: LDVC = size(VC ,1).

YMEANS — Vector of length (NL(1) + 1) * (NL(2) + 1) * ... * (NL(*n*) + 1) containing the subgroup means. (Output)
Suppose the factors are *A*, *B*, and *C*. The ordering of the means is grand mean, *A* means, *B* means, *C* means, *AB* means, *AC* means, *BC* means, and *ABC* means.

FORTRAN 90 Interface

Generic: CALL ABALD (NL, Y, NRF, INDRF, NFEF, INDEF, AOV [, ...])
Specific: The specific interface names are S_ABALD and D_ABALD.

FORTRAN 77 Interface

Single: CALL ABALD (NF, NL, Y, NRF, INDRF, NEF, NFEF, INDEF, CONPER, IPRINT, MODEL, AOV, EMS, VC, LDVC, YMEANS)
Double: The double precision name is DABALD.

Description

Routine **ABALD** analyzes a balanced complete experimental design for a fixed, random, or mixed model. The analysis includes an analysis of variance table, and computation of subgroup means and variance component estimates. A choice of two parameterizations of the variance components for the model can be made.

Scheffé (1959, pages 274-289) discusses the parameterization for **MODEL** = 1. For example, consider the following model equation with fixed factor *A* and random factor *B*:

$$y_{ijk} = \mu + \alpha_i + b_j + c_{ij} + e_{ijk} \quad i = 1, 2, \ldots, a; j = 1, 2, \ldots, b; k = 1, 2, \ldots, n$$

The fixed effects α_i's are subject to the restriction

$$\sum_{i=1}^{a} \alpha_i = 0$$

the b_j's are random effects identically and independently distributed

$$N\left(0, \sigma_B^2\right)$$

c_{ij} are interaction effects each distributed

$$N\left(0, \frac{a-1}{a}\sigma_{AB}^2\right)$$

and are subject to the restrictions

$$\sum_{i=1}^{a} c_{ij} = 0 \quad \text{for} \quad j = 1, 2, \ldots, b$$

and the e_{ijk}'s are errors identically and independently distributed $N(0, \sigma^2)$. In general, interactions of fixed and random factors have sums over subscripts corresponding to fixed factors equal to zero. Also in general, the variance of a random interaction effect is the associated variance component times a product of ratios for each fixed factor in the random interaction term. Each ratio depends on the number of levels in the fixed factor. In the earlier example, the random interaction *AB* has the ratio $(a - 1)/a$ as a multiplier of

$$\sigma_{AB}^2$$

and

$$\text{var}\left(y_{ijk}\right) = \sigma_B^2 + \frac{a-1}{a}\sigma_{AB}^2 + \sigma^2$$

In a three-way crossed classification model, an *ABC* interaction effect with *A* fixed, *B* random, and *C* fixed would have variance

$$\frac{(a-1)(c-1)}{ac}\sigma^2_{ABC}$$

Searle (1971, pages 400-401) discusses the parameterization for **MODEL** = 0. This parameterization does not have the summation restrictions on the effects corresponding to interactions of fixed and random factors. Also, the variance of each random interaction term is the associated variance component, i.e., without the multiplier. This parameterization is also used with unbalanced data, which is one reason for its popularity with balanced data also. In the earlier example,

$$\text{var}\left(y_{ijk}\right) = \tilde{\sigma}^2_B + \tilde{\sigma}^2_{AB} + \sigma^2$$

Searle (1971, pages 400-404) compares these two parameterizations. Hocking (1973) considers these different parameterizations and concludes they are equivalent because they yield the same variance-covariance structure for the responses. Differences in covariances for individual terms, differences in expected mean square coefficients and differences in *F* tests are just a consequence of the definition of the individual terms in the model and are not caused by any fundamental differences in the models. For the earlier two-way model, Hocking states that the relations between the two parameterizations of the variance components are

$$\sigma^2_B = \tilde{\sigma}^2_B + \tfrac{1}{a}\tilde{\sigma}^2_{AB}$$
$$\sigma^2_{AB} = \tilde{\sigma}^2_{AB}$$

where

$$\tilde{\sigma}^2_B \text{ and } \tilde{\sigma}^2_{AB}$$

are the variance components in the parameterization with **MODEL** = 0.

The computations for degrees of freedom and sums of squares are the same regardless of the option specified by **MODEL**. **ABALD** first computes degrees of freedom and sum of squares for a full factorial design. Degrees of freedom for effects in the factorial design that are missing from the specified model are pooled into the model effect containing the fewest subscripts but still containing the factorial effect. If no such model effect exists, the factorial effect is pooled into error. If more than one such effect exists, a terminal error message is issued indicating a misspecified model.

The analysis of variance method is used for estimating the variance components. This method solves a linear system in which the mean squares are set to the expected mean squares. A problem that Hocking (1985, pages 324-330) discusses is that this method can yield a negative variance component estimate. Hocking suggests a diagnostic procedure for locating the cause of the negative estimate. It may be necessary to re-examine the assumptions of the model.

The percentage of variation explained by each random effect is computed (output in **VC**(*i*, 7)) as the variance of the associated random effect divided by the variance of *y*. The two parameterizations can lead to different values because of the different definitions of the individual terms in the model. For example, the percentage associated with the *AB* interaction term in the earlier two-way mixed model is computed for **MODEL** = 1 using the formula

$$VC(3,7) = \frac{\frac{a-1}{a}\sigma^2_{AB}}{\sigma^2_B + \frac{a-1}{a}\sigma^2_{AB} + \sigma^2}$$

while for the parameterization **MODEL** = 0, the percentage is computed using the formula

$$VC(3,7) = \frac{\tilde{\sigma}^2_{AB}}{\tilde{\sigma}^2_B + \tilde{\sigma}^2_{AB} + \sigma^2}$$

In each case, the variance compenents are replaced by their estimates (stored in **VC**(*i*, 6)).

Confidence intervals on the variance components are computed using the method discussed by Graybill (1976, Theorem 15.3.5, page 624, and Note 4, page 620). Routine **CIDMS** is used.

Comments

Workspace may be explicitly provided, if desired, by use of **A2ALD**/**DA2ALD**. The reference is:

```
CALL A2ALD (NF, NL, Y, NRF, INDRF, NEF, NFEF, INDEF, CONPER, IPRINT, MODEL, AOV,
    EMS, VC, LDVC, YMEANS, WK, IWK, CHWK)
```

The additional arguments are as follows:

WK — Work vector of length 3 * 2**NF** + 2 * **NEF**+ *m* + 4.

IWK — Work vector of length max(2**NF**, **NF** + **NEF** + **LINDEF**)+ 2**NF** - 1 + **NF** * $2^{\text{NF}-1}$.

CHWK — **CHARACTER** * 13 vector of length max(**NEF** + 3, 2^{n} - 1). If **IPRINT** = 0, **CHWK** is not referenced and can be a vector of length one.

Examples

Example 1

An analysis of a generalized randomized block design is performed using data discussed by Kirk (1982, Table 6.10-1, pages 293-297). The model is

$$y_{ijk} = \mu + \alpha_i + b_j + c_{ij} + e_{ijk} \qquad i = 1, 2, 3, 4; j = 1, 2, 3, 4; k = 1, 2$$

where y_{ijk} is the response for the k-th experimental unit in block j with treatment i; the α_i's are the treatment effects and are subject to the restriction

$$\sum_{i=1}^{2} \alpha_i = 0$$

the b_j's are block effects identically and independently distributed

$$N\left(0,\sigma_B^2\right)$$

c_{ij} are interaction effects each distributed

$$N(0,\tfrac{3}{4}\sigma_{AB}^2)$$

and are subject to the restrictions

$$\sum_{i=1}^{4} c_{ij} = 0 \text{ for } j = 1,2,3,4$$

and the e_{ijk}'s are errors, identically and independently distributed $N(0, \sigma^2)$. The interaction effects are assumed to be distributed independently of the errors.

The data are given in the following table:

1	3, 6	3, 1	2, 2	3, 2
2	4, 5	4, 2	3, 4	3, 3
3	7, 8	7, 5	6, 5	6, 6
4	7, 8	9, 10	10, 9	8, 11

```
      USE ABALD_INT

      IMPLICIT   NONE
      INTEGER    LINDEF, NEF, NF, NOBS, NRF
      PARAMETER  (LINDEF=4, NEF=3, NF=3, NOBS=32, NRF=2)
!
      INTEGER    INDEF(LINDEF), INDRF(NRF), IPRINT, MODEL, NFEF(NEF), &
                 NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA NL/4, 4, 2/
      DATA INDRF/2, 3/
      DATA NFEF/1, 1, 2/
```

```
      DATA INDEF/1, 2, 1, 2/
      DATA Y/3.0, 6.0, 3.0, 1.0, 2.0, 2.0, 3.0, 2.0, 4.0, 5.0, 4.0, &
          2.0, 3.0, 4.0, 3.0, 3.0, 7.0, 8.0, 7.0, 5.0, 6.0, 5.0, &
          6.0, 6.0, 7.0, 8.0, 9.0, 10.0, 10.0, 9.0, 8.0, 11.0/
!
      IPRINT = 1
      MODEL  = 1
      CALL ABALD  (NL, Y, NRF, INDRF, NFEF, INDEF, AOV, IPRINT=IPRINT, &
      MODEL=MODEL)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean   Var. (percent)
 Y             91.932     84.368           1.09      5.375           20.27


                    * * * Analysis of Variance * * *
                                 Sum of        Mean              Prob. of
  Source               DF     Squares      Square  Overall F   Larger F
  Model                15       216.5       14.43     12.154     0.0000
  Error                16        19.0        1.19
  Corrected Total      31       235.5

                                 Sum of        Mean              Prob. of
  Source               DF     Squares      Square         F     Larger F
  A                     3      194.50     64.8333     32.873     0.0000
  B                     3        4.25      1.4167      1.193     0.3440
  AB                    9       17.75      1.9722      1.661     0.1802

       * * * EMS * * *
         Error  AB   B   A
  A          1   2   0   8
  B          1   0   8
  AB         1   2
  Error      1

                  * * * Variance Components * * *
                                      95.0% Confidence Interval
  Variance                            -------------------------
  Component     Estimate     Percent   Lower Limit   Upper Limit
  B               0.0286       1.897       0.00000        2.3168
  AB              0.3924      19.483       0.00000        2.7580
  Error           1.1875      78.621       0.65868        2.7506

  * * * Subgroup Means * * *
   A Means (N=8)
  1        2.7500
  2        3.5000
  3        6.2500
  4        9.0000
   B Means (N=8)
  1        6.0000
  2        5.1250
  3        5.1250
  4        5.2500
    AB  Means (N=2)
```

1	1	4.5000
1	2	2.0000
1	3	2.0000
1	4	2.5000
2	1	4.5000
2	2	3.0000
2	3	3.5000
2	4	3.0000
3	1	7.5000
3	2	6.0000
3	3	5.5000
3	4	6.0000
4	1	7.5000
4	2	9.5000
4	3	9.5000
4	4	9.5000

Example 2

An analysis of a split-plot design is performed using data discussed by Milliken and Johnson (1984, Table 24.1, page 297). Label the two treatment factors *A* and *C*. Denote the treatment combination aick as that at the *i*-th level of *A* and the *k*-th level of *C*. The model is

$$y_{ijk} = \mu + \alpha_i + b_j + d_{ij} + \delta_{ik} + e_{ijk} \quad i = 1, 2; j = 1, 2; k = 1, 2, 3, 4$$

where y_{ijk} is the response for the *j*-th experimental unit with treatment combination $a_i c_k$; the α_i's are the effects due to treatment factor *A*, the b_j's are block effects identically and independently distributed

$$N\left(0, \sigma_B^2\right)$$

the d_{ij} are split plot errors that are identically and independently distributed

$$N\left(0, \sigma_{AB}^2\right)$$

the γ_k's are the effects due to treatment factor *C*, the δ_{ik}'s are interaction effects between factors *A* and *C*, and the e_{ijk}'s are identically and independently distributed $N(0, \sigma^2)$. The block effects, whole plot errors, and split plot errors are independent.

The data are given in the following table.

1	1	35.4	37.9
	2	41.6	40.3
2	1	36.7	38.2
	2	42.7	41.6
3	1	34.8	36.4
	2	43.6	42.8
4	1	39.5	40.0
	2	44.5	47.6

```
      USE ABALD_INT

      IMPLICIT   NONE
      INTEGER    LDVC, LINDEF, NEF, NF, NOBS, NRF
      PARAMETER  (LINDEF=7, NEF=5, NF=3, NOBS=16, NRF=1)
!
      INTEGER    INDEF(LINDEF), INDRF(NRF), IPRINT, NFEF(NEF), NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA NL/4, 2, 2/
      DATA INDRF/2/
      DATA NFEF/1, 1, 2, 1, 2/
      DATA INDEF/1, 2, 1, 2, 3, 1, 3/
      DATA Y/35.4, 37.9, 41.6, 40.3, 36.7, 38.2, 42.7, 41.6, 34.8, &
          36.4, 43.6, 42.8, 39.5, 40.0, 44.5, 47.6/
!
      IPRINT = -2
      CALL ABALD (NL, Y, NRF, INDRF, NFEF, INDEF, AOV, IPRINT=IPRINT)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error       Mean   Var. (percent)
 Y            95.574     83.401          1.452      40.22           3.609

                    * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
  Source              DF      Squares      Square  Overall F  Larger F
  Model               11        182.0       16.55      7.852    0.0306
  Error                4          8.4        2.11
  Corrected Total     15        190.4

                                Sum of        Mean             Prob. of
```

```
Source                DF     Squares      Square          F  Larger F
A                      3      40.190      13.397      5.802    0.0914
B                      1     131.102     131.102     56.775    0.0048
AB                     3       6.928       2.309      1.096    0.4476
C                      1       2.250       2.250      1.068    0.3599
AC                     3       1.550       0.517      0.245    0.8612

         * * * EMS * * *
       Error  AC   C  AB   B   A
A          1   0   0   2   0   4
B          1   0   0   2   8
AB         1   0   0   2
C          1   0   8
AC         1   2
Error      1

                * * * Variance Components * * *
                                        95.0% Confidence Interval
Variance                                -------------------------
Component     Estimate     Percent   Lower Limit   Upper Limit
B               16.099      87.938        2.2597       16686.7
AB               0.101       0.551        0.0000          15.1
Error            2.108      11.512        0.7565          17.4

* * * Subgroup Means * * *
 A Means (N=4)
1       38.8000
2       39.8000
3       39.4000
4       42.9000
 B Means (N=8)
1       37.3625
2       43.0875
 C Means (N=8)
1       39.8500
2       40.6000
  AB  Means (N=2)
1  1       36.6500
1  2       40.9500
2  1       37.4500
2  2       42.1500
3  1       35.6000
3  2       43.2000
4  1       39.7500
4  2       46.0500
  AC  Means (N=2)
1  1       38.5000
1  2       39.1000
2  1       39.7000
2  2       39.9000
3  1       39.2000
3  2       39.6000
4  1       42.0000
4  2       43.8000
  BC  Means (N=4)
1  1       36.6000
1  2       38.1250
2  1       43.1000
2  2       43.0750
```

Example 3

An analysis of a split-plot factorial design is performed using data discussed by Kirk (1982, Table 11.2-1, pages 493 - 496). Label the two treatment factors *A* and *C*. Denote the treatment combination $a_i c_k$ as that at the *i*-th level of *A* and the *k*-th level of *C*. The model is

$$y_{ijk} = \mu + \alpha_i + b_{jj} + \gamma_k + \delta_{ik} + e_{ijk} \qquad i = 1, 2; j = 1, 2, 3, 4; k = 1, 2, 3, 4$$

where y_{ijk} is the response for the *j*-th experimental unit with treatment combination $a_i c_k$; the α_i's are the effects due to treatment factor *A* and are subject to the restriction

$$\sum_{i=1}^{2} \alpha_i = 0$$

the b_{ij}'s are block effects identically and independently distributed

$$N\left(0, \sigma_B^2\right)$$

the γ_k's are the effects due to treatment factor *C* and are subject to the restriction

$$\sum_{k=1}^{4} \gamma_k = 0$$

the δ_{ik}'s are interaction effects between factors *A* and *C* and are subject to the restrictions

$$\sum_{i=1}^{2} \delta_{ik} = 0$$

for each *k*, and

$$\sum_{k=1}^{4} \delta_{ik} = 0$$

for each *i*, and the e_{ijk}'s are identically and independently distributed $N(0, \sigma^2)$. The block effects are assumed to be distributed independently of the errors.

The data are given in the following table:

1	1	3	4	7	7
	2	6	5	8	8
	3	3	4	7	9
	4	3	3	6	8
2	5	1	2	5	10
	6	2	3	6	10
	7	2	4	5	9
	8	2	3	6	11

```
      USE ABALD_INT

      IMPLICIT   NONE
      INTEGER    LINDEF, NEF, NF, NOBS, NRF
      PARAMETER  (LINDEF=6, NEF=4, NF=3, NOBS=32, NRF=-1)
!
      INTEGER    INDEF(LINDEF), INDRF(-NRF), IPRINT, MODEL, NFEF(NEF), &
                 NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA NL/2, 4, 4/
      DATA INDRF/2/
      DATA NFEF/1, 2, 1, 2/
      DATA INDEF/1, 1, 2, 3, 1, 3/
      DATA Y/3.0, 4.0, 7.0, 7.0, 6.0, 5.0, 8.0, 8.0, 3.0, 4.0, 7.0, 9.0, &
           3.0, 3.0, 6.0, 8.0, 1.0, 2.0, 5.0, 10.0, 2.0, 3.0, 6.0, &
            10.0, 2.0, 4.0, 5.0, 9.0, 2.0, 3.0, 6.0, 11.0/
!
      IPRINT = 1
      MODEL  = 1
      CALL ABALD (NL, Y, NRF, INDRF, NFEF, INDEF, AOV, IPRINT=IPRINT, &
      MODEL=MODEL)
      END
```

Output

```
 Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
 Variable   (percent)  R-squared  of Model Error        Mean  Var. (percent)
 Y             96.125     93.327           0.712       5.375           13.25

                    * * * Analysis of Variance * * *
                                Sum of        Mean              Prob. of
  Source               DF     Squares      Square  Overall F  Larger F
  Model                13       226.4       17.41     34.350    0.0000
  Error                18         9.1        0.51
  Corrected Total      31       235.5
```

```
                                    Sum of        Mean             Prob. of
Source                  DF        Squares      Square          F  Larger F
A                        1          3.125      3.1250      2.000    0.2070
AB                       6          9.375      1.5625      3.082    0.0296
C                        3        194.500     64.8333    127.890    0.0000
AC                       3         19.375      6.4583     12.740    0.0001
       * * * EMS * * *
       Error  AC   C  AB   A
A          1   0   0   4  16
AB         1   0   0   4
C          1   0   8
AC         1   4
Error      1
                * * * Variance Components * * *
                                       95.0% Confidence Interval
Variance                               -------------------------
Component     Estimate     Percent   Lower Limit   Upper Limit
AB             0.26389      34.234       0.00000        1.7760
Error          0.50694      65.766       0.28944        1.1086

* * * Subgroup Means * * *
A Means (N=16)
1        5.6875
2        5.0625
 B Means (N=8)
1        4.8750
2        6.0000
1        5.3750
2        5.2500
 C Means (N=8)
1        2.7500
2        3.5000
1        6.2500
2        9.0000
  AB  Means (N=4)
1  1        5.2500
1  2        6.7500
1  3        5.7500
1  4        5.0000
2  1        4.5000
2  2        5.2500
2  3        5.0000
2  4        5.5000
  AC  Means (N=4)
1  1        3.7500
1  2        4.0000
1  3        7.0000
1  4        8.0000
2  1        1.7500
2  2        3.0000
2  3        5.5000
2  4       10.0000
  BC  Means (N=2)
1  1        2.0000
1  2        3.0000
1  3        6.0000
1  4        8.5000
2  1        4.0000
2  2        4.0000
2  3        7.0000
```

```
      2  4        9.0000
      1  1        2.5000
      1  2        4.0000
      1  3        6.0000
      1  4        9.0000
      2  1        2.5000
      2  2        3.0000
      2  3        6.0000
      2  4        9.5000
        ABC   Means (N=1)
      1  1  1        3.0000
      1  1  2        4.0000
      1  1  3        7.0000
      1  1  4        7.0000
      1  2  1        6.0000
      1  2  2        5.0000
      1  2  3        8.0000
      1  2  4        8.0000
      1  3  1        3.0000
      1  3  2        4.0000
      1  3  3        7.0000
      1  3  4        9.0000
      1  4  1        3.0000
      1  4  2        3.0000
      1  4  3        6.0000
      1  4  4        8.0000
      2  1  1        1.0000
      2  1  2        2.0000
      2  1  3        5.0000
      2  1  4       10.0000
      2  2  1        2.0000
      2  2  2        3.0000
      2  2  3        6.0000
      2  2  4       10.0000
      2  3  1        2.0000
      2  3  2        4.0000
      2  3  3        5.0000
      2  3  4        9.0000
      2  4  1        2.0000
      2  4  2        3.0000
      2  4  3        6.0000
      2  4  4       11.0000
```

ANEST

Analyzes a completely nested random model with possibly unequal numbers in the subgroups.

Required Arguments

NF — Number of factors (number of subscripts) in the model including error. (Input)

IEQ — Equal numbers option. (Input)

IEQ	Description
0	Unequal numbers in the subgroups
1	Equal numbers in the subgroups

NL — Vector with the number of levels. (Input)
If IEQ = 1, NL is of length NF and contains the number of levels for each of the factors. In this case, the following additional variables are referred to in the description of ANEST:

	Description
LNL	NL(1) + NL(1) * NL(2) + … + NL(1) * NL(2) * … * NL(NF - 1)
LNLNF	NL(1) * NL(2) * …* NL(NF - 1)
NOBS	The number of observations. NOBS equals NL(1) * NL(2) * … * NL(NF).

If IEQ = 0, NL contains the number of levels of each factor at each level of the factor in which it is nested. In this case, the following additional variables are referred to in the description of ANEST:

	Description
LNL	Length of NL.
LNLNF	Length of the subvector of NL for the last factor.
NOBS	Number of observations. NOBS equals the sum of the last LNLNF elements of NL.

For example, a random one-way model with two groups, five responses in the first group and ten in the second group, would have LNL = 3, LNLNF = 2, NOBS = 15, NL(1) = 2, NL(2) = 5, and NL(3) = 10.

Y — Vector of length NOBS containing the responses. (Input)
The elements of Y are ordered lexicographically, i.e., the last model subscript changes most rapidly, the next next to last model subscript changes the next most rapidly, and so forth, with the first subscript changing the slowest.

AOV — Vector of length 15 containing statistics relating to the analysis of variance. (Output)

I	AOV(I)
1	Degrees of freedom for regression
2	Degrees of freedom for error
3	Total degrees of freedom
4	Sum of squares for regression
5	Sum of squares for error
6	Total sum of squares
7	Regression mean square
8	Error mean square
9	*F*-statistic
10	*p*-value
11	R^2 (in percent)
12	Adjusted R^2 (in percent)
13	Estimated standard deviation of the model error
14	Mean of the response (dependent) variable
15	Coefficient of variation (in percent)

Optional Arguments

CONPER — Confidence level for two-sided interval estimates on the variance components, in percent. (Input)
Default: CONPER = 95.0.
CONPER percent confidence intervals are computed, hence, CONPER must be in the interval [0.0, 100.0). CONPER often will be 90.0, 95.0, or 99.0. For one-sided intervals with confidence level ONECL, ONECL in the interval [50.0, 100.0), set
CONPER = 100.0 - 2.0 * (100.0 - ONECL).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

EMS — Vector of length (NF + 1) * NF/2 with expected mean square coefficients. (Output)

VC — NF by 9 matrix containing statistics relating to the particular variance components in the model. (Output)
Rows of VC correspond to the NF factors. Columns of VC are as follows:

	Description
1	Degrees of freedom
2	Sum of squares
3	Mean squares
4	*F*-statistic
5	*p*-value for *F* test
6	Variance component estimate
7	Percent of variance explained by variance component
8	Lower endpoint for a confidence interval on the variance component
9	Upper endpoint for a confidence interval on the variance component

A test for the error variance equal to zero cannot be performed. VC(NF, 4) and VC(NF, 5) are set to NaN (not a number).

LDVC — Leading dimension of VC exactly as specified in the dimension statement in the calling program. (Input)
Default: LDVC= size(VC ,1)

YMEANS — Vector containing the subgroup means. (Output)

IEQ	Length of YMEANS
0	1 + NL(1) + NL(2) + … NL(LNL - LNLNF) (See the description of argument NL for definitions of LNL and LNLNF.)
1	1 + NL(1) + NL(1) * NL(2) + … + NL(1) * NL(2) * … * NL(NF - 1)

If the factors are labeled *A*, *B*, *C*, and error, the ordering of the means is grand mean, *A* means, *AB* means, and then *ABC* means.

NMISS — Number of missing values in Y. (Output)
Elements of Y equal to NaN (not a number) are omitted from the computations.

FORTRAN 90 Interface

Generic: CALL ANEST (NF, IEQ, NL, Y, AOV [, …])
Specific: The specific interface names are S_ANEST and D_ANEST.

FORTRAN 77 Interface

Single: CALL ANEST (NF, IEQ, NL, Y, CONPER, IPRINT, AOV, EMS, VC, LDVC, YMEANS, NMISS)

Double: The double precision name is DANEST.

Description

Routine ANEST analyzes a nested random model with equal or unequal numbers in the subgroups. The analysis includes an analysis of variance table and computation of subgroup means and variance component estimates. Anderson and Bancroft (1952, pages 325-330) discuss the methodology. The analysis of variance method is used for estimating the variance components. This method solves a linear system in which the mean squares are set to the expected mean squares. A problem that Hocking (1985, pages 324-330) discusses is that this method can yield negative variance component estimates. Hocking suggests a diagnostic procedure for locating the cause of a negative estimate. It may be necessary to reexamine the assumptions of the model.

Comments

Workspace may be explicitly provided, if desired, by use of A2EST/DA2EST. The reference is:

```
CALL A2EST (NF, IEQ, NL, Y, CONPER, IPRINT, AOV, EMS, VC, LDVC, YMEANS, NMISS, WK,
    IWK, CHWK)
```

The additional arguments are as follows:

WK — Work vector of length NOBS.

IWK — Work vector of length 5 * NF + (2 * LNL - LNLNF).

CHWK — CHARACTER * 10 vector of length 2 * NF + 1. If IPRINT = 0, CHWK is not referenced and can be a vector of length one.

Examples

Example 1

An analysis of a three-factor nested random model with equal numbers in the subgroups is performed using data discussed by Snedecor and Cochran (1967, Table 10.16.1, pages 285-288). The responses are calcium concentrations (in percent, dry basis) as measured in the leaves of turnip greens. Four plants are taken at random, then three leaves are randomly selected from each plant. Finally, from each selected leaf two samples are taken to determine calcium concentration. The model is

$$y_{ijk} = \mu + \alpha_i + \beta_{jj} + e_{ijk} \quad i = 1, 2, 3, 4; j = 1, 2, 3; k = 1, 2$$

where y_{ijk} is the calcium concentration for the *k*-th sample of the *j*-th leaf of the *i*-th plant, the α_i's are the plant effects and are taken to be independently distributed

$$N\left(0,\sigma^2\right)$$

the β_{ij}'s are leaf effects each independently distributed

$$N\left(0,\sigma_\beta^2\right)$$

and the ε_{ijk}'s are errors each independently distributed $N(0, \sigma^2)$. The effects are all assumed to be independently distributed. The data are given in the following table:

Plant	Leaf	Samples	
1	1	3.28	3.09
	2	3.52	3.48
	3	2.88	2.80
2	1	2.46	2.44
	2	1.87	1.92
	3	2.19	2.19
3	1	2.77	2.66
	2	3.74	3.44
	3	2.55	2.55
4	1	3.78	3.87
	2	4.07	4.12
	3	3.31	3.31

```
      USE ANEST_INT

      IMPLICIT   NONE
      INTEGER    NF, NOBS
      PARAMETER  (NF=3, NOBS=24)
!
      INTEGER    IEQ, IPRINT, NL(NF)
      REAL       AOV(15), Y(NOBS)
!
      DATA Y/3.28, 3.09, 3.52, 3.48, 2.88, 2.80, 2.46, 2.44, 1.87, &
          1.92, 2.19, 2.19, 2.77, 2.66, 3.74, 3.44, 2.55, 2.55, 3.78, &
          3.87, 4.07, 4.12, 3.31, 3.31/
      DATA NL/4, 3, 2/
!
      IEQ    = 1
      IPRINT = 1
      CALL ANEST (NF, IEQ, NL, Y, AOV, IPRINT=IPRINT)
      END
```

Output

```
Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
Y            99.222     98.510        0.08158      3.012           2.708

                  * * * Analysis of Variance * * *
                                 Sum of        Mean             Prob. of
 Source               DF       Squares      Square  Overall F  Larger F
 Model                11         10.19      0.9264    139.216    0.0000
 Error                12          0.08      0.0067
 Corrected Total      23         10.27
                                 Sum of        Mean             Prob. of
 Source               DF       Squares      Square         F   Larger F
 A                     3       7.56034     2.52011     7.665    0.0097
 B                     8       2.63020     0.32878    49.406    0.0000

 * * * Expected Mean Square Coefficients * * *
                Error    Effect B    Effect A
 Effect A           1           2           6
 Effect B           1           2
 Error              1

                * * * Variance Components * * *
                                      95.0% Confidence Interval
 Variance                             -------------------------
 Component     Estimate     Percent   Lower Limit   Upper Limit
 A              0.36522      68.530      0.039551        5.7867
 B              0.16106      30.221      0.069669        0.6004
 Error          0.00665       1.249      0.003422        0.0181

     A Means
 1        3.1750
 2        2.1783
 3        2.9517
 4        3.7433

      AB Means
 1  1        3.1850
 1  2        3.5000
 1  3        2.8400
 2  1        2.4500
 2  2        1.8950
 2  3        2.1900
 3  1        2.7150
 3  2        3.5900
 3  3        2.5500
 4  1        3.8250
 4  2        4.0950
 4  3        3.3100
```

Example 2

An analysis of a three-factor nested random model with unequal numbers in the subgroups is performed. The data are given in the following table:

1	1 2	23.0 31.0	19.0 37.0	
2	1 2	33.0 29.0	29.0	
3	1	36.0	29.0	33.0
4	1 2 3 4 5 6 7 8 9	11.0 23.0 33.0 23.0 26.0 39.0 20.0 24.0 36.0	21.0 18.0	
5	1	25.0	33.0	
6	1 2 3 4 5 6 7 8 9 10	28.0 25.0 32.0 41.0 35.0 16.0 30.0 40.0 32.0 44.0	31.0 42.0 36.0	

```
      USE ANEST_INT

      IMPLICIT   NONE
      INTEGER    LNL, NF, NOBS
      PARAMETER  (LNL=32, NF=3, NOBS=36)
!
      INTEGER    IEQ, IPRINT, NL(LNL)
      REAL       AOV(15), Y(NOBS)
!
      DATA Y/23.0, 19.0, 31.0, 37.0, 33.0, 29.0, 29.0, 36.0, 29.0, &
          33.0, 11.0, 21.0, 23.0, 18.0, 33.0, 23.0, 26.0, 39.0, 20.0, &
          24.0, 36.0, 25.0, 33.0, 28.0, 31.0, 25.0, 42.0, 32.0, 36.0, &
          41.0, 35.0, 16.0, 30.0, 40.0, 32.0, 44.0/
      DATA NL/6, 2, 2, 1, 9, 1, 10, 2, 2, 2, 1, 3, 2, 2, 1, 1, 1, 1, &
          1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1/
!
      IEQ    = 0
      IPRINT = 1
      CALL ANEST (NF, IEQ, NL, Y, AOV, IPRINT=IPRINT)
```

```
END
```

Output

```
Dependent  R-squared   Adjusted  Est. Std. Dev.              Coefficient of
Variable   (percent)  R-squared  of Model Error       Mean  Var. (percent)
Y             85.376     53.470           5.31      29.53           17.98

                  * * * Analysis of Variance * * *
                                Sum of        Mean             Prob. of
 Source               DF      Squares      Square  Overall F  Larger F
 Model                24       1810.8       75.45      2.676    0.0459
 Error                11        310.2       28.20
 Corrected Total      35       2121.0

                                Sum of        Mean             Prob. of
 Source               DF      Squares      Square          F  Larger F
 A                     5       461.42     92.2845      0.988    0.4601
 B                    19      1349.38     71.0202      2.519    0.0597

 * * * Expected Mean Square Coefficients * * *
                 Error    Effect B    Effect A
 Effect A      1.00000     1.96503     5.37778
 Effect B      1.00000     1.28990
 Error         1.00000

                 * * * Variance Components * * *
                                     95.0% Confidence Interval
 Variance                            -------------------------
 Component     Estimate     Percent   Lower Limit   Upper Limit
 A               -0.214         NaN           NaN           NaN
 B               33.199      54.073          0.00        100.59
 Error           28.197      45.927         14.15         81.29

     A Means
 1       27.5000
 2       30.3333
 3       32.6667
 4       24.9091
 5       29.0000
 6       33.2308
      AB Means
  1   1       21.0000
  1   2       34.0000
  2   1       31.0000
  2   2       29.0000
  3   1       32.6667
  4   1       16.0000
  4   2       20.5000
  4   3       33.0000
  4   4       23.0000
  4   5       26.0000
  4   6       39.0000
  4   7       20.0000
  4   8       24.0000
  4   9       36.0000
  5   1       29.0000
  6   1       29.5000
```

```
       6   2       33.5000
       6   3       34.0000
       6   4       41.0000
       6   5       35.0000
       6   6       16.0000
       6   7       30.0000
       6   8       40.0000
       6   9       32.0000
       6  10       44.0000
```

CTRST

Computes contrast estimates and sums of squares.

Required Arguments

NI — Vector of length NGROUP containing the number of responses for each of the NGROUP groups. (Input)

YMEANS — Vector of length NGROUP containing the sample mean for each group or each level of a classification variable. (Input)

C — NGROUP by NCTRST matrix containing in each column the coefficients for a particular contrast. (Input)

EST — Vector of length NCTRST containing the contrast estimates. (Output)

SS — Vector of length NCTRST containing the sum of squares associated with each contrast. (Output)

Optional Arguments

NGROUP — Number of groups or number of sample means involved in the contrasts. (Input)
Default: NGROUP = size (NI,1).

NCTRST — Number of contrasts. (Input)
Default: NCTRST = size (C,2).

LDC — Leading dimension of C exactly as specified in the dimension statement in the calling program. (Input)
Default: LDC = size (C,1).

FORTRAN 90 Interface

Generic: CALL CTRST (NI, YMEANS, C, EST, SS [, ...])
Specific: The specific interface names are S_CTRST and D_CTRST.

FORTRAN 77 Interface

Single: CALL CTRST (NGROUP, NI, YMEANS, NCTRST, C, LDC, EST, SS)

Double: The double precision name is **DCTRST**.

Description

Routine **CTRST** computes an estimate of a linear combination of means using the sample means input in **YMEANS**. The sum of squares associated with each estimate is also computed.

Contrasts (linear combinations of means whose coefficients sum to zero) are customarily of interest. Orthogonal contrasts (Neter and Wasserman 1974, pages 470-471) are often used to partition the among-groups sum of squares from a one-way analysis of variance. The following discussion uses the term "contrast", however, the term "linear combination of means," which places no restriction on the coefficients, is equally valid.

Let

$$\bar{y}_1, \bar{y}_2, \ \ldots, \bar{y}_k$$

be the *k*(= **NGROUP**) sample means, and let $\mu_1, \mu_2, \ldots, \mu_k$ be the associated population means. Let $c_{1j}, c_{2j}, \ldots, c_{kj}$ be the contrast coefficients for contrast *j* (stored in column *j* of the matrix *C*). The estimate of

$$l_j = \sum_{i=1}^{k} c_{ij}\mu_i$$

is

$$\hat{l}_j$$

(stored as the *j*-th element of **EST**) computed by

$$\hat{l}_j = \sum_{i=1}^{k} c_{ij}\bar{y}_i$$

The associated sum of squares Q_j (stored as the *j*-th element of *SS*) is computed by

$$Q_j = \frac{\hat{l}_j^2}{\sum_{i=1}^{k} c_{ij}^2 / n_i}$$

Comments

Informational Error

Type	Code	Description
1	1	A column of C does not sum to zero within the computed tolerance. Customarily, contrasts (linear combinations of means whose coefficients sum to zero) are of interest.

Example

The following example is taken from Neter and Wasserman (1974, Table 13.1, page 432, Table 14.3, page 463, pages 470-471). Three orthogonal contrasts are defined that partition the among-group sum of squares (258.0) from a one-way analysis of variance. The first contrast compares groups 1 and 2, the second contrast compares groups 3 and 4, the third contrast compares a weighted average of groups 1 and 2 with a weighted average of groups 3 and 4.

```
      USE CTRST_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NGROUP, LDC, NCTRST, I
      PARAMETER  (NGROUP=4, LDC=NGROUP, NCTRST=3)
      INTEGER    NI(NGROUP), J, NOUT
      REAL       EST(NCTRST), SS(NCTRST), C(LDC,NCTRST), YMEANS(NGROUP)
!
      DATA YMEANS/15.0, 13.0, 19.0, 27.0/
      DATA NI/2, 3, 3, 2/
      DATA (C(I,1),I=1,NGROUP)/1.0, -1.0, 0.0, 0.0/
      DATA (C(I,2),I=1,NGROUP)/0.0, 0.0, 1.0, -1.0/
      DATA (C(I,3),I=1,NGROUP)/0.4, 0.6, -0.6, -0.4/
!
      CALL CTRST (NI, YMEANS, C, EST, SS)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Contrast  Estimate  Sum of Squares'
      DO 10  J=1, NCTRST
         WRITE (NOUT,'(1X,I4,5X,F7.1,3X,F10.1)') J, EST(J), SS(J)
   10 CONTINUE
      END
```

Output

```
 Contrast Estimate Sum of Squares
   1         2.0          4.8
```

```
         2        -8.0         76.8
         3        -8.4        176.4
```

SCIPM

Computes simultaneous confidence intervals on all pairwise differences of means.

Required Arguments

NI — Vector of length NGROUP containing the number of observations in each mean. (Input)

YMEANS — Vector of length NGROUP containing the means. (Input)

DFS2 — Degrees of freedom for s^2. (Input)

S2 — s^2, the estimated variance of an observation. (Input)
The variance of YMEANS(I) is estimated by S2/NI(I).

STAT — NGROUP * (NGROUP - 1)/2 by 5 matrix containing the statistics relating to the difference of means. (Output)

	Description
1	Group number for the *i*-th mean
2	Group number for the *j*-th mean
3	Difference of means (*i*-th mean) - (*j*-th mean)
4	Lower confidence limit for the difference
5	Upper confidence limit for the difference

Optional Arguments

NGROUP — Number of means. (Input)
Default: NGROUP = size(NI,1).

IMETH — Method used for constructing confidence intervals on all pairwise differences of means. (Input)
Default: IMETH = 0.

IMETH	Method
0	Tukey (if equal group sizes), Tukey-Kramer method (otherwise)
1	Dunn-Sidak method
2	Bonferroni method
3	Scheffe method
4	One-at-a-time *t* method - *LSD* test

CONPER — Confidence percentage for the simultaneous interval estimation. (Input)
Default: CONPER = 95.0.

IMETH	CONPER
0	Percentage must be greater than or equal to 0.0 and less than 100.0.
≥ 1	Percentage must be greater than or equal to 90.0 and less than or equal to 99.0.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSTAT = size (STAT,1).

FORTRAN 90 Interface

Generic: CALL SCIPM (NI, YMEANS, DFS2, S2, STAT [, ...])
Specific: The specific interface names are S_SCIPM and D_SCIPM.

FORTRAN 77 Interface

Single: CALL SCIPM (NGROUP, NI, YMEANS, DFS2, S2, IMETH, CONPER, IPRINT, STAT, LDSTAT)
Double: The double precision name is DSCIPM.

Description

Routine **SCIPM** computes simultaneous confidence intervals on all $k^* = k(k-1)/2$ pairwise comparisons of k means $\mu_1, \mu_2, \ldots, \mu_k$ in the one-way analysis of variance model. Any of several methods can be chosen. A good review of these methods is given by Stoline (1981). Also the methods are discussed in many elementary statistics texts, e.g., Kirk (1982, pages 114-127).

Let s^2 (input in **S2**) be the estimated variance of a single observation. Let ν be the degrees of freedom (input in **DFS2**) associated with s^2: Let $\alpha = 1 - \text{CONPER}/100.0$. The methods are summarized as follows:

Tukey method: The Tukey method gives the narrowest simultaneous confidence intervals for all pairwise differences of means $\mu_i - \mu_j$ in balanced ($n_1 = n_2 = \ldots = n_k = n$) one-way designs. The method is exact and uses the Studentized range distribution. The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm q_{1-\alpha;k,\nu}\sqrt{\frac{s^2}{n}}$$

where $q_{1-\alpha;k,\nu}$ is the $(1-\alpha)100$ percentage point of the Studentized range distribution with parameters k and ν.

Tukey-Kramer method: The Tukey-Kramer method is an approximate extension of the Tukey method for the unbalanced case. (The method simplifies to the Tukey method for the balanced case.) The method always produces confidence intervals narrower than the Dunn-Sidak and Bonferroni methods. Hayter (1984) proved that the method is conservative, *i*.e., the method guarantees a confidence coverage of at least $(1-\alpha)100\%$. Hayter's proof gave further support to earlier recommendations for its use (Stoline 1981). (Methods that are currently better are restricted to special cases and only offer improvement in severely unbalanced cases, see, e.g., Spurrier and Isham 1985). The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm q_{1-\alpha;k,\nu}\sqrt{\frac{s^2}{2n_i} + \frac{s^2}{2n_j}}$$

Dunn-Šidák method: The Dunn-Šidák method is a conservative method. The method gives wider intervals than the Tukey-Kramer method. (For large ν and small α and k, the difference is only slight.) The method is slightly better than the Bonferroni method and is based on an improved Bonferroni (multiplicative) inequality (Miller, pages 101, 254-255). The method uses the t distribution (see IMSL routine **TIN**, in Chapter 17, "Probability Distribution Function and Inverses"). The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm t_{\frac{1}{2}+\frac{1}{2}(1-\alpha)^{1/k^*};\nu}\sqrt{\frac{s^2}{n_i} + \frac{s^2}{n_j}}$$

where $t_{f;\nu}$ is the $100f$ percentage point of the t distribution with ν degrees of freedom.

Bonferroni method: The Bonferroni method is a conservative method based on the Bonferroni (additive) inequality (Miller, page 8). The method uses the *t* distribution. The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm t_{1-\alpha/(2k^*);\nu}\sqrt{\frac{s^2}{n_i} + \frac{s^2}{n_j}}$$

Scheffé method: The Scheffé method is an overly conservative method for simultaneous confidence intervals on pairwise difference of means. The method is applicable for simultaneous confidence intervals on all contrasts, i.e., all linear combinations

$$\sum_{i=1}^{k} c_i\mu_i \text{ where } \sum_{i=1}^{k} c_i = 0$$

The method can be recommended here only if a large number of confidence intervals on contrasts in addition to the pairwise differences of means are to be constructed. The method uses the *F* distribution (see IMSL routine **FIN**, in Chapter 17, "Probability Distribution Function and Inverses". The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm \sqrt{(k-1)F_{1-\alpha;k-1,\nu}(\frac{s^2}{n_i} + \frac{s^2}{n_j})}$$

where $F_{1-\alpha;k-1,\nu}$ is the $(1-\alpha)100$ percentage point of the *F* distribution with $k-1$ and ν degrees of freedom.

One-at-a-time *t* method (Fisher's LSD): The one-at-a-time *t* method is the method appropriate for constructing a single confidence interval. The confidence percentage input is appropriate for one interval at a time. The method has been used widely in conjunction with the overall test of the null hypothesis $\mu_1 = \mu_2 = \ldots = \mu_k$ by the use of the *F* statistic. Fisher's LSD (least significant difference) test is a two-stage test that proceeds to make pair-wise comparisons of means only if the overall *F* test is significant.

Milliken and Johnson (1984, page 31) recommend LSD comparisons after a significant *F* only if the number of comparisons is small and the comparisons were planned prior to the analysis. If many unplanned comparisons are made, they recommend Scheffe's method. If the *F* test is insignificant, a few planned comparisons for differences in means can still be performed by using either Tukey, Tukey-Kramer, Dunn-Šidák or Bonferroni methods. Because the *F* test is insignificant, Scheffe's method will not yield any significant differences. The formula for the difference $\mu_i - \mu_j$ is given by

$$\bar{y}_i - \bar{y}_j \pm t_{1-\frac{\alpha}{2};\nu}\sqrt{\frac{s^2}{n_i} + \frac{s^2}{n_j}}$$

Comments

Workspace may be explicitly provided, if desired, by use of **S2IPM**/**DS2IPM**. The reference is:

```
CALL S2IPM (NGROUP, NI, YMEANS, DFS2, S2, IMETH, CONPER, IPRINT, STAT, LDSTAT,
    WK, IWK)
```

The additional arguments are as follows:

WK — Real work vector of length NGROUP.

IWK — Integer work vector of length NGROUP.

Example

Simultaneous 99% confidence intervals are computed for all pairwise comparisons of 5 means from a one-way analysis of variance design. In order to compare the results of each method, all the options for IMETH are used for input. The data are given by Kirk (1982, Table 3.5-1, page 117). In the output, pairs of means declared not equal are indicated by the letter N. The other pairs of means (for which there is insufficient evidence from the data to declare the means are unequal) are indicated by an equal sign (=).

```
      USE SCIPM_INT

      IMPLICIT   NONE
      INTEGER    LDSTAT, NGROUP
      PARAMETER  (NGROUP=5, LDSTAT=NGROUP*(NGROUP-1)/2)
!
      INTEGER    IMETH, IPRINT, NI(NGROUP)
      REAL       CONPER, DFS2, S2, STAT(LDSTAT,5), YMEANS(NGROUP)
!
      DATA YMEANS/36.7, 48.7, 43.4, 47.2, 40.3/
      DATA NI/10, 10, 10, 10, 10/
!
      DFS2   = 45.0
      S2     = 28.8
      CONPER = 99.0
      IPRINT = 1
      DO 10  IMETH=0, 4
      CALL SCIPM (NI, YMEANS, DFS2, S2, STAT, IMETH=IMETH, &
      CONPER=CONPER, IPRINT=IPRINT)
   10 CONTINUE
      END
```

Output

```
                     Simultaneous Confidence Intervals
                   for All Pairwise Differences of Means
                           (Tukey Method)

                                          99.0% Confidence Interval
                                         --------------------------

    Group I  Group J  Mean I - Mean J   Lower Limit   Upper Limit
N         1        2            -12.0       -20.261        -3.739
=         1        3             -6.7       -14.961         1.561
N         1        4            -10.5       -18.761        -2.239
=         1        5             -3.6       -11.861         4.661
=         2        3              5.3        -2.961        13.561
```

=	2	4	1.5	-6.761	9.761
N	2	5	8.4	0.139	16.661
=	3	4	-3.8	-12.061	4.461
=	3	5	3.1	-5.161	11.361
=	4	5	6.9	-1.361	15.161

Simultaneous Confidence Intervals
for All Pairwise Differences of Means
(Dunn-Sidak Method)

				99.0% Confidence Interval	
	Group I	Group J	Mean I - Mean J	Lower Limit	Upper Limit
N	1	2	-12.0	-20.445	-3.555
=	1	3	-6.7	-15.145	1.745
N	1	4	-10.5	-18.945	-2.055
=	1	5	-3.6	-12.045	4.845
=	2	3	5.3	-3.145	13.745
=	2	4	1.5	-6.945	9.945
=	2	5	8.4	-0.045	16.845
=	3	4	-3.8	-12.245	4.645
=	3	5	3.1	-5.345	11.545
=	4	5	6.9	-1.545	15.345

Simultaneous Confidence Intervals
for All Pairwise Differences of Means
(Bonferroni Method)

				99.0% Confidence Interval	
	Group I	Group J	Mean I - Mean J	Lower Limit	Upper Limit
N	1	2	-12.0	-20.449	-3.551
=	1	3	-6.7	-15.149	1.749
N	1	4	-10.5	-18.949	-2.051
=	1	5	-3.6	-12.049	4.849
=	2	3	5.3	-3.149	13.749
=	2	4	1.5	-6.949	9.949
=	2	5	8.4	-0.049	16.849
=	3	4	-3.8	-12.249	4.649
=	3	5	3.1	-5.349	11.549
=	4	5	6.9	-1.549	15.349

Simultaneous Confidence Intervals
for All Pairwise Differences of Means
(Scheffe Method)

				99.0% Confidence Interval	
	Group I	Group J	Mean I - Mean J	Lower Limit	Upper Limit
N	1	2	-12.0	-21.317	-2.683
=	1	3	-6.7	-16.017	2.617
N	1	4	-10.5	-19.817	-1.183
=	1	5	-3.6	-12.917	5.717
=	2	3	5.3	-4.017	14.617
=	2	4	1.5	-7.817	10.817
=	2	5	8.4	-0.917	17.717
=	3	4	-3.8	-13.117	5.517
=	3	5	3.1	-6.217	12.417
=	4	5	6.9	-2.417	16.217

Simultaneous Confidence Intervals
for All Pairwise Differences of Means

(One-at-a-Time t Method--LSD Test)

				99.0% Confidence Interval	
	Group I	Group J	Mean I - Mean J	Lower Limit	Upper Limit
N	1	2	-12.0	-18.455	-5.545
N	1	3	-6.7	-13.155	-0.245
N	1	4	-10.5	-16.955	-4.045
=	1	5	-3.6	-10.055	2.855
=	2	3	5.3	-1.155	11.755
=	2	4	1.5	-4.955	7.955
N	2	5	8.4	1.945	14.855
=	3	4	-3.8	-10.255	2.655
=	3	5	3.1	-3.355	9.555
N	4	5	6.9	0.445	13.355

SNKMC

Performs Student-Newman-Keuls multiple comparison test.

Required Arguments

YMEANS — Vector of length NGROUP containing the means. (Input)

SEMEAN — Effective estimated standard error of a mean. (Input)
In fixed effects models, SEMEAN equals the estimated standard error of a mean. For example, in a one-way model

$$\text{SEMEAN} = \sqrt{s^2 / n}$$

where s^2 is the estimate of σ^2 and n is the number of responses in a sample mean. In models with random components, use

$$\text{SEMEAN} = \text{SEDIF} / \sqrt{2}$$

where SEDIF is the estimated standard error of the difference of two means.

DFSE — Degrees of freedom associated with SEMEAN. (Input)

ALPHA — Significance level of test. (Input)
ALPHA must be in the interval [0.01, 0.10].

IEQMNS — Vector of length NGROUP - 1 indicating the size of groups of means declared to be equal. (Output)
IEQMNS(I) = J indicates the I-th smallest mean and the next J - 1 larger means are declared equal. IEQMNS(I) = 0 indicates no group of means starts with the I-th smallest mean.

Optional Arguments

NGROUP — Number of groups under consideration. (Input)
Default: NGROUP = size (YMEANS,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

FORTRAN 90 Interface

Generic: CALL SNKMC (YMEANS, SEMEAN, DFSE, ALPHA, IEQMNS [, ...])
Specific: The specific interface names are S_SNKMC and D_SNKMC.

FORTRAN 77 Interface

Single: CALL SNKMC (NGROUP, YMEANS, SEMEAN, DFSE, ALPHA, IPRINT, IEQMNS)
Double: The double precision name is DSNKMC.

Description

Routine SNKMC performs a multiple comparison analysis of means using the Student-Newman-Keuls method. The null hypothesis is equality of all possible ordered subsets of a set of means. This null hypothesis is tested using the studentized range for each of the corresponding subsets of sample means. The method is discussed in many elementary statistics texts, e.g., Kirk (1982, pages 123-125).

Comments

Workspace may be explicitly provided, if desired, by use of S2KMC/DS2KMC. The reference is:

```
CALL S2KMC (NGROUP, YMEANS, SEMEAN, DFSE, ALPHA, IPRINT, IEQMNS, WK, IWK)
```

The additional arguments are as follows:

WK — Vector of length NGROUP containing YMEANS in ascending order. (Output)

IWK — Work vector of length 2 * NGROUP.

Example

A multiple comparisons analysis is performed using data discussed by Kirk (1982, pages 123-125). In the output, means that are not connected by a common underline are declared different.

```
      USE UMACH_INT
      USE SNKMC_INT

      IMPLICIT   NONE
```

```
      INTEGER    IEQMNS(4), IPRINT, N, NOUT
      REAL       ALPHA, DFSE, S2, SEMEAN, SQRT, YMEANS(5)
      INTRINSIC  SQRT
!
      DATA YMEANS/36.7, 48.7, 43.4, 47.2, 40.3/
!
      CALL UMACH (2, NOUT)
      S2     = 28.8
      N      = 10
      SEMEAN = SQRT(S2/N)
      DFSE   = 45.0
      ALPHA  = .01
      IPRINT = 1
      CALL SNKMC (YMEANS, SEMEAN, DFSE, ALPHA, IEQMNS, IPRINT=IPRINT)
      WRITE (NOUT,99999) IEQMNS
99999 FORMAT (' IEQMNS = ', 4I3)
      END
```

Output

```
 Group          1          5          3          4          2
 Mean       36.70      40.30      43.40      47.20      48.70

       AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
                  BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
                              CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

 IEQMNS =   3  3  3  0
```

CIDMS

Computes a confidence interval on a variance component estimated as proportional to the difference in two mean squares in a balanced complete experimental design.

Required Arguments

DF1 — Degrees of freedom for effect 1. (Input)

EFMS1 — Mean square for effect 1. (Input)

DF2 — Degrees of freedom for effect 2. (Input)

EFMS2 — Mean square for effect 2. (Input)

VCHAT — Estimated variance component. (Input)
VCHAT = (EFMS1 - EFMS2)/a, where a is some positive constant.

CONINT — Vector of length 2 containing the lower and upper endpoints of the confidence interval, respectively. (Output)

Optional Arguments

CONPER — Confidence level for two-sided interval estimate on the variance component, in percent. (Input)
Default: CONPER = 95.0.
A CONPER percent interval is computed, hence, CONPER must be in the interval [0.0, 100.0). CONPER often will be 90.0, 95.0, or 99.0. For a one-sided interval with confidence level ONECL, ONECL in the interval [50.0, 100.0), set
CONPER = 100.0 - 2.0 * (100.0 - ONECL).

IMETH — Method option. (Input)
Default: IMETH = 0.

IOPT	Action
0	Graybill's Method
1	Bross' Method

FORTRAN 90 Interface

Generic: CALL CIDMS (DF1, EFMS1, DF2, EFMS2, VCHAT, CONINT [, ...])
Specific: The specific interface names are S_CIDMS and D_CIDMS.

FORTRAN 77 Interface

Single: CALL CIDMS (DF1, EFMS1, DF2, EFMS2, VCHAT, CONPER, IMETH, CONINT)
Double: The double precision name is DCIDMS.

Description

Routine CIDMS computes a confidence interval on a variance component that has been estimated as proportional to the difference of two mean squares. Let

$$\hat{\gamma}_1^2 \text{ and } \hat{\gamma}_2^2$$

(stored in EFMS1 and EFMS2, respectively) be the two mean squares. The variance component estimate

$$\hat{\sigma}^2$$

(stored in VCHAT) is assumed to be of the form

$$\hat{\sigma}^2 = \frac{\hat{\gamma}_1^2 - \hat{\gamma}_2^2}{a}$$

where *a* is some positive constant. Two methods for computing a confidence interval on σ^2 can be used. For IMETH = 0, the method discussed by Graybill (1976, Theorem 15.3.5, page 624, and Note 4, page 620) is used. The result was proposed by Williams (1962). For IMETH = 1, the method due to Bross (1950) and discussed by Anderson and Bancroft (1952, page 322) is used.

Routine CIDMS can also be used when a variance component is estimated by the difference of two linear combinations of mean squares, each linear combination contains nonnegative coefficients, and the two linear combinations do not use any of the same mean squares. Let

$$\sum_{i=1}^{k} c_i \hat{\gamma}_i^2 \text{ and } \sum_{i=1}^{k} d_i \hat{\gamma}_i^2$$

be the two linear combinations (stored in **EFMS1** and **EFMS2**, respectively). The variance component estimate

$$\hat{\sigma}^2$$

(stored in **VCHAT**) is assumed to be of the form

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{k} c_i \hat{\gamma}_i^2 - \sum_{i=1}^{k} d_i \hat{\gamma}_i^2}{a}$$

where a is some positive constant, the c_i's and d_i's are nonnegative, and for $i = 1, 2, \ldots, k$, $c_i d_i = 0$. Satterthwaite (1946) approximations as discussed by Graybill (1976, pages 642-643) can be used to arrive at approximate degrees of freedom for each linear combination of mean squares for input into **CIDMS**. Let ν_i be the degrees of freedom associated with the i-th mean square

$$\hat{\gamma}_i^2$$

The degrees of freedom stored in **DF1** and **DF2** should be taken to be

$$\frac{\left(\sum_{i=1}^{k} c_i \hat{\gamma}_i^2\right)^2}{\sum_{i=1}^{k} c_i^2 (\hat{\gamma}_i^2)^2 / \nu_i}$$

and

$$\frac{\left(\sum_{i=1}^{k} d_i \hat{\gamma}_i^2\right)^2}{\sum_{i=1}^{k} d_i^2 (\hat{\gamma}_i^2)^2 / \nu_i}$$

respectively.

Comments

Informational error

Type	Code	Description
1	1	One or more endpoints of CI are set to zero.

Example

This example computes a confidence interval on a variance component estimated by a difference of mean squares using a nested design discussed by Graybill (1976, pages 635-636). The nested design gave the following analysis of variance table:

A	5	385.4	$\gamma_1^2 = \sigma^2 + 3\sigma_B^2 + 12\sigma_A^2$
B within A	18	85.4	$\gamma_2^2 = \sigma^2 + 3\sigma_B^2$
Error	48	12.3	$\gamma_3^2 = \sigma^2$

A confidence interval of

$$\sigma_A^2$$

is computed using the method of Graybill. (Note that the lower endpoint of the confidence interval, which is 3.136, is given incorrectly by Graybill [page 636]. Graybill uses an incorrect value for $F_{0.975;5,18}$ in his computations.)

```
      USE CIDMS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       CONINT(2), DF1, DF2, EFMS1, EFMS2, VCHAT
!
      DF1    = 5.0
      EFMS1  = 385.4
      DF2    = 18.0
      EFMS2  = 85.4
      VCHAT  = (EFMS1-EFMS2)/12.0
!
      CALL CIDMS (DF1, EFMS1, DF2, EFMS2, VCHAT, CONINT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) CONINT
99999 FORMAT (' Lower confidence limit', F9.3, /' Upper confidence ', &
            'limit', F9.3)
      END
```

Output

```
Lower confidence limit    3.136
Upper confidence limit  186.464
```

ROREX

Reorders the responses from a balanced complete experimental design.

Required Arguments

NL — Vector of length NF containing the number of levels for each of the NF factors. (Input)
NL(I) is the number of levels for the I-th factor.

IORD — Vector of length NF indicating the ordering of the responses in vector YIN. (Input)
IORD(I) = J means the model subscript corresponding to factor I is altering J-th most rapidly.

YIN — Vector of length NL(1) * NL(2) * ... * NL(NF) containing the responses in the order specified by IORD. (Input)

JORD — Vector of length NF indicating the new ordering of the responses in vector YOUT. (Input)
JORD (K) = L means the model subscript corresponding to factor K is altering L-th most rapidly.

YOUT — Vector of length NL(1) * NL(2) * ... * NL(NF) containing the responses in the order specified by JORD. (Output)

Optional Arguments

NF — Number of factors (number of subscripts) in the model, including error. (Input)
Default: NF = size (NL,1).

FORTRAN 90 Interface

Generic: CALL ROREX (NL, IORD, YIN, JORD, YOUT [, ...])
Specific: The specific interface names are S_ROREX and D_ROREX.

FORTRAN 77 Interface

Single: CALL ROREX (NF, NL, IORD, YIN, JORD, YOUT)
Double: The double precision name is DROREX.

Description

Typically, responses from a balanced complete experimental design are stored in a pattern that takes advantage of the design structure, consequently, the full set of model subscripts is not needed to identify each response. Routine **ROREX** assumes the usual pattern, which requires that one model subscript changes most rapidly, another changes next most rapidly, and so on, throughout the input data vector **YIN**. In many programs, including IMSL programs for this kind of data, the computations and ordering of output are dependent on which subscripts are moving most rapidly relative to others, within the pattern, in the input data. Data may be available in a form that needs reordering within the pattern before entry to an analysis routine. Routine **ROREX** reorders data in **YIN**, as controlled by **JORD**, and returns the reordered data in **YOUT**.

Let k (stored in **NF**) be the number of factors, and for $j = 1, 2, \ldots, k$, let n_j (stored as the j-th element of **NL**) be the number of levels in the j-th factor. Let the data in **YIN** be denoted by

$$y_{i_1 i_2, \ldots, i_k}$$

where for $j = 1, 2, \ldots, k$, $i_j = 1, 2, \ldots, n_j$. For every response in **YIN**, let p_r denote the model subscript i_j that is altering r-th most rapidly for r and j in the set $\{1, 2, \ldots, k\}$ For every response in **YOUT**, let q_s have a similar definition. Let P_r and Q_s equal the number of levels for the factor whose model subscript is altering r-th and s-th most rapidly in **YIN** and **YOUT**, respectively.

The m-th element of **YIN**, denoted by

$$y_{p_1 p_2 \ldots p_k}$$

with

$$m = p_1 + \sum_{u=2}^{k} (p_u - 1) \prod_{v=1}^{u-1} p_v$$

can be found using $p1$ given by

$$m_1 = m$$

$$p_1 = \begin{cases} m_1 \text{ modulo } P_1 & \text{if } m_1 \text{ modulo } P_1 \text{ is nonzero} \\ P_1 & \text{if } m_1 \text{ modulo } P_1 \text{ is zero} \end{cases}$$

and then for $r = 2, 3, \ldots, k$, p_r given by

$$m_r = \frac{m_{r-1} - p_{r-1}}{P_{r-1}} + 1,$$

$$p_r = \begin{cases} m_r \text{ modulo } P_r & \text{if } m_r \text{ modulo } P_r \text{ is nonzero} \\ P_r & \text{if } m_r \text{ modulo } P_r \text{ is zero} \end{cases}$$

The *m*-th element of **YOUT**, denoted by

$$y_{q_1 q_2 \ldots q_k}$$

is given by replacing the *p*'s by *q*'s in the formulas in the preceding equations.

Comments

Workspace may be explicitly provided, if desired, by use of **R2REX**/**DR2REX**. The reference is:

```
CALL R2REX (NF, NL, IORD, YIN, JORD, YOUT, IWK)
```

The additional argument is:

IWK — Work vector of length 4 * NF.

Example

The input responses y_{ijk} are ordered in **YIN** so that the subscript *i* varies most rapidly, *j* the next most rapidly, and *k* the least rapidly. Routine **ROREX** is used to reorder the responses into standard order, i.e., with the subscript *i* varying least rapidly, *j* the next most rapidly, and *k* the most rapidly.

```
          USE ROREX_INT
          USE WRRRL_INT

          IMPLICIT   NONE
          INTEGER    NF, NOBS
          PARAMETER  (NF=3, NOBS=24)
    !
          INTEGER    IORD(NF), JORD(NF), NL(NF)
          REAL       YIN(NOBS), YOUT(NOBS)
          CHARACTER  CLABEL(1)*6, RLABEL(1)*4
          DATA       CLABEL/'NUMBER'/, RLABEL/'NONE'/
    !
          DATA NL/2, 3, 4/
          DATA IORD/1, 2, 3/
          DATA JORD/3, 2, 1/
          DATA YIN/1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, &
              11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, &
              21.0, 22.0, 23.0, 24.0/
    !
          CALL ROREX (NL, IORD, YIN, JORD, YOUT)
    !
          CALL WRRRL ('YOUT', YOUT, RLABEL, CLABEL, 1, NOBS, 1, 0, '(F4.1)')
```

```
END
```

Output

```
                                    YOUT
     1     2     3     4     5     6     7     8     9    10    11    12    13
   1.0   7.0  13.0  19.0   3.0   9.0  15.0  21.0   5.0  11.0  17.0  23.0   2.0

    14    15    16    17    18    19    20    21    22    23    24
   8.0  14.0  20.0   4.0  10.0  16.0  22.0   6.0  12.0  18.0  24.0
```

Categorical and Discrete Data Analysis

Routines

Usage Notes

Routines for modeling and analyzing a two- or higher-dimensional contingency table are described in this chapter. Also included are routines for modeling responses from some discrete distributions when discrete or continuous covariates are measured.

The Basic Data Structures

The most common of the three data structures used by the routines in this chapter is a multidimensional (or multi-way) contingency table input as a real vector with length equal to the product of the number of categories for each dimension. This structure may be obtained from a data matrix **X** via the routine **FREQ** in Chapter 1, "Basic Statistics". Alternatively, multi-way tables may be created and input directly by the user. The multi-way structure is used by all of the log-linear modeling routines (**PRPFT**, **CTLLN**, **CTPAR**, **CTASC**, and **CTSTP**), and is also used in the randomization tests routine, **CTRAN**.

A second data structure used by the categorical generalized linear models routine, **CTGLM**, is the data matrix **X**. In **CTGLM** (and elsewhere), if **X** has many identical rows, at least on the variables of interest, consider using *Chapter 1* routine **CSTAT** to add a frequency variable to a reduced matrix **X**. The transposed output from this routine can replace **X** as input to **CTGLM**, and **CTGLM** will perform its computations faster (with a linear speed up) on the reduced matrix.

Finally, two-way tables are input into routines **CTCHI**, **CTTWO**, **CTPRB**, **CTEPR**, and **CTWLS** as two-dimensional real arrays. As with the multidimensional arrays, two-dimensional arrays may be created via *Chapter 1* routine **FREQ**, in which case the leading dimension must equal the number of categories for the first dimension in the table, or they can be created and input directly by the user. Alternatively, the routine **TWFRQ** from *Chapter 1* may be used to obtain the two-way frequency table.

Types of Analysis

Routines **CTCHI** ($r \times c$) and **CTTWO** (2×2) (see Chapter 1, "Basic Statistics") compute many statistics of interest in a two-way table. Statistics computed by these routines include the usual chi-squared statistics, measures of association, Kappa, and many others. Asymptotic statistics for a two-way table that are not computed by either **CTCHI** or **CTTWO** can probably be computed by routines **CTRAN** or **CTWLS**, but note that these latter two routines require more setup since they require that the user indicate how the statistics are to be computed. Exact probabilities for two-way tables can be computed by **CTPRB**, but this routine uses the total enumeration algorithm and, thus, often uses orders of magnitude more computer time that **CTEPR**, which computes the same probabilities by use of the network algorithm (but can still be quite expensive).

The routines in the second section are all concerned with hierarchical log-linear models (see, e.g., Bishop, Fienberg, and Holland 1975). The routines in Chapter 1, "Basic Statistics" will often be used to obtain the multi-dimensional tables input into these routines, or the table will be input directly by the user. If the hierarchical is not known, routine **CTASC** will often be the first routine considered. The partial association statistics computed by this routine can be used to obtain a rough estimate of the model to be used. This rough model can then be refined through the use of **CTSTP**, which does stepwise model building. Of course, both of these routines are subject to the usual problems associated with building models once the data have been collected: the resulting models may not be correct.

Once a model has been selected (provisional or otherwise), routine **CTLLN** can be used to compute and print many model statistics (parameter estimates, residuals, goodness of fit tests, etc.). If only the parameter estimates and associated variance/covariance matrix are needed, **CTPAR** can be used instead. Both of these routines can compute estimates when sampling and/or structural zeros (cells in the table with observed or restricted counts of zero, respectively) are present in the table, as can all routines in this section.

The algorithm underlying all of the routines in the second section is the iterative proportional fitting algorithm, which is implemented in routine **PRPFT**. When structural or sampling zeros are present in the table, this algorithm can be quite slow to converge. Also, only the expected cell counts are returned by **PRPFT**, it can be quite difficult to determine degrees of freedom when structural zeros are present in the data. Because a structural zero is a restriction on the parameter space, 1 degree of freedom must be subtracted for each structural zero in the multiway table. The difficulty is in determining where the subtraction should occur. All routines in this section use a Cholesky factorization of $X^T X$ where X is the "design matrix." This is used to determine which effects should lose degrees of freedom because of structural zeros. Sampling zeros, although they can lead to infinite parameter estimates, do not subtract from the total degrees of freedom. See Clarkson and Jennrich (1991), or Baker, Clarke, and Lane (1985) for details.

Routine **CTRAN** computes generalized Mantel-Haenszel statistics in stratified $r \times c$ tables. Generalized Mantel-Haenszel statistics assume that the "direction" of departure from the null hypothesis is consistent from one table to the next. Under this assumption, statistics computed for each table are pooled across all strata yielding a more powerful test than could be obtained otherwise. The statistics computed include measures of correlation, location, and independence using user selected row and/or column scores. Details can be found in (Koch, Amara, and Atkinson 1983) or in the "Algorithm" section for **CTRAN**.

The routine **CTGLM** in the fourth section is concerned with generalized linear models (see McCullagh and Nelder 1983) in discrete data. This routine may be used to compute estimates and associated statistics in probit, logistic, minimum extreme value, Poisson, negative binomial (with known number of successes), and logarithmic models. Classification variables as well as weights, frequencies and additive constants may be used so that quite general linear models can be fit. Residuals, a measure of influence, the coefficient estimates, and other statistics are returned for each model fit. When infinite parameter estimates are required, extended maximum likelihood estimation may be used. Log-linear models may be fit in **CTGLM** through the use of Poisson regression models.

Results from Poisson regression models involving structural and sampling zeros will be identical to the results obtained from the log-linear model routines but will be fit by a quasi-Newton algorithm rather than through iterative proportional fitting.

The weighted least-squares analysis of Grizzle, Starmer, and Koch (1969) is implemented in routine **CTWLS**. In this routine, the user first transforms the observed probability estimates (in predefined ways) and then fits a linear model to the transformed estimates using generalized least squares. Multivariate hypotheses associated with the coefficient estimates for the linear model fit may then be tested. In this way, many statistics of interest such as generalized Kappa statistics and parameter estimates in logistic models may be estimated. Of course, the logistic models fit by **CTWLS** use a generalized least-squares criterion rather than the maximum likelihood criterion used to compute the logistic model estimates in **CTGLM**. The generalized least-squares estimates will generally differ somewhat from estimates computed via maximum likelihood.

Other Routines

The routines in Chapter 1, "Basic Statistics" may be used to create the data structures discussed above. These routines can also create one-dimensional frequency tables, which may then be used by routine **CHIGF** (Chapter 7, "Tests of Goodness of Fit and Randomness") to compute chi-squared goodness-of-fit test statistics or with routines **VHSTP** or **HHSTP** (see Chapter 16, "Line Printer Graphics") to prepare histograms. Routines **CTRHO**, **TETCC** , **BSCAT**, and **BSPBS** (see Chapter 3, "Correlation") may be used to compute some measures of correlation in two-way contingency tables.

CTTWO

Performs a chi-squared analysis of a 2 by 2 contingency table.

Required Arguments

TABLE — 2 by 2 matrix containing the observed counts in the contingency table. (Input)

EXPECT — 3 by 3 matrix containing the expected values of each cell in TABLE under the null hypothesis of independence, in the first 2 rows and 2 columns, and the marginal totals in the last row and column. (Output)

CHICTR — 3 by 3 matrix containing the contributions to chi-squared for each cell in TABLE in the first 2 rows and 2 columns. (Output)
The last row and column contain the total contribution to chi-squared for that row or column.

CHISQ — Vector of length 15 containing statistics associated with this contingency table. (Output)

I	CHISQ(I)
1	Pearson chi-squared statistic
2	Probability of a larger Pearson chi-squared
3	Degrees of freedom for chi-squared
4	Likelihood ratio G^2 (chi-squared)
5	Probability of a larger G^2
6	Yates corrected chi-squared
7	Probability of a larger corrected chi-squared
8	Fisher's exact test (one tail)
9	Fisher's exact test (two tail)
10	Exact mean
11	Exact standard deviation

The following statistics are based upon the chi-squared statistic CHISQ(1)

I	CHISQ(I)
12	Phi (Φ)
13	The maximum possible Φ

I	CHISQ(I)
14	Contingency coefficient *P*
15	The maximum possible contingency coefficient

STAT — 24 by 5 matrix containing statistics associated with this table. (Output)
Each row of the matrix corresponds to a statistic.

Row	Statistic
1	Gamma
2	Kendall's τ_b
3	Stuart's τ_c
4	Somers' *D* (row)
5	Somers' *D* (column)
6	Product moment correlation
7	Spearman rank correlation
8	Goodman and Kruskal τ (row)
9	Goodman and Kruskal τ (column)
10	Uncertainty coefficient *U* (normed)
11	Uncertainty $U_{r\|c}$ (row)
12	Uncertainty $U_{c\|r}$ (column)
13	Optimal prediction λ (symmetric)
14	Optimal prediction $\lambda_{r\|c}$ (row)
15	Optimal prediction $\lambda_{c\|r}$ (column)
16	Optimal prediction $\lambda^*_{r\|c}$ (row)
17	Optimal prediction $\lambda^*_{c\|r}$ (column)
18	Yule's *Q*
19	Yule's *Y*
20	Crossproduct ratio
21	Log of crossproduct ratio
22	Test for linear trend
23	Kappa
24	McNemar test of symmetry

If a statistic is not computed, its value is reported as NaN (not a number). The columns are as follows:

Column	Statistic
1	Estimated statistic
2	Its estimated standard error for any parameter value
3	Its estimated standard error under the null hypothesis
4	*z*-score for testing the null hypothesis
5	*p*-value for the test in column 4

In the McNemar test, column 1 contains the statistic, column 2 contains the chi-squared degrees of freedom, column 4 contains the exact *p*-value, and column 5 contains the chi-squared asymptotic *p*-value.

Optional Arguments

LDTABL — Leading dimension of `TABLE` exactly as specified in the dimension statement of the calling program. (Input)
Default: `LDTABL` = size (`TABLE`,1).

ICMPT — Computing option. (Input)
If `ICMPT` = 0, all of the values in `CHISQ` and `STAT` are computed. `ICMPT` = 1 means compute only the first 11 values of `CHISQ`, and no values of `STAT` are computed.
Default: `ICMPT` = 0.

IPRINT — Printing option. (Input)
`IPRINT` = 0 means no printing is performed. If `IPRINT` = 1, printing is performed.
Default: `IPRINT` = 0.

LDEXPE — Leading dimension of `EXPECT` exactly as specified in the dimension statement of the calling program. (Input)
Default: `LDEXPE` = size (`EXPECT`,1).

LDCHIC — Leading dimension of `CHI` exactly as specified in the dimension statement of the calling program. (Input)
Default: `LDCHI` = size (`CHI`,1).

LDSTAT — Leading dimension of `STAT` exactly as specified in the dimension statement of the calling program. (Input)
Default: `LDSTAT` = size (`STAT`,1).

FORTRAN 90 Interface

Generic: `CALL CTTWO (TABLE, EXPECT, CHICTR, CHISQ, STAT [, ...])`

Specific: The specific interface names are `S_CTTWO` and `D_CTTWO`.

FORTRAN 77 Interface

Single: `CALL CTTWO (TABLE, LDTABL, ICMPT, IPRINT, EXPECT, LDEXPE, CHICTR, LDCHIC, CHISQ, STAT, LDSTAT)`

Double: The double precision name is `DCTTWO`.

Description

Routine `CTTWO` computes statistics associated with 2 × 2 contingency tables. Always computed are chi-squared tests of independence, expected values based upon the independence assumption, contributions to chi-squared in a test of independence, and row and column marginal totals. Optionally, when `ICMPT` = 0, `CTTWO` can compute some measures of association, correlation, prediction, uncertainty, the McNemar test for symmetry, a test for linear trend, the odds and the log odds ratio, and the Kappa statistic.

Other IMSL routines that may be of interest include `TETCC` in Chapter 3, "Correlation" (for computing the tetrachoric correlation coefficient) and `CTCHI` in this chapter (for computing statistics in other than 2 × 2 contingency tables).

Notation

Let x_{ij} denote the observed cell frequency in the *ij* cell of the table and *n* denote the total count in the table. Let $p_{ij} = p_{i\bullet}p_{\bullet j}$ denote the predicted cell probabilities (under the null hypothesis of independence) where $p_{i\bullet}$ and $p_{\bullet j}$ are the row and column relative marginal frequencies, respectively. Next, compute the expected cell counts as $e_{ij} = n\, p_{ij}$.

Also required in the following are a_{uv} and b_{uv}, $u, v = 1, \ldots, n$. Let (r_s, c_s) denote the row and column response of observation *s*. Then, $a_{uv} = 1$, 0, or -1, depending upon whether $r_u < r_v$, $r_u = r_v$, or $r_u > r_v$, respectively. The b_{uv} are similarly defined in terms of the c_s's.

The Chi-squared Statistics

For each cell of the four cells in the table, the contribution to chi-squared is given as $(x_{ij} - e_{ij})^2/e_{ij}$. The Pearson chi-squared statistic (denoted as $\mathbf{X}^2$) is computed as the sum of the cell contributions to chi-squared. It has, of course, 1 degree of freedom and tests the null hypothesis of independence, i.e., of $H_0 : p_{ij} = p_{i\bullet}p_{\bullet j}$. Reject the null hypothesis if the computed value of $\mathbf{X}^2$ is too large.

Compute G^2, the maximum likelihood equivalent of $\mathbf{X}^2$, as

$$-2.0\sum_{i,j} x_{ij}\ln\left(x_{ij}/np_{ij}\right)$$

G^2 is asymptotically equivalent to $\mathbf{X}^2$ and tests the same hypothesis with the same degrees of freedom.

Measures Related to Chi-squared (Phi and the Contingency Coefficient)

Two measures related to chi-squared but which do not depend upon sample size are phi,

$$\phi = \sqrt{\chi^2/n}$$

and the contingency coefficient,

$$P = \sqrt{\chi^2/\left(n+\chi^2\right)}$$

Since these statistics do not depend upon sample size and are large when the hypothesis of independence is rejected, they may be thought of as measures of association and may be compared across tables with different sized samples. While P has a range between 0.0 and 1.0 for any given table, the upper bound of P is actually somewhat less than 1.0 (see Kendall and Stuart 1979, page 577). In order to understand association within a table, consider also the maximum possible P(**CHISQ**(15)) and the maximum possible ϕ (**CHISQ**(13)). The significance of both statistics is the same as that of the $\mathbf{X}^2$ statistic, **CHISQ**(1).

The distribution of the $\mathbf{X}^2$ statistic in finite samples approximates a chi-squared distribution. To compute the expected mean and standard deviation of the $\mathbf{X}^2$ statistic, Haldane (1939) uses the multinomial distribution with fixed table marginals. The exact mean and standard deviation generally differ little from the mean and standard deviation of the associated chi-squared distribution.

Fisher's exact test

Fisher's exact test is a conservative but uniformly most powerful unbiased test of equal row (or column) cell probabilities in the 2 × 2 table. In this test, the row and column marginals are assumed fixed, and the hypergeometric distribution is used to obtain the significance level of the test. A one- or a two-sided test is possible. See Kendall and Stuart (1979, page 582) for a discussion.

Standard Errors and p-values for Some Measures of Association

In rows 1 through 7 of **STAT**, estimated standard errors and asymptotic *p*-values are reported. Routine **CTTWO** computes these standard errors in two ways. The first estimate, in column 2 of matrix **STAT**, is asymptotically valid for any value of the statistic. The second estimate, in column 3 of **STAT**, is only correct under the null hypothesis of no association. The *z*-scores in column 4 are computed using this second estimate of the standard errors, and the *p*-values in column 5 are computed from these *z*-scores. See Brown and Benedetti (1977) for a discussion and formulas for the standard errors in column 3.

Measures of Association for Ranked Rows and Columns

The measures of association ϕ and *P* do not require any ordering of the row and column categories. Routine **CTTWO** also computes several measures of association for tables in which the rows and column categories correspond to ranked observations. Two of these measures, the product-moment correlation and the Spearman correlation, are correlation coefficients that are computed using assigned scores for the row and column categories. In the product-moment correlation, this score is the cell index, while in the Spearman rank correlation, this score is the average of the tied ranks of the row or column marginals. Other scores are possible.

Other measures of associations, Gamma, Kendall's τ_b, Stuart's τ_c and Somers' *D*, are also computed similarly to a correlation coefficient in that the numerator in these statistics in some sense is a "covariance." In fact, these measures differ only in their denominators, their numerators being the "covariance" between the a_{uv}'s and the b_{uv}'s defined earlier. The numerator is computed as

$$\sum_u \sum_v a_{uv} b_{uv}$$

Since the product $a_{uv}b_{uv} = 1$ if both a_{uv} and b_{uv} are 1 or -1, it is easy to show that the "covariance" is twice the total number of agreements minus the number disagreements between the row and column variables where a disagreement occurs when $a_{uv}b_{uv} = -1$.

Kendall's τ_b is computed as the correlation between the a_{uv}'s and the b_{uv}'s (see Kendall and Stuart 1979, page 583). Stuart suggested a modification to the denominator of τ in which the denominator becomes the largest possible value of the "covariance." This value turns out to be approximately $2n^2$ in 2×2 tables, and this is the value used in the denominator of Stuart's τ_c. For large n, $\tau_c \approx 2\ \tau_b$.

Gamma can be motivated in a slightly different manner. Because the "covariance" of the a_{uv}'s and the b_{uv}'s can be thought of as two times the number of agreements minus the number of disagreements [$2(A - D)$, where A is the number of agreements and D is the number of disagreements], gamma is motivated as the probability of agreement minus the probability of disagreement, given that either agreement or disagreement occurred. This is just $(A - D)/(A + D)$.

Two definitions of Somers' D are possible, one for rows and a second for columns. Somers' D for rows can be thought of as the regression coefficient for predicting a_{uv} from b_{uv}. Moreover, Somers' D for rows is the probability of agreement minus the probability of disagreement, given that the column variable, b_{uv}, is not zero. Somers' D for columns is defined in a similar manner.

A discussion of all of the measures of association in this section can be found in Kendall and Stuart (1979, starting on page 592).

The crossproduct ratio is also sometimes thought of as a measure of association (see Bishop, Feinberg and Holland 1975, page 14). It is computed as:

$$\frac{p_{11} \cdot p_{22}}{p_{12} \cdot p_{21}}$$

The log of the crossproduct ratio is the log of this quantity.

The Yule's Q and Yule's Y are related to the cross product ratio. They are computed as:

$$Q = \frac{p_{11} \cdot p_{22} - p_{12} \cdot p_{21}}{p_{11} \cdot p_{22} + p_{12} \cdot p_{21}}$$

$$Y = \frac{\sqrt{p_{11} \cdot p_{22}} - \sqrt{p_{12} \cdot p_{21}}}{\sqrt{p_{11} \cdot p_{22}} + \sqrt{p_{12} \cdot p_{21}}}$$

Measures of Prediction and Uncertainty

The Optimal Prediction Coefficients

The measures in this section do not require any ordering of the row or column variables. They are based entirely upon probabilities. Most are discussed in Bishop, Feinberg, and Holland (1975, page 385).

Consider predicting or classifying the column variable for a given value of the row variable. The best classification for each row under the null hypothesis of independence is the column that has the highest marginal probability (and thus the highest probability for the row under the independence assumption). The probability of misclassification is then one minus this marginal probability. On the other hand, if independence is not assumed so that the row and columns variables are dependent, then within each row one would classify the column variables according to the category with the highest row conditional probability. The probability of misclassification for the row is then one minus this conditional probability.

Define the optimal prediction coefficient $\lambda_{c|r}$ for predicting columns from rows as the proportion of the probability of misclassification that is eliminated because the random variables are not independent. It is estimated by:

$$\lambda_{c|r} = \frac{\left(1 - p_{\bullet m}\right)\left(1 - \sum_i p_{im}\right)}{1 - p_{\bullet m}}$$

where m is the index of the maximum estimated probability in the row (p_{im}) or row margin ($p_{\bullet m}$). A similar coefficient is defined for predicting the rows from the columns. The symmetric version of the optimal prediction λ is obtained by summing the numerators and denominators of $\lambda_{r|c}$ and $\lambda_{c|r}$ and dividing. Standard errors for these coefficients are given in Bishop, Feinberg, and Holland (1975, page 388).

A problem with the optimal prediction coefficients λ is that they vary with the marginal probabilities. One way to correct for this is to use row conditional probabilities. The optimal prediction λ^* coefficients are defined as the corresponding λ coefficients in which one first adjusts the row (or column) marginals to the same number of observations. This yields

$$\lambda^*_{c|r} = \frac{\sum_i \max_j p_{j|i} - \max_j(\sum_i p_{j|i})}{R - \max_j \sum_i p_{j|i}}$$

where i indexes the rows and j indexes the columns, and $p_{j|i}$ is the (estimated) probability of column j given row i.

$$\lambda^*_{r|c}$$

is similarly defined.

Goodman and Kruskal τ

A second kind of prediction measure attempts to explain the proportion of the explained variation of the row (column) measure given the column (row) measure. Define the total variation in the rows to be

$$n/2 - (\sum_i x^2_{i\bullet})/(2n)$$

This is $1/(2n)$ times the sums of squares of the a_{uv}'s.

With this definition of variation, the Goodman and Kruskal τ coefficient for rows is computed as the reduction of the total variation for rows accounted for by the columns divided by the total variation for the rows. To compute the reduction in the total variation of the rows accounted for by the columns, define the total variation for the rows within column *j* as

$$q_j = x_{\bullet j}/2 - \left(\sum_i x_{ij}^2\right) / \left(2x_{i\bullet}\right)$$

Define the total variation for rows within columns as the sum of the q_j's. Consistent with the usual methods in the analysis of variance, the reduction in the total variation is the difference between the total variation for rows and the total variation for rows within the columns.

Goodman and Kruskal's τ columns is similarly defined. See Bishop, Feinberg, and Holland (1975, page 391) for the standard errors.

The Uncertainty Coefficients

The uncertainty coefficient for rows is the increase in the log-likelihood that is achieved by the most general model over the independence model divided by the marginal log-likelihood for the rows. This is given by

$$U_{r|c} = \frac{\Sigma_{i,j} x_{ij} \log\left(x_{i\bullet} x_{\bullet j} / \left(n x_{ij}\right)\right)}{\sum_i x_{i\bullet} \log\left(x_{i\bullet}/n\right)}$$

The uncertainty coefficient for columns is similarly defined. The symmetric uncertainty coefficient contains the same numerator as $U_{r|c}$ and $U_{c|r}$ but averages the denominators of these two statistics. Standard errors for *U* are given in Brown (1983).

Kruskal-Wallis

The Kruskal-Wallis statistic for rows is a one-way analysis-of-variance-type test that assumes that the column variable is monotonically ordered. It tests the null hypothesis that the row populations are identical, using average ranks for the column variable. This amounts to a test of $H_0 : p_{1\bullet} = p_{2\bullet}$. The Kruskal-Wallis statistic for columns is similarly defined. Conover (1980) discusses the Kruskal-Wallis test.

Test for Linear Trend

The test for a linear trend in the column probabilities assumes that the row variable is monotonically ordered. In this test, the probability for column 1 is predicted by the row index using weighted simple linear regression. The slope is given by

$$\hat{\beta} = \frac{\sum_j x_{\bullet j}\left(x_{1j}/x_{\bullet j} - x_{1\bullet}/n\right)\left(j - \bar{j}\right)}{\sum_j x_{\bullet j}\left(j - \bar{j}\right)^2}$$

where

$$\bar{j} = \sum_j x_{\bullet j} j / n$$

is the average row index. An asymptotic test that the slope is zero may be obtained as the usual large sample regression test of zero slope.

Kappa

Kappa is a measure of agreement. In the Kappa statistic, the rows and columns correspond to the responses of two judges. The judges agree along the diagonal and disagree off the diagonal. Let $p_O = p_{11} + p_{22}$ denote the probability that the two judges agree, and let $p_C = p_{1\bullet}p_{\bullet 1} + p_{2\bullet}p_{\bullet 2}$ denote the expected probability of agreement under the independence model. Kappa is then given by $(p_O - p_C)/(1 - p_C)$.

McNemar Test

The McNemar test is also a test of symmetry in square contingency tables. It tests the null hypothesis $H_O : \theta_{ij} = \theta_{ji}$. The test statistic with 1 degree of freedom is computed as

$$\sum_{i<j} \frac{\left(x_{ij} - x_{ji}\right)^2}{\left(x_{ij} + x_{ji}\right)}$$

Its exact probability may be computed via the binomial distribution.

Comments

Informational errors

		Description
4	8	At least one marginal total is zero. The remainder of the analysis cannot proceed.
3	9	Some expected table values are less than 1.0. Some asymptotic *p*-values may not be good.
3	10	Some expected table values are less than 2.0. Some asymptotic *p*-values may not be good.
3	11	20% of the table expected values are less than 5.

Example

The following example from Kendall and Stuart (1979, pages 582-583) compares the teeth in breast-fed versus bottle-fed babies.

```
      USE CTTWO_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCHIC, LDEXPE, LDSTAT, LDTABL
      PARAMETER  (IPRINT=1, LDCHIC=3, LDEXPE=3, LDSTAT=24, LDTABL=2)
!
      REAL       CHICTR(LDCHIC,3), CHISQ(15), EXPECT(LDEXPE,3), &
                 STAT(LDSTAT,5), TABLE(LDTABL,2)
!
      DATA TABLE/4, 1, 16, 21/


!
      CALL CTTWO (TABLE, EXPECT, CHICTR, CHISQ, STAT, IPRINT=IPRINT)
      END
```

Output

```
       TABLE
         1       2
1    4.00   16.00
2    1.00   21.00

                Expected values
                Col 1       Col 2    Marginal
Row 1         2.3810     17.6190     20.0000
Row 2         2.6190     19.3810     22.0000
Marginal      5.0000     37.0000     42.0000

      Contributions to chi-squared
            Col 1       Col 2       Total
Row 1      1.1010      0.1488      1.2497
Row 2      1.0009      0.1353      1.1361
Total      2.1018      0.2840      2.3858

             CHISQ
                           1
```

```
     Pearson chi-squared      2.3858
     p-value                  0.1224
     Degrees of freedom       1.0000
     Likelihood ratio         2.5099
     p-value                  0.1131
     Yates chi-squared        1.1398
     p-value                  0.2857
     Fisher (one tail)        0.1435
     Fisher (two tail)        0.1745
     Exact mean               1.0244
     Exact std dev            1.3267
     Phi                      0.2383
     Max possible phi         0.3855
     Contingency coef.        0.2318
     Max possible coef.       0.3597
```

STAT					
	Statistic	Std err.	Std err. 0	t-value	p-value
Gamma	0.6800	0.3135	0.4395	1.5472	0.1218
Kendall’s tau B	0.2383	0.1347	0.1540	1.5472	0.1218
Stuart’s tau C	0.1542	0.0997	NaN	1.5472	0.1218
Somers’ D row	0.1545	0.0999	0.0999	1.5472	0.1218
Somers’ D col	0.3676	0.1966	0.2376	1.5472	0.1218
Correlation	0.2383	0.1347	0.1540	1.5472	0.1218
Spearman rank	0.2383	0.1347	0.1540	1.5472	0.1218
GK tau row	0.0568	0.0641	NaN	NaN	NaN
GK tau col	0.0568	0.0609	NaN	NaN	NaN
U normed	0.0565	0.0661	NaN	NaN	NaN
U row	0.0819	0.0935	NaN	NaN	NaN
U col	0.0432	0.0516	NaN	NaN	NaN
Lamda sym	0.1200	0.0779	NaN	NaN	NaN
Lamda row	0.0000	0.0000	NaN	NaN	NaN
Lamda col	0.1500	0.1031	NaN	NaN	NaN
Lamda star row	0.0000	0.0000	NaN	NaN	NaN
Lamda star col	0.1761	0.1978	NaN	NaN	NaN
Yule’s Q	0.6800	0.3135	0.4770	1.4255	0.1540
Yule’s Y	0.3923	0.2467	0.2385	1.6450	0.1000
Ratio	5.2500	NaN	NaN	NaN	NaN
Log ratio	1.6582	1.1662	0.9540	1.7381	0.0822
Linear trend	-0.1545	0.1001	NaN	-1.5446	0.1224
Kappa	0.1600	0.1572	0.1600	1.0000	0.3173
McNemar	13.2353	1.0000	NaN	0.0000	0.0003

```
*** WARNING  ERROR 11 from CTTWO.  Twenty percent of the table expected
***          values are less than 5.0.
```

CTCHI

Performs a chi-squared analysis of a two-way contingency table.

Required Arguments

TABLE — NROW by NCOL matrix containing the observed counts in the contingency table. (Input)

EXPECT — (NROW + 1) by (NCOL + 1) matrix containing the expected values of each cell in TABLE, under the null hypothesis, in the first NROW rows and NCOL columns and the marginal totals in the last row and column. (Output)

CHICTR — (NROW +1) by (NCOL +1) matrix containing the contributions to chi-squared for each cell in TABLE in the first NROW rows and NCOL columns. (Output)
The last row and column contain the total contribution to chi-squared for that row or column.

CHISQ — Vector of length 10 containing chi-squared statistics associated with this contingency table. (Output)

I	CHISQ(I)
1	Pearson chi-squared statistic
2	Probability of a larger Pearson chi-squared
3	Degrees of freedom for chi-squared
4	Likelihood ratio G^2 (chi-squared)
5	Probability of a larger G^2
6	Exact mean
7	Exact standard deviation

The following statistics are based upon the chi-squared statistic CHISQ(1). If ICMPT = 1, NaN (not a number) is reported.

I	CHISQ(I)
8	Phi
9	Contingency coefficient
10	Cramer's *V*

STAT — 23 by 5 matrix containing statistics associated with this table. (Output)
If `ICMPT` = 1, `STAT` is not referenced and may be a vector of length 1. Each row of the matrix corresponds to a statistic.

Row	Statistic
1	Gamma
2	Kendall's τ_b
3	Stuart's τ_c
4	Somers' *D* for rows given columns
5	Somers' *D* for columns given rows
6	Product moment correlation
7	Spearman rank correlation
8	Goodman and Kruskal τ for rows given columns
9	Goodman and Kruskal τ for columns given rows
10	Uncertainty coefficient *U* (symmetric)
11	Uncertainty $U_{r\mid c}$ (rows)
12	Uncertainty $U_{c\mid r}$ (columns)
13	Optimal prediction λ (symmetric)
14	Optimal prediction $\lambda_{r\mid c}$ (rows)
15	Optimal prediction $\lambda_{c\mid r}$ (columns)
16	Optimal prediction $\lambda^*_{r\mid c}$ (rows)
17	Optimal prediction $\lambda^*_{c\mid r}$ (columns)
18	Test for linear trend in row probabilities if `NROW`= 2. If `NROW` is not 2, a test for linear trend in column probabilities if `NCOL`= 2.
19	Kruskal-Wallis test for no row effect
20	Kruskal-Wallis test for no column effect
21	Kappa (square tables only)

Row	Statistic
22	McNemar test of symmetry (square tables only)
23	McNemar one degree of freedom test of symmetry (square tables only)

If a statistic cannot be computed, its value is reported as NaN (not a number). The columns are as follows:

Column	Statistic
1	The estimated statistic
2	Its standard error for any parameter value
3	Its standard error under the null hypothesis
4	The *t* value for testing the null hypothesis
5	*p*-value of the test in column 4

In the McNemar tests, column 1 contains the statistic, column 2 contains the chi-squared degrees of freedom, column 4 contains the exact *p*-value (one degree of freedom only), and column 5 contains the chi-squared asymptotic *p*-value. The Kruskal-Wallis test is the same except no exact *p*-value is computed.

Optional Arguments

NROW — Number of rows in the table. (Input)
Default: `NROW` = size (`TABLE`,1).

NCOL — Number of columns in the table. (Input)
Default: `NCOL` = size (`TABLE`,2).

LDTABL — Leading dimension of `TABLE` exactly as specified in the dimension statement of the calling program. (Input)
Default: `LDTABL` = size (`TABLE`,1).

ICMPT — Computing option. (Input)
If `ICMPT` = 0, all of the values in `CHISQ` and `STAT` are computed. `ICMPT` = 1 means compute only the first 5 values of `CHISQ` and none of the values in `STAT`. (All values not computed are set to NaN (not a number).
Default: `ICMPT` = 0.

IPRINT — Printing option. (Input)
`IPRINT` = 0 means no printing is performed. If `IPRINT` = 1, printing is performed.
Default: `IPRINT` = 0.

LDEXPE — Leading dimension of **EXPECT** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDEXPE** = size (**EXPECT**,1).

LDCHIC — Leading dimension of **CHICTR** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCHIC** = size (**CHICTR**,1).

LDSTAT — Leading dimension of **STAT** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDSTAT** = size (**STAT**,1).

FORTRAN 90 Interface

Generic: **CALL CTCHI** (**TABLE**, **EXPECT**, **CHICTR**, **CHISQ**, **STAT** [, ...])
Specific: The specific interface names are **S_CTCHI** and **D_CTCHI**.

FORTRAN 77 Interface

Single: **CALL CTCHI** (**NROW**, **NCOL**, **TABLE**, **LDTABL**, **ICMPT**, **IPRINT**, **EXPECT**, **LDEXPE**, **CHICTR**, **LDCHIC**, **CHISQ**, **STAT**, **LDSTAT**)
Double: The double precision name is **DCTCHI**.

Description

Routine **CTCHI** computes statistics associated with an $r \times c$ (**NROW** × **NCOL**) contingency table. The routine **CTCHI** always computes the chi-squared test of independence, expected values, contributions to chi-squared, and row and column marginal totals. Optionally, when **ICMPT** = 0, **CTCHI** can compute some measures of association, correlation, prediction, uncertainty, the McNemar test for symmetry, a test for linear trend, the odds and the log odds ratio, and the Kappa statistic.

Other IMSL routines that may be of interest include **TETCC** in Chapter 3, for computing the tetrachoric correlation coefficient, **CTTWO**, for computing statistics in a 2 × 2 contingency table, and **CTPRB**, for computing the exact probability of an $r \times c$ contingency table.

Notation

Let x_{ij} denote the observed cell frequency in the $_{ij}$ cell of the table and n denote the total count in the table. Let $p_{ij} = p_{i\bullet}p_{\bullet j}$ denote the predicted cell probabilities under the null hypothesis of independence where $p_{i\bullet}$ and $p_{\bullet j}$ are the row and column marginal relative frequencies, respectively. Next, compute the expected cell counts as $e_{ij} = n\, p_{ij}$.

Also required in the following are a_{uv} and b_{uv}, $u, v = 1, \ldots, n$. Let (r_s, c_s) denote the row and column response of observation s. Then, $a_{uv} = 1, 0$, or -1, depending upon whether $r_u < r_v$, $r_u = r_v$, or $r_u > r_v$, respectively. The b_{uv} are similarly defined in terms of the c_s's.

The Chi-squared Statistics

For each cell in the table, the contribution to $\mathbf{X}^2$ is given as $(x_{ij} - e_{ij})^2/e_{ij}$. The Pearson chi-squared statistic (denoted $\mathbf{X}^2$) is computed as the sum of the cell contributions to chi-squared. It has $(r - 1)(c - 1)$ degrees of freedom and tests the null hypothesis of independence, i.e., that $H_0 : p_{ij} = p_{i\bullet}p_{\bullet j}$. The null hypothesis is rejected if the computed value of $\mathbf{X}^2$ is too large.

Compute G^2, the maximum likelihood equivalent of $\mathbf{X}^2$, as

$$G^2 = -2\sum_{i,j} x_{ij} \ln\left(x_{ij}/np_{ij}\right)$$

G^2 is asymptotically equivalent to $\mathbf{X}^2$ and tests the same hypothesis with the same degrees of freedom.

Measures Related to Chi-squared (Phi, Contingency Coefficient, and Cramer's *V*)

Three measures related to chi-squared but that do not depend upon the sample size are

phi,

$$\phi = \sqrt{\chi^2/n}$$

the contingency coefficient,

$$P = \sqrt{\chi^2/\left(n + \chi^2\right)}$$

and Cramer's *V*,

$$V = \sqrt{\chi^2/\left(n \min\left(r,c\right)\right)}$$

Since these statistics do not depend upon sample size and are large when the hypothesis of independence is rejected, they may be thought of as measures of association and may be compared across tables with different sized samples. While both *P* and *V* have a range between 0.0 and 1.0, the upper bound of *P* is actually somewhat less than 1.0 for any given table (see Kendall and Stuart 1979, page 587). The significance of all three statistics is the same as that of the $\mathbf{X}^2$ statistic, `CHISQ`(1).

The distribution of the $\mathbf{X}^2$ statistic in finite samples approximates a chi-squared distribution. To compute the exact mean and standard deviation of the $\mathbf{X}^2$ statistic, Haldane (1939) uses the multinomial distribution with fixed table marginals. The exact mean and standard deviation generally differ little from the mean and standard deviation of the associated chi-squared distribution.

Standard Errors and p-values For Some Measures of Association

In rows 1 through 7 of **STAT**, estimated standard errors and asymptotic *p*-values are reported. Estimates of the standard errors are computed in two ways. The first estimate, in column 2 of matrix **STAT**, is asymptotically valid for any value of the statistic. The second estimate, in column 3 of the matrix, is only correct under the null hypothesis of no association. The *z*-scores in column 4 of matrix **STAT** are computed using this second estimate of the standard errors. The *p*-values in column 5 are computed from this *z*-score. See Brown and Benedetti (1977) for a discussion and formulas for the standard errors in column 3.

Measures of Association for Ranked Rows and Columns

The measures of association, ϕ, *P*, and *V*, do not require any ordering of the row and column categories. Routine **CTCHI** also computes several measures of association for tables in which the rows and column categories correspond to ranked observations. Two of these tests, the product-moment correlation and the Spearman correlation, are correlation coefficients computed using assigned scores for the row and column categories. The cell indices are used for the product-moment correlation while the average of the tied ranks of the row and column marginals is used for the Spearman rank correlation. Other scores are possible.

Gamma, Kendall's τ_b, Stuart's τ_c, and Somers' *D* are measures of association that are computed like a correlation coefficient in the numerator. In all of these measures, the numerator is computed as the "covariance" between the a_{uv}'s and b_{uv}'s defined above, i.e., as

$$\sum_u \sum_v a_{uv} b_{uv}$$

Recall that a_{uv} and b_{uv} can take values -1, 0, or 1. Since the product $a_{uv}b_{uv} = 1$ only if a_{uv} and b_{uv} are both 1 or are both -1, it is easy to show that this "covariance" is twice the total number of agreements minus the number of disagreements where a disagreement occurs when $a_{uv}b_{uv} = -1$.

Kendall's τ_b is computed as the correlation between the a_{uv}'s and the b_{uv}'s (see Kendall and Stuart 1979, page 593). In a rectangular table ($r \neq c$), Kendall's τ_b cannot be 1.0 (if all marginal totals are positive). For this reason, Stuart suggested a modification to the denominator of τ in which the denominator becomes the largest possible value of the "covariance." This maximizing value is approximately $n^2m/(m - 1)$, where $m = \min(r, c)$. Stuart's τ_c uses this approximate value in its denominator. For large *n*, $\tau_c \approx m\ \tau_b/(m - 1)$.

Gamma can be motivated in a slightly different manner. Because the "covariance" of the a_{uv}'s and the b_{uv}'s can be thought of as twice the number of agreements minus the disagreements, $(2(A - D)$, where A is the number of agreements and D is the number of disagreements), gamma is motivated as the probability of agreement minus the probability of disagreement, given that either agreement or disagreement occurred. This is just $\gamma = (A - D)/(A + D)$.

Two definitions of Somers' D are possible, one for rows and a second for columns. Somers' D for rows can be thought of as the regression coefficient for predicting a_{uv} from b_{uv}. Moreover, Somers' D for rows is the probability of agreement minus the probability of disagreement, given that the column variable, b_{uv}, is not zero. Somers' D for columns is defined in a similar manner.

A discussion of all of the measures of association in this section can be found in Kendall and Stuart (1979, starting on page 592).

Measures of Prediction and Uncertainty

The Optimal Prediction Coefficients

The measures in this section do not require any ordering of the row or column variables. They are based entirely upon probabilities. Most are discussed in Bishop, Feinberg, and Holland (1975, page 385).

Consider predicting (or classifying) the column for a given row in the table. Under the null hypothesis of independence, one would choose the column with the highest column marginal probability for all rows. In this case, the probability of misclassification for any row is one minus this marginal probability. If independence is not assumed, then within each row one would choose the column with the highest row conditional probability, and the probability of misclassification for the row becomes one minus this conditional probability.

Define the optimal prediction coefficient $\lambda_{c|r}$ for predicting columns from rows as the proportion of the probability of misclassification that is eliminated because the random variables are not independent. It is estimated by

$$\lambda_{c|r} = \frac{(1 - p_{\bullet m}) - (1 - \sum_i p_{im})}{1 - p_{\bullet m}}$$

where m is the index of the maximum estimated probability in the row (p_{im}) or row margin ($p_{\bullet m}$). A similar coefficient is defined for predicting the rows from the columns. The symmetric version of the optimal prediction λ is obtained by summing the numerators and denominators of $\lambda_{r|c}$ and $\lambda_{c|r}$ and by dividing. Standard errors for these coefficients are given in Bishop, Feinberg, and Holland (1975, page 388).

A problem with the optimal prediction coefficients λ is that they vary with the marginal probabilities. One way to correct for this is to use row conditional probabilities. The optimal prediction λ^* coefficients are defined as the corresponding λ coefficients in which one first adjusts the row (or column) marginals to the same number of observations. This yields

$$\lambda^*_{c|r} = \frac{\sum_i \max_j p_{j|i} - \max_j(\sum_i p_{j|i})}{R - \max_j \sum_i p_{j|i}}$$

where *i* indexes the rows, *j* indexes the columns, and $p_{j|i}$ is the (estimated) probability of column *j* given row *i*.

$$\lambda^*_{r|c}$$

is similarly defined.

Goodman and Kruskal τ

A second kind of prediction measure attempts to explain the proportion of the explained variation of the row (column) measure given the column (row) measure. Define the total variation in the rows to be

$$n/2 - (\sum_i x_{i\bullet}^2)/(2n)$$

Note that this is 1/(2*n*) times the sums of squares of the a_{uv}'s.

With this definition of variation, the Goodman and Kruskal τ coefficient for rows is computed as the reduction of the total variation for rows accounted for by the columns, divided by the total variation for the rows. To compute the reduction in the total variation of the rows accounted for by the columns, note that the total variation for the rows within column *j* is defined as

$$q_j = x_{\bullet j}/2 - (\sum_i x_{ij}^2)/(2x_{i\bullet})$$

The total variation for rows within columns is the sum of the q_j's. Consistent with the usual methods in the analysis of variance, the reduction in the total variation is given as the difference between the total variation for rows and the total variation for rows within the columns.

Goodman and Kruskal's τ for columns is similarly defined. See Bishop, Feinberg, and Holland (1975, page 391) for the standard errors.

The Uncertainty Coefficients

The uncertainty coefficient for rows is the increase in the log-likelihood that is achieved by the most general model over the independence model, divided by the marginal log-likelihood for the rows. This is given by

$$U_{r|c} = \frac{\sum_{i,j} x_{ij} \log(x_{i\bullet} x_{\bullet j} / (n x_{ij}))}{\sum_i x_{i\bullet} \log(x_{i\bullet} / n)}$$

The uncertainty coefficient for columns is similarly defined. The symmetric uncertainty coefficient contains the same numerator as $U_{r|c}$ and $U_{c|r}$ but averages the denominators of these two statistics. Standard errors for U are given in Brown (1983).

Kruskal-Wallis

The Kruskal-Wallis statistic for rows is a one-way analysis-of-variance-type test that assumes the column variable is monotonically ordered. It tests the null hypothesis that no row populations are identical, using average ranks for the column variable. The Kruskal-Wallis statistic for columns is similarly defined. Conover (1980) discusses the Kruskal-Wallis test.

Test for Linear Trend

When there are two rows, it is possible to test for a linear trend in the row probabilities if one assumes that the column variable is monotonically ordered. In this test, the probabilities for row 1 are predicted by the column index using weighted simple linear regression. This slope is given by

$$\hat{\beta} = \frac{\sum_j x_{\bullet j} \left(x_{1j} / x_{\bullet j} - x_{1\bullet} / n \right) \left(j - \bar{j} \right)}{\sum_j x_{\bullet j} \left(j - \bar{j} \right)^2}$$

where

$$\bar{j} = \sum_j x_{\bullet j} j / n$$

is the average column index. An asymptotic test that the slope is zero may then be obtained (in large samples) as the usual regression test of zero slope.

In two-column data, a similar test for a linear trend in the column probabilities is computed. This test assumes that the rows are monotonically ordered.

Kappa

Kappa is a measure of agreement computed on square tables only. In the Kappa statistic, the rows and columns correspond to the responses of two judges. The judges agree along the diagonal and disagree off the diagonal. Let

$$p_o = \sum_i x_{ii} / n$$

denote the probability that the two judges agree, and let

$$p_c = \sum_i e_{ii} / n$$

denote the expected probability of agreement under the independence model. Kappa is then given by $(p_O - p_C)/(1 - p_C)$.

McNemar Tests

The McNemar test is a test of symmetry in a square contingency table, that is, it is a test of the null hypothesis $H_O : \theta_{ij} = \theta_{ji}$. The multiple-degrees-of-freedom version of the McNemar test with $r(r - 1)/2$ degrees of freedom is computed as

$$\sum_{i<j} \frac{\left(x_{ij} - x_{ji}\right)^2}{\left(x_{ij} + x_{ji}\right)}$$

The single-degree-of-freedom test assumes that the differences $x_{ij} - x_{ji}$ are all in one direction. The single-degree-of-freedom test will be more powerful than the multiple-degrees-of-freedom test when this is the case. The test statistic is given as

$$\frac{(\sum_{i<j}(x_{ij} - x_{ji}))^2}{\sum_{i<j}(x_{ij} + x_{ji})}$$

Its exact probability may be computed via the binomial distribution.

Comments

Informational errors

Type	Code	Description
3	1	Twenty percent of the expected values are less than 5.
3	2	The degrees of freedom for chi-squared are greater than 30. The exact mean, standard deviation, and normal distribution function should be used.
3	3	Some expected table values are less than 2. Some asymptotic *p*-values may not be good.
3	4	Some expected values are less than 1. Some asymptotic *p*-values may not be good.

Example

The following example is taken from Kendall and Stuart (1979). It involves the distance vision in the right and left eyes, and especially illustrates the use of Kappa and McNemar tests. Most other test statistics are also computed.

```
      USE CTCHI_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDSTAT, NCOL, NROW
      PARAMETER  (IPRINT=1, LDSTAT=23, NCOL=4, NROW=4)
!
      REAL       CHICTR(NROW+1,NCOL+1), CHISQ(10), EXPECT(NROW+1,NCOL+1), &
                     STAT(LDSTAT,5), TABLE(NROW,NCOL)
!
      DATA TABLE/821, 116, 72, 43, 112, 494, 151, 34, 85, 145, 583, &
          106, 35, 27, 87, 331/


!
      CALL CTCHI (TABLE, EXPECT, CHICTR, CHISQ, STAT, IPRINT=IPRINT)
      END
```

Output

```
              Table Values
           1       2       3       4
 1   821.0   112.0    85.0    35.0
 2   116.0   494.0   145.0    27.0
 3    72.0   151.0   583.0    87.0
 4    43.0    34.0   106.0   331.0

                        Expected Values
          row totals in column 5, column totals in row 5
               1           2           3           4           5
 1      341.69      256.92      298.49      155.90     1053.00
 2      253.75      190.80      221.67      115.78      782.00
 3      289.77      217.88      253.14      132.21      893.00
 4      166.79      125.41      145.70       76.10      514.00
 5     1052.00      791.00      919.00      480.00     3242.00

                   Contibutions to Chi-squared
```

row totals in column 5, column totals in row 5

	1	2	3	4	5
1	672.36	81.74	152.70	93.76	1000.56
2	74.78	481.84	26.52	68.08	651.21
3	163.66	20.53	429.85	15.46	629.50
4	91.87	66.63	10.82	853.78	1023.10
5	1002.68	650.73	619.88	1031.08	3304.37

Chi-square Statistics

Pearson	3304.3682
p-value	0.0000
DF	9.0000
G**2	2781.0188
p-value	0.0000
Exact mean	9.0028
Exact std.	4.2402
Phi	1.0096
P	0.7105
Cramer’s V	0.5829

Table Statistics

	statistic	standard error	std. error under Ho	t-value testing Ho	p-value
Gamma	0.7757	0.0123	0.0149	52.19	0.0000
Tau B	0.6429	0.0122	0.0123	52.19	0.0000
Tau C	0.6293	0.0121	NaN	52.19	0.0000
D-Row	0.6418	0.0122	0.0123	52.19	0.0000
D-Column	0.6439	0.0122	0.0123	52.19	0.0000
Correlation	0.6926	0.0128	0.0172	40.27	0.0000
Spearman	0.6939	0.0127	0.0127	54.66	0.0000
GK tau rows	0.3420	0.0123	NaN	NaN	NaN
GK tau col.	0.3430	0.0122	NaN	NaN	NaN
U - Sym.	0.3171	0.0110	NaN	NaN	NaN
U - rows	0.3178	0.0110	NaN	NaN	NaN
U - cols.	0.3164	0.0110	NaN	NaN	NaN
Lambda-sym.	0.5373	0.0124	NaN	NaN	NaN
Lambda-row	0.5374	0.0126	NaN	NaN	NaN
Lambda-col.	0.5372	0.0126	NaN	NaN	NaN
l-star-rows	0.5506	0.0136	NaN	NaN	NaN
l-star-col.	0.5636	0.0127	NaN	NaN	NaN
Lin. trend	NaN	NaN	NaN	NaN	NaN
Kruskal row	1561.4861	3.0000	NaN	NaN	0.0000
Kruskal col	1563.0300	3.0000	NaN	NaN	0.0000
Kappa	0.5744	0.0111	0.0106	54.36	0.0000
McNemar	4.7625	6.0000	NaN	NaN	0.5746
McNemar df=1	0.9487	1.0000	NaN	0.35	0.3301

CTPRB

Computes exact probabilities in a two-way contingency table.

Required Arguments

TABLE — NROW by NCOL matrix containing the contingency table cell frequencies. (Input)

PRE — Probability of a more extreme table where "extreme" is taken in the Neyman-Pearson sense. (Output)
A table is more extreme if its probability (for fixed marginals) is less than or equal to PRT.

Optional Arguments

NROW — Number of rows in the contingency table. (Input)
Default: NROW = size (TABLE,1).

NCOL — Number of columns in the contingency table. (Input)
Default: NCOL = size (TABLE,2).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

PRT — Probability of the observed table assuming fixed row and column marginal totals. (Output)

PCHEK — Sum of the probabilities of all tables with the same marginal totals. (Output)
PCHEK should be 1.0. Deviation from 1.0 indicates a numerical error.

FORTRAN 90 Interface

Generic: CALL CTPRB (TABLE, PRE [, ...])
Specific: The specific interface names are S_CTPRB and D_CTPRB.

FORTRAN 77 Interface

Single: CALL CTPRB (NROW, NCOL, TABLE, LDTABL, PRT, PRE, PCHEK)
Double: The double precision name is DCTPRB.

Description

Routine **CTPRB** computes exact probabilities for an $r \times c$ contingency table for fixed row and column marginals where r = **NROW** and c = **NCOL**. Let f_{ij} denote the element in row i and column j of a table, and let $f_{i\bullet}$ and $f_{\bullet j}$ denote the row and column marginals. Under the independence hypothesis, the (conditional) probability for fixed marginals of a table is given by

$$P_f = \frac{\prod_{i=1}^{r} f_{i\bullet}! \prod_{j=1}^{c} f_{\bullet j}!}{f_{\bullet\bullet}! \prod_{i=1}^{r} \prod_{j=1}^{c} f_{ij}!}$$

where $f_{\bullet\bullet}$ is the total number of counts in the table and $x!$ denotes x factorial. When the f_{ij} are obtained from the input table (f_{ij} = **TABLE**(i, j)), P_f = **PRT**. **PRE** is the sum over all more extreme tables of the probability of each table.

In **CTPRB**, a more extreme table is defined in the probabilistic sense. Table X is more extreme than the input table if the conditional probability computed for table X (for the same marginal sums) is less than the conditional probability computed for the input table. The user should note that this definition of "more extreme" can be considered as "two-sided" in the cell counts.

Because **CTPRB** uses total enumeration in computing the probability of a more extreme table, the amount of computer time required increases very rapidly with the size of the table. Tables, with either a large total count $f_{\bullet\bullet}$ or in which the product rc is not small, should not be analyzed with **CTPRB**. Rather, either the approximate methods of Agresti, Wackerly, and Boyett (1979) should be used or algorithms that do not require total enumeration should be used (see Pagano and Halvorsen [1981], or Mehta and Patel [1983]).

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2PRB/DC2PRB**. The reference is:

   ```
   CALL C2PRB (NROW, NCOL, TABLE, LDTABL, PRT, PRE, PCHCK, IWK)
   ```

   The additional argument is:

   IWK — Work vector of length (**NROW** + 2)(**NCOL** + 2).

2. Informational error

Type	Code	Description
3	1	There are no observed counts in TABLE. PRE, PRT, and PCHEK are set to NaN (not a number).

3. Routine **CTPRB** computes a two-tailed Fisher exact probability in 2 by 2 tables. For one-tailed Fisher exact probabilities, use routine **CTTWO**.

Example

In this example, **CTPRB** is used to compute the exact conditional probability for a 2×2 contingency table. The input table is given as:

$$\begin{bmatrix} 8 & 12 \\ 8 & 2 \end{bmatrix}$$

```
      USE UMACH_INT
      USE CTPRB_INT

      IMPLICIT   NONE
      INTEGER    NCOL, LDTABL
      PARAMETER  (NCOL=2, LDTABL=2)
!
      INTEGER    NOUT
      REAL       PCHEK, PRE, PRT, TABLE(LDTABL,NCOL)
!
      DATA TABLE/8, 8, 12, 2/
!
      CALL UMACH (2, NOUT)
!
      CALL CTPRB (TABLE, PRE, PRT=PRT, PCHEK=PCHEK)
!
      WRITE(NOUT,'('' PRT = '', F12.4, /, '' PRE = '', F12.4, /, &
           & '' PCHEK = '', F10.4)')  PRT, PRE, PCHEK
      END
```

Output

```
 PRT =       0.0390
 PRE =       0.0577
 PCHEK =     1.0000
```

CTEPR

Computes Fisher's exact test probability and a hybrid approximation to the Fisher exact test probability for a contingency table using the network algorithm.

Required Arguments

TABLE — NROW by NCOL matrix containing the contingency table. (Input)

PRE — Table *p*-value. (Output)
PRE is the probability of a more extreme table, where "extreme" is in a probabilistic sense. If EXPECT < 0, then the Fisher exact probability is returned. Otherwise, a hybrid approximation to Fisher's exact probability is computed.

Optional Arguments

NROW — The number of rows in the table. (Input)
Default: NROW = size (TABLE,1).

NCOL — The number of columns in the table. (Input)
Default: NCOL = size (TABLE,2).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

EXPECT — Expected value used in the hybrid approximation to Fisher's exact test algorithm for deciding when to use asymptotic probabilities when computing path lengths. (Input)
Default: EXPECT = 5.0.
If EXPECT ≤ 0.0, then asymptotic theory probabilities are not used and Fisher exact test probabilities are computed. Otherwise, asymptotic probabilities are used in computing path lengths whenever PERCNT or more of the cells in the table for which path lengths are to be computed have estimated expected values of EXPECT or more, with no cell having expected value less than EMIN. See the "Description" section for details. Use EXPECT = 5.0 to obtain the "Cochran" condition.

PERCNT — Percentage of remaining cells that must have estimated expected values greater than EXPECT before asymptotic probabilities can be used in computing path lengths. (Input)
Default: PERCNT = 80.0.
See argument EXPECT for details. Use PERCNT = 80.0 to obtain the "Cochran" condition.

EMIN — Minimum cell estimated expected value allowed for asymptotic chi-squared probabilities to be used. (Input)
Default: **EMIN** = 1.0.
See argument **EXPECT** for details. Use **EMIN** = 1.0 to obtain the "Cochran" condition.

PRT — Probability of the observed table for fixed marginal totals. (Output)

FORTRAN 90 Interface

Generic: **CALL CTEPR** (**TABLE**, **PRE** [, ...])
Specific: The specific interface names are **S_CTEPR** and **D_CTEPR**.

FORTRAN 77 Interface

Single: **CALL CTEPR** (**NROW**, **NCOL**, **TABLE**, **LDTABL**, **EXPECT**, **PERCNT**, **EMIN**, **PRT**, **PRE**)
Double: The double precision name is **DCTEPR**.

Description

Routine **CTEPR** computes Fisher exact probabilities or a hybrid algorithm approximation to Fisher exact probabilities for a $r \times c$ contingency tables with fixed row and column marginals where r = **NROW** is the number of rows in the table and c = **NCOL** is the number of columns in the table. Let f_{ij} denote the frequency count in row i and column j of a table, and let $f_{i\bullet}$ and $f_{\bullet j}$ denote the total row and column frequency count for row i and column j, respectively. Under the independence hypothesis, the (conditional) probability of the observed table for fixed row and column marginal totals is given by

$$P_f = \frac{\prod_{i=1}^{r} f_{i\bullet}! \prod_{j=1}^{c} f_{\bullet j}!}{f_{\bullet\bullet}! \prod_{i=1}^{r} \prod_{j=1}^{c} f_{ij}!}$$

where $f_{\bullet\bullet}$ is the total number of counts in the table and $x!$ denotes x factorial. When the f_{ij} are equal to the input table so that f_{ij} = **TABLE** (i, j), then let P_0 = **PRT** be the resulting value for P_f.

In **CTEPR**, a more extreme table is defined in the probabilistic sense. Table X is more extreme than the input table if the conditional probability computed for table X (for the same marginal sums) is less than the conditional probability computed for the input table. Let p = **PRE** be the probability of a more extreme table. Then

$$p = \sum_{P \le P_0} P_f$$

The user should note that this definition of "more extreme" can be considered as "two-sided" in the cell counts.

Routine CTEPR uses the hybrid network algorithm of Mehta and Patel (1983, 1986a, 1986b) with the Clarkson and Fan (1989) modifications to compute the probability of a more extreme table. The hybrid algorithm uses asymptotic probabilities for tables encountered in which PERCNT percent of the table expected values are greater than or equal to EXPECT, and all expected values are greater than EMIN. When PERCNT = 80, EXPECT = 5, and EMIN = 1, this is the "Cochran" rule. Although the hybrid network algorithm can be orders of magnitude faster than the total enumeration algorithm used in routine CTPRB, the amount of computer time required by CTEPR still increases very rapidly with the size of the table. Caution should be used whenever computer time is a consideration.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2EPR/DC2EPR. The reference is:

```
CALL C2EPR (NROW, NCOL, TABLE, LDTABL, EXPECT, PERCNT, EMIN, PRT, PRE, FACT,
    ICO, IRO, KYY, IDIF, IRN, KEY, LDKEY, IPOIN, STP, LDSTP, IFRQ, DLP, DSP, TM,
    KEY2, IWK, RWK)
```

   The additional arguments are as follows:

   FACT — Work vector of length NTOT + 1 where NTOT is the total count in the table.

   ICO — Work vector of length MX where MX = max(NROW, NCOL).

   IRO — Work vector of length MX.

   KYY — Work vector of length MX.

   IDIF — Work vector of length MN where MN = max(NROW, NCOL).

   IRN — Work vector of length MN.

   KEY — Work vector of length 2 * LDKEY.

   LDKEY — Leading dimension of KEY exactly as specified in the dimension statement in the calling program. (Input)

   IPOIN — Work vector of length 2 * LDKEY.

   STP — Work vector of length 2 * LDSTP.

   LDSTP — Leading dimension of STP exactly as specified in the dimension statement in the calling program. (Input)

   IFRQ — Work vector of length 6 * LDSTP.

   DLP — Work vector of length 2 * LDKEY.

   DSP — Work vector of length 2 * LDKEY.

   TM — Work vector of length 2 * LDKEY.

   KEY2 — Work vector of length 2 * LDKEY.

   IWK — Work vector of length max((NROW + NCOL + 1)(5 + 2 * MX), 800 + 7 * MX).

   RWK — Work vector of length max(400 + MX + 1, NROW + NCOL + 1).

The exact value of **LDKEY** and **LDSTP** required is not known in advance. Common values to try are **LDKEY** = 1000 and **LDSTP** = 30000.

2. Informational errors

Type	Code	Description
3	1	All of the elements of TABLE are zero.
4	2	The product of the marginal totals is greater than can be exactly represented in an integer variable so the hash table key cannot be computed. The computations cannot proceed.
4	3	LDKEY is too small. To increase LDKEY when invoking CTEPR/DCTEPR, increase the total workspace used. A doubling of the total workspace is a good place to begin.
4	4	LDSTP is too small. To increase LDSTP when invoking CTEPR/DCTEPR, increase the total workspace used. A doubling of the total workspace is a good place to begin.
4	5	The current value for IWKIN is too small. It is not possible to give the value for IWKIN required, but you might try doubling the amount. Refer to IWKIN in the Reference Material section.

3. Routine **CTEPR**/**DCTEPR** will use all available workspace. It is not unusual for **CTEPR**/**DCTEPR** to require 200,000 floating-point units of workspace.

4. When **C2EPR**/**DC2EPR** is called by **CTEPR**/**DCTEPR**, **LDSTP** = 30 * **LDKEY**.

5. Although not a restriction, it is not generally practical to call this routine with large tables that are not sparse and in which the hybrid approximation to Fisher's exact test (see the Description section) has little effect. For example, although it is feasible to compute exact probabilities for the table

$$\begin{matrix} 1 & 8 & 5 & 4 & 4 & 2 & 2 \\ 5 & 3 & 3 & 4 & 3 & 1 & 0 \\ 10 & 1 & 4 & 0 & 0 & 0 & 0 \end{matrix}$$

computing exact probabilities for a similar table that has been enlarged by the addition of an extra row (or column) may not be feasible.

Example

In this example, **CTEPR** is used to compute the hybrid approximation to the Fisher exact probability for a 3 × 6 contingency table using the Cochran condition. Because of the large initial counts and the input arguments **EXPECT** = 5, **PERCNT** = 80, and **EMIN** = 1, the hybrid algorithm significantly reduces the computation effort in this example. The input table is given as

$$\begin{bmatrix} 20 & 20 & 0 & 0 & 0 \\ 10 & 10 & 2 & 2 & 1 \\ 20 & 20 & 0 & 0 & 0 \end{bmatrix}$$

```
      USE UMACH_INT
      USE CTEPR_INT

      IMPLICIT   NONE
      INTEGER    LDTABL, NCOL
      PARAMETER  (NCOL=5, LDTABL=3)
!
      INTEGER    NOUT
      REAL       PRE, PRT, TABLE(LDTABL,NCOL)
!
      DATA TABLE/20.0, 10.0, 20.0, 20.0, 10.0, 20.0, 0.0, 2.0, 0.0, &
          0.0, 2.0, 0.0, 0.0, 1.0, 0.0/
!
      CALL UMACH (2, NOUT)
!
      CALL CTEPR (TABLE, PRE, PRT=PRT)
!
      WRITE (NOUT,99999) PRT, PRE
!
99999 FORMAT (' PRT = ', E12.4, '  PRE = ', F8.4)
!
      END
```

Output

```
 PRT =   0.1915E-04  PRE =   0.0601
```

For comparison, the usual asymptotic chi-squared *p*-value (which may be computed through the use of routine CTCHI, do not use CTEPR) is computed as 0.0323, and the Fisher exact probability (which may be computed through CTEPR by setting EXPECT = 0.0) is computed as 0.0598 and requires approximately ten times more computer time than the hybrid method. The Fisher exact probability and the usual asymptotic chi-squared probability will often be quite different. When it may be used, the hybrid algorithm can lead to significantly greater savings in computer time.

PRPFT

Performs iterative proportional fitting of a contingency table using a loglinear model.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Input)

TABLE — Vector of length NCLVAL(1) * NCLVAL (2) * ... * NCLVAL(NCLVAR) containing the entries in the cells of the table to be fit. (Input)
See Comment 3 for comments on the ordering of the elements of TABLE.

NVEF — Vector of length NEF that contains the number of classification variables associated with each effect. (Input)

INDEF — Vector of length NVEF(1) + ...+ NVEF(NEF) that contains, in consecutive positions, the indices of the variables that are included in each effect. (Input)
The entries in INDEF are sequenced so that the first NVEF(1) elements contain the indices of the variables in effect 1, the next NVEF(2) elements of INDEF contain the indices of the variables in effect 2, etc. See Comment 4 for an example.

FIT — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR). (Input/Output)
On input, FIT contains the initial estimates of the cell counts. Structural zeros in the model are specified by setting the corresponding element of FIT to 0.0. All other elements of FIT must be positive. 1.0 may be used if no other estimate of the cell counts is available. See Comment 3 for the ordering of the elements of FIT. On output, FIT contains the fitted table.

Optional Arguments

NCLVAR — Number of classification variables. (Input)
Default: NCLVAR = size (NCLVAL,1).

NEF — Number of effects in the model. (Input)
A marginal table is implied by each effect in the model. Lower order effects should not be included since their inclusion is automatic (e.g., do not include effects *A* or *B* if effect *AB* is in the model).
Default: NEF = size (NVEF,1).

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum deviation between an observed and a fitted marginal total is less than **EPS**. **EPS** = 0.10 is a typical value.
Default: **EPS** = .10.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 15 is a typical value.
Default: **MAXIT** = 30.

FORTRAN 90 Interface

Generic: **CALL PRPFT** (**NCLVAL**, **TABLE**, **NVEF**, **INDEF**, **FIT** [, ...])
Specific: The specific interface names are **S_PRPFT** and **D_PRPFT**.

FORTRAN 77 Interface

Single: **CALL PRPFT** (**NCLVAR**, **NCLVAL**, **TABLE**, **NEF**, **NVEF**, **INDEF**, **EPS**, **MAXIT**, **FIT**)
Double: The double precision name is **DPRPFT**.

Description

Routine **PRPFT** uses the iterative proportional-fitting algorithm to fit a log-linear hierarchical model to a contingency table. Structural zeros are allowed. A hierarchical model is a factorial model in which lower-order terms are always present. Thus, in a three-way table with classification variable names *A*, *B*, and *C*, the following models are all hierarchical models.

$$\begin{array}{llll} A & B & C & AB \\ A & B & C & AB \quad BC \\ A & C & AC & \\ A & B & C & AB \quad AC \quad BC \end{array}$$

Many other hierarchical models exist for the three-way table. Since all hierarchical models can be completely specified by the higher-order interactions (the lower-order interactions will always be present), no lower-order effects are included in model specification.

Corresponding to each hierarchical interaction is a marginal table. Iterations in **PRPFT** proceed by fitting marginal tables successively until the desired precision is achieved.

A structural zero is a cell in the table that, by design or otherwise, can have no observations, i.e., the count for the cell must be zero. Structural zeros are specified by setting the corresponding element in FIT to zero on input. Routine PRPFT is best suited for tables with no structural zeros and in which the initial estimates input in FIT are all 1. The user should be aware that the algorithm may take (much) longer to converge when this is not the case.

Sampling zeros are cells that are not structural zeros, but for which no count is observed. Routine PRPFT requires the absence of sampling zeros in all marginal tables that are fit. One common way method of achieving this is to add a constant, often 0.5, to each cell prior to fitting the table.

Comments

1. Workspace may be explicitly provided, if desired, by use of P2PFT/DP2PFT. The reference is:

   ```
   CALL P2PFT (NCLVAR, NCLVAL, TABLE, NEF, NVEF, INDEF, EPS, MAXIT, FIT, AMAR,
       INDEX, WK, IWK)
   ```

   The additional arguments are as follows.

   AMAR — Work vector with length equal to the sum from J = 1 to NEF of the product of the non-zero elements of NCLVAL(INDEF(I)) for I = 1 to NVEF(J).

   INDEX — Work vector of length NEF.

   WK — Work vector with length equal to the maximum over J = 1 to NEF of the product of the nonzero elements of NCLVAL(INDEF(I)), for I = 1 to NVEF(J).

   IWK — Work vector of length 2 * NCLVAR.

2. Informational errors

Type	Code	Description
3	11	The algorithm did not converge to the desired accuracy within MAXIT iterations.
4	12	A marginal total for an effect is zero. Since FIT indicates this is not a structural zero, the algorithm will not converge properly. One way to proceed is to add a constant to all cells in the table.

3. The cells of the vectors TABLE and FIT are sequenced so that the first variable cycles from 1 to NCLVAL(1), which is the slowest, the second variable cycles from 1 to NCLVAL(2), which is the next slowest, etc., up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) the fastest.

   Example: For NCLVAR = 3, NCLVAL(1) = 2, NCLVAL(2) = 3, and NCLVAL(3) = 2, the cells of table X(I, J, K) are entered into TABLE(1) through TABLE(12) in the following order. X(1, 1, 1), X(1, 1, 2), X(1, 2, 1), X(1, 2, 2), X(1, 3, 1), X(1, 3, 2), X(2, 1, 1), X(2, 1, 2), X(2, 2, 1), X(2, 2, 2), X(2, 3, 1), X(2, 3, 2). The elements of FIT are similarly sequenced.

4. **INDEF** is used to describe the marginal tables to be fit. For example, if **NCLVAR** = 3 and the first effect is to fit the marginal table for variables 1 and 3 and the second effect is to fit the marginal table for variable 2, then: **NEF** = 2, **NVEF**(1) = 2, and **NVEF**(2) = 1.

   Since the sum of the **NVEF**(I) is 3, then **INDEF** is a vector of length 3 with values. **INDEF** (1) = 1, **INDEF**(2) = 3, and **INDEF**(3) = 2.

5. Typically, **MAXIT** = 5 is sufficient. If **PRPFT** does not converge, try using double precision, increasing **MAXIT**, or using the values output in **FIT** as input for another call to **PRPFT**.

Example

The following example is taken from Bishop, Feinberg, and Holland (1975, page 87). The data are originally from Bartlett (1935). This example examines the survival of plants (factor *A* = factor 2) at different values for time of planting (factor *C* = factor 3) and length of cutting (factor *B* = factor 1). The sample size for each level of *B* and *C* is fixed at 240.

		B = **1**				***B*** = **2**	
		A				***A***	
		1	2			1	2
C	1	156	84	***C***	1	84	156
	2	107	133		2	31	209

The model to be fit is given by:

$$\ln\left(m_{ijk}\right) = \mu + \alpha_i + \beta_j + \alpha\beta_{ij} + \gamma_k + \alpha\gamma_{ik} + \beta\gamma_{jk}$$

where m_{ijk} is the cell expected value for levels *i*, *j*, and *k* of factors *A*, *B*, and *C*, respectively.

```
      USE PRPFT_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NCLVAR, NEF
      PARAMETER  (NCLVAR=3, NEF=3)
!
      INTEGER    INDEF(6), MAXIT, NCLVAL(NCLVAR), NOUT, NVEF(NEF)
      REAL       EPS, FIT(8), TABLE(8)
!
      DATA NCLVAL/2, 2, 2/, NVEF/2, 2, 2/
      DATA INDEF/1, 2, 1, 3, 2, 3/, EPS/0.0001/, MAXIT/15/
      DATA TABLE/156, 107, 84, 31, 84, 133, 156, 209/
      DATA FIT/8*1.0/
!
```

```
      CALL PRPFT (NCLVAL, TABLE, NVEF, INDEF, FIT, EPS=EPS, MAXIT=MAXIT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) FIT
99999 FORMAT (' FIT =', 8F7.1)
      END
```

Output

```
 FIT =  161.1  101.9   78.9   36.1   78.9  138.1  161.1  203.9
```

CTLLN

Computes model estimates and associated statistics for a hierarchical log-linear model.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Input)

TABLE — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) containing the entries in the cells of the table to be fit. (Input)
See Comment 3 for comments on the ordering of the elements of TABLE.

NVEF — Vector of length NEF containing the number of classification variables associated with each effect. (Input)

INDEF — Vector of length NVEF(1) + ... + NVEF(NEF) containing, in consecutive positions, the indices of the variables that are included in each effect. (Input)
The entries in INDEF are sequenced so that the first NVEF(1) elements contain the indices of the variables in effect 1, the next NVEF(2) elements of INDEF contain the indices of the variables in effect 2, etc. See Comment 4 for an example.

FIT — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) containing the model estimates of the cell frequencies. (Input/Output)
On input, FIT contains the initial estimates of the cell counts. Structural zeros in the model are specified by setting the corresponding element of FIT to 0.0. All other elements of FIT may be set to 1.0 if no other estimate of the expected cell counts is available. On output, FIT contains the fitted table. See Comment 3 for the ordering of the elements of FIT. If an element of FIT is positive but the corresponding element in TABLE is zero, then the element is called a sampling zero. Sampling zeros may effect the number of parameters that can be estimated, but they will not effect the degrees of freedom in chi-squared tests. See the Description section.

NCOEF — Number of regression coefficients in the model. (Output)

COEF — NCOEF by 4 matrix containing the estimated coefficients and associated statistics. (Output)
Dummy variables used in fitting the log-linear model are generated using the IDUMMY = 3 option of routine GRGLM (see Chapter 2, "Regression"). For this option, the *k*-th dummy variable for classification variable I is the (0, 1) indicator variable for the *k*-th level of the classification variable minus the (0, 1) indicator variable for the NCLVAL(I)-th level of the classification variable.

	Statistic
1	Coefficient estimate
2	Estimated standard error of the estimated coefficient
3	Asymptotic normal score for testing that the coefficient is zero
4	*p*-value associated with the normal score in column 3 (two-sided alternative).

COV — NCOEF by NCOEF covariance matrix for the estimated parameters. (Output)

RESID — NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR) by 4 matrix containing residual statistics for each cell in the table. (Output)

Column	Statistics
1	Signed square root of the contribution to chi-squared
2	Contribution to the likelihood ratio
3	Freeman-Tukey deviate
4	Residual difference

STAT — Vector of length 4 containing output statistics for the model. (Output)

I	STAT(I)
1	Log-likelihood.
2	Likelihood ratio statistic for testing the fit of the model.
3	Degrees of freedom in the likelihood ratio statistic. This statistic corrects for parameters that cannot be estimated because of sampling zeros.
4	*p*-value corresponding to the likelihood ratio statistic.

Optional Arguments

NCLVAR — Number of classification variables. (Input)

A variable specifying a margin in the table is a classification variable. The first classification variable is named *A*, the second classification variable is named *B*, etc.

Default: NCLVAR = size (NCLVAL,1).

NEF — Number of effects in the model. (Input)

A marginal table is implied by each effect in the model. Lower-order effects should not be included since their inclusion is automatic in the hierarchical models fit here (e.g., do not include effects *A* or *B* if effect *AB* is in the model).

Default: NEF = size (NVEF,1).

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum deviation between an observed and a fitted marginal total is less than **EPS**. **EPS** = 0.10 is a typical value.
Default: **EPS** = 0.10.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 15 is a typical value.
Default: **MAXIT** = 30

TOL — Tolerance used in determining linear dependence in **COV**. (Input)
TOL = 100.0 **AMACH**(4) is a common choice. See the documentation for routine **AMACH** in Reference Material.
Default: **TOL** = 1.19e-5 for single precision and 2.d–14 for double precision.

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	TABLE, FIT, RESID, COEF, COV, and STAT are printed.

LDCOEF — Leading dimension of **COEF** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOEF** = size (**COEF**,1).

LDCOV — Leading dimension of **COV** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOV** = size (**COV**,1).

LDRESI — Leading dimension of **RESID** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDRESI** = size (**RESID**,1).

FORTRAN 90 Interface

Generic: **CALL CTLLN** (**NCLVAL**, **TABLE**, **NVEF**, **INDEF**, **FIT**, **NCOEF**, **COEF**, **COV**, **RESID**, **STAT** [, ...])

Specific: The specific interface names are **S_CTLLN** and **D_CTLLN**.

FORTRAN 77 Interface

Single: CALL CTLLN (NCLVAR, NCLVAL, TABLE, NEF, NVEF, INDEF, EPS, MAXIT, TOL, IPRINT, FIT, NCOEF, COEF, LDCOEF, COV, LDCOV, RESID, LDRESI, STAT)

Double: The double precision name is DCTLLN.

Description

Routine CTLLN computes statistics of interest for a hierarchical model in a log-linear analysis of a multidimensional contingency table. Among the statistics computed are the expected cell values, cell residuals, the log-linear parameters and their estimated variances and covariances, the log-likelihood for the model (plus a constant), and a likelihood-ratio test of the model (versus the alternative that the cell probabilities are free to vary, subject only to the marginal constraints). In addition, CTLLN can print and label all statistics that it computes.

Routine PRPFT is used to find the maximum likelihood estimates of the expected cell counts (FIT). These expected values are then used as input to routine CTPAR in order to compute estimates of the parameters in the model and their estimated covariances.

The matrix RESID contains various residuals that may be used in analyzing the model. These residuals are discussed in detail by Bishop, Feinberg, and Holland (1975, pages 136-137), among others. Each is computed from the cell observed (o_i) and expected (fitted, f_i) values according to the following methods:

1. The signed square root of the contributions to X^2 are computed as

$$\left(o_i - f_i\right)\sqrt{f_i}$$

2. The contributions to the likelihood ratio (G^2) are computed as $2o_i \log(o_i/f_i)$

3. Freeman-Tukey deviates are computed as

$$\sqrt{o_i} + \sqrt{o_i + 1} - \sqrt{4f_i + 1}$$

4. The residual differences are computed as $o_i - f_i$

The log-likelihood STAT(1) is computed as

$$\sum_{i=1}^{n} -o_i \log\left(f_i\right)$$

where n is the number of cells in the table. The likelihood ratio statistic for testing the fit of the model is computed as

$$G^2 = \sum_{i=1}^{n} 2o_i \log\left(\frac{o_i}{f_i}\right)$$

which for large samples follows a chi-squared distribution.

The number of degrees of freedom in G^2 is computed as the number of cells in the table, excluding structural zeros, minus the number of parameters that could be estimated if there were no sampling zeros. When there are either structural or sampling zeros in the model, some parameters may not be estimable because they are infinite. Parameters that cannot be estimated due to structural zeros are not counted in the number of parameters estimated when computing the degrees of freedom for $\mathbf{X}^2$. Parameters that cannot be estimated because of sampling zeros are counted as estimated parameters when computing the degrees of freedom for $\mathbf{X}^2$.

To explain the calculation of degrees of freedom, note that extended maximum likelihood estimates may be written as

$$\hat{\beta} = \hat{\beta}_F + \rho\hat{\beta}_\infty$$

where

$$\hat{\beta},\ \hat{\beta}_F \text{ and } \rho\hat{\beta}_\infty$$

are coefficient vectors, and $\rho \to \infty$. Routine **CTLLN** estimates the finite portion of the estimates,

$$\hat{\beta}_F$$

The infinite portion,

$$\hat{\beta}_\infty$$

ensures that the fitted values for zero marginal cells corresponding to a term in the hierarchical model have estimated expectation of zero. Thus, **CTLLN** fits the finite portion of extended maximum likelihood estimates where the extension is to $\pm\infty$. Because the Hessian elements corresponding to infinite parameters are zero, the Hessian is computed from a reduced likelihood in which cells leading to infinite estimates have been eliminated. The user is referred to Clarkson and Jennrich (1991) for details.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2LLN/DC2LLN. The reference is:

   ```
   CALL C2LLN (NCLVAR, NCLVAL, TABLE, NEF, NVEF, INDEF, EPS, MAXIT, TOL, IPRINT,
      FIT, NCOEF, COEF, LDCOEF, COV, LDCOV, RESID, LDRESI, STAT, AMAR, INDEX,
      NCVEF, IXEF, IINDEF, IA, INDCL, CLVAL, REG, X, D, XMIN, XMAX, COVWK, WK, IWK)
   ```

   The additional arguments are as follows.

   AMAR — Vector of length equal to the sum over all effects in the model (J = 1 to NEF) of the length of the marginal table required for the effect. The length of each marginal table is computed as the product of the number of class values for each classification variable in the effect (the product of the nonzero elements of NCLVAL(INDEF(I)) where I ranges from K(J) through K(J)+ NVEF(J) - 1. Here, K(1) = 1 and K(J + 1) = K(J) + NVEF(J).)

   INDX — Vector of length NEF.

   NCVEF — Vector of length 2NCLVAR - 1.

   IXEF — Vector of length NCLVAR * $2^{NCLVAR-1}$.

   IINDEF — Vector of length NVEF(1) + … + NVEF(NEF).

   IA — Vector of length NCLVAR.

   INDCL — Vector of length NCLVAR.

   CLVAL — Vector of length NCLVAL(1) + … + NCLVAL(NCLVAR).

   REG — Vector of length NCOEF + 1.

   X — Vector of length NCOEF if there exists both structural and sampling zeros in TABLE; otherwise, it is of length NCLVAR.

   D — Vector of length NCOEF + 1.

   XMIN — Vector of length NCOEF.

   XMAX — Vector of length NCOEF.

   COVWK — Vector of length $NCOEF^2$ if there exists both structural and sampling zeros in TABLE. Otherwise, COVWK is not referenced and can be dimensioned of length one.

   WK — Vector of length max(*g*, NCOEF + 1) if IPRINT = 0; otherwise, WK is of length max(*g*, 6*m*, *n*) where *m* is the maximal element in NCLVAL, *n* is the length of TABLE, and *g* equals the maximum over all effects in the model (J = 1, NEF) of the length of the marginal table required for the effect. The length of the marginal table is computed as the product of the number of class values for each classification variable in the effect (the product of the nonzero elements of NCLVAL(INDEF(I)) where I ranges from K(J) through K(J) + NVEF(J) - 1, where K(1) = 1 and K(J + 1) = K(J) + NVEF(J)).

   IWK — Vector of length 2 * NCLVAR + *z* + 1 where *z* is the number of structural zeros in TABLE.

2. Informational errors

Type	Code	Description
3	1	The optimization algorithm did not converge to the desired accuracy within MAXIT iterations. Some of the estimated statistics may not be accurate.
3	5	The label for one or more of the tables exceeds the buffer limit.
3	11	The label for one or more effects exceeds the buffer limit.
4	2	LDCOEF or LDCOV is less than NCOEF.

3. The cells of the vectors TABLE and ZERO are sequenced so that the first variable cycles from 1 to NCLVAL(1) the slowest, the second variable cycles from 1 to NCLVAL(2) the next slowest, etc., up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) the fastest.

   Example: For NCLVAR = 3, NCLVAL(1) = 2, NCLVAL(2) = 3, and NCLVAL(3) = 2, the cells of table X(I, J, K) are entered into TABLE(1) through TABLE(12) in the following order: X(1, 1, 1), X(1, 1, 2), X(1, 2, 1), X(1, 2, 2), X(1, 3, 1), X(1, 3, 2), X(2, 1, 1), X(2, 1, 2), X(2, 2, 1), X(2, 2, 2), X(2, 3, 1), X(2, 3, 2). The elements of FIT are similarly sequenced.

4. INDEF is used to describe the marginal tables to be fit. For example, if NCLVAR = 3 and the first effect is to fit the marginal table for variables 1 and 3 and the second effect is to fit the marginal table for variable 2, then: NEF = 2, NVEF(1) = 2, and NVEF(2) = 1. Since the sum of the NVEF(I) is 3, then INDEF is a vector of length 3 with values: INDEF(1) = 1, INDEF(2) = 3, and INDEF(3) = 2.

Example

The example illustrates the use of CTLLN in a simple four-way table in which the first three factors have two levels, and the fourth factor has three levels. The data, taken from Lee (1977), involve brand preference in different situations.

```
      USE CTLLN_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCOEF, LDCOV, LDRESI, LTAB, MAXIT, NCLVAR
      REAL       EPS
      PARAMETER  (EPS=0.01, IPRINT=1, LDCOEF=10, LDCOV=10, LDRESI=24, &
                 LTAB=24, MAXIT=10, NCLVAR=4)
!
      INTEGER    INDEF(6), NCLVAL(NCLVAR), NCOEF, NEF, NVEF(3)
      REAL       COEF(LDCOEF,4), COV(LDCOV,LDCOV), FIT(LTAB), &
                 RESID(LDRESI,4), STAT(4), TABLE(LTAB)
!
      DATA TABLE/19, 57, 29, 63, 29, 49, 27, 53, 23, 47, 33, 66, 47, &
           55, 23, 50, 24, 37, 42, 68, 43, 52, 30, 42/
      DATA NEF/3/, NVEF/2, 2, 2/, INDEF/2, 4, 1, 4, 2, 3/
      DATA NCLVAL/3, 2, 2, 2/, FIT/24*1.0/
!
```

```
      CALL CTLLN (NCLVAL, TABLE, NVEF, INDEF, FIT, NCOEF, COEF, &
                  COV, RESID, STAT, EPS=EPS, MAXIT=MAXIT, IPRINT=IPRINT)
!
      END
```

Output

```
 Fitted Model: (B*D, A*D, B*C)

 Variable   Number of Levels
 1 A              3
 2 B              2
 3 C              2
 4 D              2

 Model Statistics
 Log-likelihood           3.7906
 Likelihood ratio          11.89
 Degrees of freedom         14.0
 P-value                  0.6154

                          Coefficient Statistics
                                     Standard   Asymptotic
                    Coefficient          Error  Z-statistic        P-value
  1 intercept            3.6827         0.0333       110.66         0.0000
  2  A(1)               -0.0591         0.0475        -1.24         0.2341
  3  A(2)                0.0278         0.0461         0.60         0.5562
  4  B                  -0.0166         0.0331        -0.50         0.6242
  5  C                  -0.0434         0.0319        -1.36         0.1943
  6  D                  -0.2783         0.0329        -8.45         0.0000
  7  A*D(1)             -0.1016         0.0475        -2.14         0.0506
  8  A*D(2)              0.0034         0.0461         0.07         0.9414
  9  B*C                -0.1438         0.0319        -4.51         0.0005
 10  B*D                -0.0684         0.0328        -2.09         0.0558

           -------------------------------
                Table 1: C = 1 D = 1
                 B = 1 by A (column)
                           1           2           3
  Observed             19.00       23.00       24.00
  Fit                  19.52       23.65       26.09
  Root chi-square      -0.12       -0.13       -0.41
  Likelihood           -1.03       -1.29       -4.02
  Freeman-Tukey        -0.06       -0.08       -0.37
  Residual             -0.52       -0.65       -2.09

                 B = 2 by A (column)
                           1           2           3
  Observed             29.00       47.00       43.00
  Fit                  30.85       37.37       41.23
  Root chi-square      -0.33        1.57        0.28
  Likelihood           -3.58       21.54        3.62
  Freeman-Tukey        -0.29        1.52        0.31
  Residual             -1.85        9.63        1.77

           -------------------------------
                Table 2: C = 1 D = 2
                 B = 1 by A (column)
```

```
                            1           2           3
Observed                57.00       47.00       37.00
Fit                     47.85       46.99       42.89
Root chi-square          1.32        0.00       -0.90
Likelihood              19.95        0.03      -10.93
Freeman-Tukey            1.29        0.04       -0.89
Residual                 9.15        0.01       -5.89

                  B = 2 by A (column)
                            1           2           3
Observed                49.00       55.00       52.00
Fit                     57.52       56.48       51.56
Root chi-square         -1.12       -0.20        0.06
Likelihood             -15.70       -2.92        0.89
Freeman-Tukey           -1.13       -0.16        0.10
Residual                -8.52       -1.48        0.44

             ------------------------------
                  Table 3: C = 2 D = 1
                   B = 1 by A (column)
                            1           2           3
Observed                29.00       33.00       42.00
Fit                     28.39       34.40       37.94
Root chi-square          0.11       -0.24        0.66
Likelihood               1.23       -2.73        8.53
Freeman-Tukey            0.16       -0.20        0.68
Residual                 0.61       -1.40        4.06

                  B = 2 by A (column)
                            1           2           3
Observed                27.00       23.00       30.00
Fit                     25.24       30.58       33.73
Root chi-square          0.35       -1.37       -0.64
Likelihood               3.64      -13.10       -7.04
Freeman-Tukey            0.39       -1.41       -0.61
Residual                 1.76       -7.58       -3.73

             ------------------------------
                  Table 4: C = 2 D = 2
                   B = 1 by A (column)
                            1           2           3
Observed                63.00       66.00       68.00
Fit                     69.58       68.32       62.37
Root chi-square         -0.79       -0.28        0.71
Likelihood             -12.51       -4.57       11.75
Freeman-Tukey           -0.78       -0.25        0.73
Residual                -6.58       -2.32        5.63

                  B = 2 by A (column)
                            1           2           3
Observed                53.00       50.00       42.00
Fit                     47.06       46.21       42.18
Root chi-square          0.87        0.56       -0.03
Likelihood              12.61        7.88       -0.36
Freeman-Tukey            0.87        0.58        0.01
Residual                 5.94        3.79       -0.18

                      Asymptotic Coefficient Covariance
                1           2           3           4           5
 1    1.1076E-03  9.7132E-05  -3.5887E-05  4.3244E-05  4.3786E-05
```

2		2.2562E-03	-1.1408E-03	-3.4043E-11	2.6829E-11
3			2.1232E-03	2.5675E-11	-5.1643E-11
4				1.0968E-03	1.4480E-04
5					1.0146E-03

	6	7	8	9	10
1	2.9815E-04	1.3065E-04	-1.6147E-05	1.4480E-04	7.6307E-05
2	1.3065E-04	7.2117E-04	-4.0976E-04	6.2343E-11	-1.0681E-11
3	-1.6147E-05	-4.0976E-04	5.7437E-04	-4.9217E-11	-2.3482E-11
4	7.6307E-05	1.2601E-11	-4.1730E-11	4.3786E-05	2.8917E-04
5	-1.4272E-11	-5.5301E-11	4.2801E-11	4.5231E-06	-4.6962E-11
6	1.0851E-03	9.7132E-05	-3.5887E-05	-4.9749E-11	3.0847E-05
7		2.2562E-03	-1.1408E-03	5.9300E-11	-1.0361E-10
8			2.1232E-03	-2.4481E-11	2.9160E-11
9				1.0146E-03	1.1201E-11
10					1.0743E-03

CTPAR

Computes model estimates and covariances in a fitted log-linear model.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Input)

NVEF — Vector of length NEF containing the number of classification variables associated with each effect. (Input)

INDEF — Vector of length NVEF(1) + … + NVEF(NEF) containing, in consecutive positions, the indices of the variables that are included in each effect. (Input)
The entries in INDEF are sequenced so that the first NVEF(1) elements contain the indices of the variables in effect 1, the next NVEF(2) elements of INDEF contain the indices of the variables in effect 2, etc. See Comment 4 for an example.

FIT — Vector of length NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR) containing the model estimates of the cell counts. (Input)
See Comment 3 for the ordering of the elements of FIT. To obtain a first iteration approximation to the optimal parameter values, the observed counts may be input in FIT, in which case a least-squares model is fit. In all cases, values of zero in FIT are assumed to correspond to structural zeros in the table. See the Description section for details.

NCOEF — Number of regression coefficients in the model. (Output)

COEF — NCOEF by 4 matrix containing the estimated coefficients and associated statistics. (Output)

	Statistic
1	Coefficient estimate
2	Estimated standard error of the estimated coefficient
3	Asymptotic normal score for testing that the coefficient is zero
4	*p*-value associated with the normal score in column 3 (two-sided alternative)

COV — NCOEF by NCOEF covariance matrix of the estimated coefficients. (Output)

Optional Arguments

NCLVAR — Number of classification variables. (Input)
A variable specifying a margin in the table is a classification variable. The first classification variable is named *A*, the second classification variable is named *B*, etc.
Default: NCLVAR = size (NCLVAL,1).

NEF — Number of effects in the model. (Input)
A marginal table is implied by each effect in the model. Lower-order effects should not be included since their inclusion is automatic in the hierarchical models fit here (e.g., do not include effects *A* or *B* if effect *AB* is in the model).
Default: NEF = size (NVEF,1).

TOL — Tolerance used in determining linear dependence in COV. (Input)
TOL = 100.0 * AMACH(4) is a common choice. See the documentation for routine AMACH in Reference Material.
Default: TOL = 1.19e-5 for single precision and 2.d –14 for double precision.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing of COEF and COV is performed.
2	COEF, COV, and FIT are printed.

In the printing, A * B(2) denotes the second variable in the *AB* interaction effect.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

FORTRAN 90 Interface

Generic: CALL CTPAR (NCLVAL, NVEF, INDEF, FIT, NCOEF, COEF, COV [, ...])
Specific: The specific interface names are S_CTPAR and D_CTPAR.

FORTRAN 77 Interface

Single: CALL CTPAR (NCLVAR, NCLVAL, NEF, NVEF, INDEF, FIT, TOL, IPRINT, NCOEF, COEF, LDCOEF, COV, LDCOV)

Double: The double precision name is DCTPAR.

Description

Routine CTPAR computes estimates of parameters and associated variances and covariances in hierarchical log-linear models. A weighted least-squares algorithm is used.

A hierarchical analysis of variance model is a factorial analysis of variance model in which a lower-order effect is included in a model whenever a higher-order effect containing it is in the model. Thus, if the effect *ADF* is in the model, then effects *A*, *D*, *F*, *AD*, *AF*, and *DF* are automatically in the model.

Input to CTPAR may be either the expected table values for the given hierarchical model as output, for example, by routine PRPFT, or the observed table values. When the fitted values are input, the estimates computed are the maximum likelihood estimates. When observed values are input, weighted least-squares estimates of the parameters in the log-linear model are computed. (Least-squares estimates and maximum likelihood estimates can also be computed via routines CTWLS and CTGLM, respectively.)

When an expected count (as input in FIT) is zero, the cell is taken to be a structural zero. Such cells are not included in the weighted least-squares analysis. Estimates corresponding to structural zeros are set to the missing value indicator (NaN). To avoid this (and to determine the total degrees of freedom for each effect), add a positive constant such as 0.5 to each of the observed cell counts of zero, the "sampling" zeros. When structural zeros are present in the data the estimates may be written as

$$\hat{\beta} = \hat{\beta}_o + \rho\hat{\beta}_I$$

where

$$\hat{\beta},\ \hat{\beta}_0,\ \text{and}\ \hat{\beta}_I$$

are vectors, and $\rho \to \infty$. Routine CTPAR estimates the finite portion of the estimate, $\hat{\beta}_0$ The infinite portion, $\hat{\beta}_I$ ensures that the fitted values for cells corresponding to structural zeros are zero (sampling zeros are considered to be structural zeros in CTPAR). If there are no structural zeros

$$\hat{\beta}_I = 0$$

Let f_i denote the i-th element of the vector **FIT**. The asymptotic variance-covariance matrix of the cell counts is estimated by a diagonal matrix $S = \text{diag}(f)$ where $\text{diag}(f)$ denotes the diagonal matrix in which $s_{ij} = 0$ for $i \neq j$ and $s_{ii} = f_i$ along the diagonal. If X denotes the design matrix for the hierarchical model (with rows in X corresponding to structural zeros omitted), and $y_i = \log f_i$, then the weighted least-squares estimates are

$$\hat{\beta}_o = \left(X^T S^{-1} X \right)^{-1} X^T S^{-1} y$$

and the estimated variance-covariance matrix is

$$(X^{\mathrm{T}}\, S^{-1} X)^{-1}$$

(see Grizzle, Starmer, and Koch [1969]).

If main effect *A* has, for example, four levels, then the design matrix *X* contains three dummy variables corresponding to this effect. Main effect dummy variables are generated as follows: For an observation f_i corresponding to level j of the effect, if $j < 3$, then the j-th dummy variable is set to 1 with the remaining dummy variables set to 0. If $j = 4$, then all three dummy variables are set to -1. Dummy variables for interactions are generated as the product of the corresponding dummy variables in the usual manner with the smallest index in the specification of the interaction varying fastest. The indices of the classification variables for each effect are always sorted from smallest to largest when computing the columns of *X*.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2PAR/DC2PAR**. The reference is:

   ```
   CALL C2PAR (NCLVAR, NCLVAL, NEF, NVEF, INDEF, FIT, TOL, IPRINT, NCOEF, COEF,
       LDCOEF, COV, LDCOV, IRANK, NCVEF, IXEF, IINDEF, IA, INDCL, CLVAL, REG, X, D,
       XMIN, XMAX, WK)
   ```

   The additional arguments are as follows:

   IRANK — Rank of **COV**.

   NCVEF — Vector of length 2^{NCLVAR} - 1.

   IXEF — Vector of length NCLVAR * $2^{\text{NCLVAR}-1}$.

   IINDEF — Vector of length NVEF(1) + … + NVEF(NEF).

   IA — Vector of length NCLVAR.

   INDCL — Vector of length NCLVAR.

   CLVAL — Vector of length NCLVAL(1) + … + NCLVAL(NCLVAR).

   REG — Vector of length NCOEF + 1.

   X — Vector of length NCLVAR.

D — Vector of length NCOEF.

XMIN — Vector of length NCOEF.

XMAX — Vector of length NCOEF.

WK — Vector of length NCOEF + 1 if IPRINT ≠ 2. Otherwise, its length is the maximum of NCOEF + 1 and the product of the two largest elements of NCLVAL.

2. Informational errors

Type	Code	Description
3	5	The label for one or more of the tables exceeds the buffer limit.
3	11	The label for one or more effects exceeds the buffer limit.
4	1	LDCOEF or LDCOV is less than NCOEF.

3. The cells of the vector FIT are sequenced so that the first variable cycles from 1 to NCLVAL(1) the slowest, the second variable cycles from 1 to NCLVAL(2) the next slowest, etc., up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) the fastest.

   Example: For NCLVAR = 3, NCLVAL(1) = 2, NCLVAL(2) = 3, and NCLVAL(3) = 2, the cells of table X(I, J, K) are entered into FIT(1) through FIT(12) in the following order: X(1, 1, 1), X(1, 1, 2), X(1, 2, 1), X(1, 2, 2), X(1, 3, 1), X(1, 3, 2), X(2, 1, 1), X(2, 1, 2), X(2, 2, 1), X(2, 2, 2), X(2, 3, 1), X(2, 3, 2).

4. INDEF is used to describe the marginal tables to be fit. For example, if NCLVAR = 3 and the first effect is to fit the marginal table for variables 1 and 3 and the second effect is to fit the marginal table for variable 2, then: NEF = 2, NVEF(1) = 2, and NVEF(2) = 1. Since the sum of the NVEF(I) is 3, then INDEF is a vector of length 3 with values: INDEF(1) = 1, INDEF(2) = 3, and INDEF(3) = 2.

Example

The example illustrates the use of CTPAR in a simple four-way table in which the first three factors have two levels, and the fourth factor has three levels. The data, which is taken from Lee (1977), involve the brand preference in different situations.

```
      USE PRPFT_INT
      USE CTPAR_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCOEF, LDCOV, LTAB, NCLVAR
      PARAMETER  (IPRINT=2, LDCOEF=13, LDCOV=13, LTAB=24, NCLVAR=4)
!
      INTEGER    INDEF(6), NCLVAL(NCLVAR), NCOEF, NVEF(3)
      REAL       COEF(LDCOEF,4), COV(LDCOV,LDCOV), FIT(LTAB), &
                 TABLE(LTAB)
!
      DATA TABLE/19, 57, 29, 63, 29, 49, 27, 53, 23, 47, 33, 66, 47, &
          55, 23, 50, 24, 37, 42, 68, 43, 52, 30, 42/
```

```
      DATA NVEF/2, 2, 2/, INDEF/2, 4, 1, 4, 2, 3/
      DATA NCLVAL/3, 2, 2, 2/, FIT/24*1.0/
!
      CALL PRPFT (NCLVAL, TABLE, NVEF, INDEF, FIT)
!
      CALL CTPAR (NCLVAL, NVEF, INDEF, FIT, NCOEF, COEF, COV, IPRINT=IPRINT)
!
      END
```

Output

```
Variable   Number of Levels
1 A              3
2 B              2
3 C              2
4 D              2

          ----------
      Table 1: B = 1 C = 1
     D (row) by A (column)
            1       2       3
 1   19.52   23.65   26.09
 2   47.85   46.99   42.89

          ----------
      Table 2: B = 1 C = 2
     D (row) by A (column)
            1       2       3
 1   28.39   34.40   37.94
 2   69.58   68.32   62.37

          ----------
      Table 3: B = 2 C = 1
     D (row) by A (column)
            1       2       3
 1   30.85   37.37   41.23
 2   57.52   56.48   51.56

          ----------
      Table 4: B = 2 C = 2
     D (row) by A (column)
            1       2       3
 1   25.24   30.58   33.73
 2   47.06   46.21   42.18

                        Coefficient Statistics
                                    Standard   Asymptotic
                   Coefficient         Error  Z-statistic        P-value
  1 intercept           3.6827        0.0333       110.66         0.0000
  2  A(1)              -0.0591        0.0475        -1.24         0.2341
  3  A(2)               0.0278        0.0461         0.60         0.5562
  4  B                 -0.0166        0.0331        -0.50         0.6242
  5  C                 -0.0434        0.0319        -1.36         0.1943
  6  D                 -0.2783        0.0329        -8.45         0.0000
  7  A*D(1)            -0.1016        0.0475        -2.14         0.0506
  8  A*D(2)             0.0034        0.0461         0.07         0.9414
  9  B*C               -0.1438        0.0319        -4.51         0.0005
 10  B*D               -0.0684        0.0328        -2.09         0.0558
```

Asymptotic Coefficient Covariance

	1	2	3	4	5
1	1.1076E-03	9.7132E-05	-3.5887E-05	4.3244E-05	4.3786E-05
2		2.2562E-03	-1.1408E-03	-3.4043E-11	2.6829E-11
3			2.1232E-03	2.5675E-11	-5.1643E-11
4				1.0968E-03	1.4480E-04
5					1.0146E-03

	6	7	8	9	10
1	2.9815E-04	1.3065E-04	-1.6147E-05	1.4480E-04	7.6307E-05
2	1.3065E-04	7.2117E-04	-4.0976E-04	6.2343E-11	-1.0681E-11
3	-1.6147E-05	-4.0976E-04	5.7437E-04	-4.9217E-11	-2.3482E-11
4	7.6307E-05	1.2601E-11	-4.1730E-11	4.3786E-05	2.8917E-04
5	-1.4272E-11	-5.5301E-11	4.2801E-11	4.5231E-06	-4.6962E-11
6	1.0851E-03	9.7132E-05	-3.5887E-05	-4.9749E-11	3.0847E-05
7		2.2562E-03	-1.1408E-03	5.9300E-11	-1.0361E-10
8			2.1232E-03	-2.4481E-11	2.9160E-11
9				1.0146E-03	1.1201E-11
10					1.0743E-03

CTASC

Computes partial association statistics for log-linear models in a multidimensional contingency table.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Input)

TABLE — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) containing the entries in the cells of the table to be fit. (Input)
See Comment 3 for comments on the ordering of the elements of TABLE.

ZERO — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) indicating structural zeros in TABLE. (Input)
ZERO has the same structure as TABLE. Structural zeros in the TABLE are specified by setting the corresponding element of ZERO to 0.0. All other elements of zero must be positive. If structural zeros do not exist in TABLE, TABLE and ZERO can share the same storage locations. See Comment 3 for the ordering of the elements of ZERO.

ASSOC — 2NCLVAR - 1 by 4 matrix containing the partial association statistics for each effect in the model. (Output)

Col	Statistic
1	Likelihood ratio partial association chi-squared for testing that all parameters in the effect are zero against a model containing all interactions of the same order
2	Degrees of freedom in chi-squared in columns 1 and 4
3	*p*-value for the chi-squared statistic in column 1
4	Number of zeros (structural and sampling) in the marginal table of the effect

The rows of ASSOC are ordered with main effects first, followed by two-way interactions, followed by the three-way interactions, etc., until the last row, which contains the single NCLVAR-way interaction. Thus, if there are 3 classification variables, there would be 8 rows in ASSOC and the rows would contain the *A*, *B*, *C*, *AB*, *AC*, *BC*, and the *ABC* effects where *A* represents the first (in INDCL) classification variable, *B* represents the second classification variable, etc.

CHIHI — NCLVAR by 5 matrix containing chi-squared statistics testing that all *k* and higher interactions are zero where *k* = 1, 2, ..., NCLVAR. (Output)
In the following, *k* is the row number of the statistic where the row numbers are 1, 2, ..., NCLVAR.

Col	Statistic
1	Likelihood ratio chi-squared statistic for testing that all interactions higher than *k* are zero against a model including all interactions of order *k*
2	*p*-value for the chi-squared value in column 1
3	Degrees of freedom for chi-squared in columns 1 and 4
4	Pearson chi-squared corresponding to column 1
5	*p*-value for the chi-squared value in column 4

CHISIM — NCLVAR by 5 matrix containing chi-squared statistics for testing that all *k*-factor interactions are simultaneously zero where *k* = 1, …, NCLVAR. (Output)
In the following, *k* is the row number of the statistic where the row numbers are 1, 2, …, NCLVAR.

Col	Statistic
1	Likelihood ratio chi-squared statistic for testing that all *k*-factor interactions are all simultaneously zero given the model in which all *k*-way interactions are present
2	*p*-value for the chi-squared value in column 1
3	Degrees of freedom for chi-squared in columns 1 and 4
4	Pearson chi-squared corresponding to column 1
5	*p*-value for the chi-squared value in column 4

Optional Arguments

NCLVAR — Number of classification variables. (Input)
A variable specifying a margin in the table is a classification variable. The first classification variable is named *A*, the second classification variable is named *B*, etc.
Default: NCLVAR = size (NCLVAL,1).

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum deviation between an observed and a fitted marginal total is less than EPS. EPS = 0.10 is a typical value.
Default: EPS = 0.10.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 15 is a typical value. When there are structural zeros a larger value, say MAXIT = 100, should be used.
Default: MAXIT = 100.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing of ASSOC, CHIHI, and CHISIM is performed.
2	ASSOC, CHIHI, CHISIM, and TABLE are printed.

LDASSO — Leading dimension of ASSOC exactly as specified in the dimension statement in the calling program. (Input)
Default: LDASSO = size (ASSOC,1).

LDCHIH — Leading dimension of CHIHI exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCHIH = size (CHIHI,1).

LDCHIS — Leading dimension of CHISIM exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCHIS = size (CHISIM,1).

FORTRAN 90 Interface

Generic: CALL CTASC (NCLVAL, TABLE, ZERO, ASSOC, CHIHI, CHISIM [, ...])
Specific: The specific interface names are S_CTASC and D_CTASC.

FORTRAN 77 Interface

Single: CALL CTASC (NCLVAR, NCLVAL, TABLE, ZERO, EPS, MAXIT, IPRINT, ASSOC, LDASSO, CHIHI, LDCHIH, CHISIM, LDCHIS)
Double: The double precision name is DCTASC.

Description

Routine CTASC computes likelihood-ratio and Pearson X^2 tests of partial-association for each effect in a hierarchical log-linear model. Also computed are likelihood ratio and Pearson chi-squared tests that all interactions above a given level are simultaneously zero. All of these tests are asymptotic in nature. All models are hierarchical so that all lower order interactions that may be composed from a higher order effect in the model are automatically included in the model. All models are fit via the iterative proportional fitting algorithm, which is implemented in routine PRPFT. The algorithm proceeds as follows:

1. The hierarchical model including all *k*-factor interactions is fit with *k* = 0, ..., *m* and *m* = **NCLVAR**. The *k* = 0 model corresponds to a constant probability in each cell in the table while the *k* = *m* model is the full model. For each value of *k*, the likelihood ratio chi-squared statistic for testing that all interactions not included in the fitted model are all simultaneously zero (against the alternative that this is not the case) is computed as

$$2\sum_i o_i \ \ln(o_i / f_i)$$

where o_i is the observed count in the *i*-th cell, f_i is the fitted count for the given model, and the summation is over all cells in the table. Also computed (for comparison, the two statistics are asymptotically equivalent) is the usual Pearson chi-squared statistic,

$$\sum_i (o_i - f_i)^2 / f_i$$

2. Let g_i = **NCLVAL**(*i*), and let

$$t = \prod_{i=1}^{m} g_i$$

and assume that there are no structural zeros in the table. Then, the number of degrees of freedom in the chi-squared statistic for testing that all k-order interactions are simultaneously zero is the sum over all k-th order interaction effects of the degrees of freedom for the effect. In the no structural zero case, the degrees of freedom for an effect may be computed as

$$\prod_j (g_j - 1)$$

where *j* indexes the factors in the effect. Denote the sum of these degrees of freedom at level *k* by s_k, and let $s_0 = 1$. Then, the degrees of freedom in the *k*-th test is given by s_k.

When more than one structural zero is present, the degrees of freedom in the chi-squared statistics are computed by fitting a least-squares model for the full hierarchical model in which all interactions are included. Routine **RGIVN** (see Chapter 2, "Regression") is used in fitting the model. Cells with sampling (as opposed to structural) zeros are included (but only when degrees of freedom are computed) by using a cell count of 0.5. Observations corresponding to structural zeros are not included. (Note that a structural zero is a model restriction that requires that the estimated count for a cell be zero. A sampling zero occurs by chance.) The degrees of freedom for each effect are found by summing over the estimated parameters for the effect. Parameters that are linearly related to previous parameters in the model (as determined through **RGIVN** via input argument TOL where TOL is

100 * AMACH(4) are not estimated. When there is only one structural zero, degrees of freedom are computed as if there were no structural zeros except for the highest level interaction term, which is given one fewer degree of freedom.

Chi-squared statistics for testing that all effects at a given level *k* are simultaneously zero (given a hierarchical model in which all effects above level *k* are absent) are computed as the difference between the chi-squared statistics testing that all *k* and higher interactions are zero and that of *k* + 1. That is, for J = 1 and 4, and I = 1, 2, ..., NCLVAR - 1, then CHISIM(I, J) = CHIHI(I, J) - CHIHI(I + 1, J), and CHISIM(NCLVAR, J) = CHIHI(NCLVAR, J).

3. For each effect, a "partial association" likelihood ratio chi-squared statistic may be used to test the hypothesis that all parameters in the effect are simultaneously zero, given a model in which all interactions at the same level (or lower) are present, and all higher level interactions are absent. The degrees of freedom for the effect are computed as in Step 2.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2ASC/DC2ASC. The reference is:

```
CALL C2ASC (NCLVAR, NCLVAL, TABLE, ZERO, EPS, MAXIT, IPRINT, ASSOC, LDASSO,
    CHIHI, LDCHIH, CHISIM, LDCHIS, FITWK, NCVEF, IXEF, AMAR, INDX, WK, IWK,
    COVWK)
```

The additional arguments are as follows:

FITWK — Work vector of length 3 * NCLVAL(1) * ... * NCLVAL(NCLVAR).

NCVEF — Work vector of length $2^{\text{NCLVAR}} - 1$.

IXEF — Work vector of length NCLVAR * $2^{(\text{NCLVAR}-1)}$

AMAR — Work vector of length *n*. In defining *n*, let *q*(*k*) be the sum of the sizes of all possible marginal tables with *k* effects. For example, *q*(2) is the sum over all possible two-way interactions I and J of NCLVAL(I) * NCLVAL(J) and *q*(NCLVAR) is the product NCLVAL(1) * ... * NCLVAL(NCLVAR). Then, *n* = max(*q*(*k*)), *k* = 1, ..., NCLVAR.

INDX — Work vector of length *m* where *m* is the maximum number of interactions at any level. That is, *m* = max(BINOM(NCLVAR, I)), I = 1, ..., NCLVAR, where BINOM(NCLVAR, I) is the binomial coefficient (see routine BINOM (IMSL MATH/LIBRARY Special Functions)).

WK — Work vector of length 3 * NCLVAL(1) * ... * NCLVAL(NCLVAR) if there exists more than one structural zero in TABLE, and of length NCLVAL(1) * ... * NCLVAL(NCLVAR) otherwise.

IWK — Work vector of length 2 * NCLVAR.

COVWK — Work vector of length (NCLVAL(1) * ... * NCLVAL(NCLVAR))^2 if there exists more than one structural zero in TABLE. Otherwise, COVWK is not referenced and can be dimensioned of length one in the calling program. On output, COVWK contains the upper triangular matrix containing the *R* matrix from a *QR* decomposition of the matrix of regressors for the full log-linear model.

2. Informational errors

Type	Code	Description
3	1	The optimization algorithm did not converge to the desired accuracy, EPS, within MAXIT iterations.
3	5	The label for one or more of the tables exceeds the buffer limit.
3	11	The label for one or more effects exceeds the buffer limit.

3. The cells of the vectors TABLE and ZERO are sequenced so that the first variable cycles from 1 to NCLVAL(1) the slowest, the second variable cycles from 1 to NCLVAL(2) the next slowest, etc., up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) the fastest.

   Example: For NCLVAR = 3, NCLVAL(1) = 2, NCLVAL(2) = 3, and NCLVAL(3) = 2, the cells of table X(I, J, K) are entered into TABLE(1) through TABLE(12) in the following order: X(1, 1, 1), X(1, 1, 2), X(1, 2, 1), X(1, 2, 2), X(1, 3, 1), X(1, 3, 2), X(2, 1, 1), X(2, 1, 2), X(2, 2, 1), X(2, 2, 2), X(2, 3, 1), X(2, 3, 2). The elements of FIT are similarly sequenced.

Programming Notes

1. When sampling zeros are present, the likelihood ratio test statistics may not follow the appropriate chi-squared distribution closely. A common (but not necessarily the best) practice in this case is to add a small positive constant, often 0.5, to each cell in the table. This addition is easily accomplished via routine SADD (IMSL MATH/LIBRARY). The addition of such a constant should not affect the computed degrees of freedom.

2. When marginal totals of zero are obtained, the optimization algorithm may be slow to converge. In this case, increase the value of argument MAXIT.

Example

The following example illustrates the use of CTASC for model building in a four-way table involving brand preference. The first three factors each have 2 levels, while the fourth factor has 3 levels. The data are originally from Lee (1977) and are printed in the output. A model with two-way interaction effects AD, BC, and BD looks promising.

```
          USE CTASC_INT

          IMPLICIT   NONE
          INTEGER    IPRINT, LDASSO, LDCHIH, LDCHIS, LTAB
          REAL       EPS
          PARAMETER  (EPS=0.01, IPRINT=2, LDASSO=15, LDCHIH=4, LDCHIS=4, &
                     LTAB=24)
    !
```

```
      INTEGER    NCLVAL(4)
      REAL       ASSOC(LDASSO,4), CHIHI(LDCHIH,5), CHISIM(LDCHIS,5), &
                 TABLE(LTAB)
!
      DATA TABLE/19, 57, 29, 63, 29, 49, 27, 53, 23, 47, 33, 66, 47, &
          55, 23, 50, 24, 37, 42, 68, 43, 52, 30, 42/
      DATA NCLVAL/3, 2, 2, 2/
!
      CALL CTASC (NCLVAL, TABLE, TABLE, ASSOC, CHIHI, CHISIM, &
                  EPS=EPS, IPRINT=IPRINT)
!
      END
```

Output

```
 Variable   Number of Levels
 1 A              3
 2 B              2
 3 C              2
 4 D              2

          ----------
    Table 1: B = 1 C = 1
   D (row) by A (column)
          1       2       3
 1   19.00   23.00   24.00
 2   57.00   47.00   37.00

          ----------
    Table 2: B = 1 C = 2
   D (row) by A (column)
          1       2       3
 1   29.00   33.00   42.00
 2   63.00   66.00   68.00

          ----------
    Table 3: B = 2 C = 1
   D (row) by A (column)
          1       2       3
 1   29.00   47.00   43.00
 2   49.00   55.00   52.00

          ----------
    Table 4: B = 2 C = 2
   D (row) by A (column)
          1       2       3
 1   27.00   23.00   30.00
 2   53.00   50.00   42.00

                  Partial Association Statistics
 Omitted                        Degrees of                 Marginal
 Effect           Chi-Square     Freedom       P-value     Zeros
 A                      0.50         2.0        0.7782       0.0
 B                      0.06         1.0        0.8010       0.0
 C                      1.92         1.0        0.1657       0.0
 D                     73.21         1.0        0.0000       0.0
 A*B                    0.22         2.0        0.8978       0.0
 A*C                    1.01         2.0        0.6050       0.0
```

A*D	6.10	2.0	0.0475	0.0
B*C	19.89	1.0	0.0000	0.0
B*D	3.74	1.0	0.0532	0.0
C*D	0.74	1.0	0.3898	0.0
A*B*C	4.57	2.0	0.1017	0.0
A*B*D	0.16	2.0	0.9223	0.0
A*C*D	1.38	2.0	0.5022	0.0
B*C*D	2.22	1.0	0.1361	0.0
A*B*C*D	0.74	2.0	0.6917	0.0

Chi-square statistics for testing that all k and higher interactions are zero.

k	Likelihood Ratio	P-Value	Degrees of Freedom	Pearson	P-Value
1	118.63	0.0000	23.0	115.71	0.0000
2	42.93	0.0008	18.0	43.90	0.0006
3	9.85	0.3631	9.0	9.87	0.3611
4	0.74	0.6917	2.0	0.74	0.6915

Chi-square statistics for testing that all k-factor interactions are simultaneously zero.

k	Likelihood Ratio	P-Value	Degrees of Freedom	Pearson	P-Value
1	75.70	0.0000	5.0	71.81	0.0000
2	33.08	0.0001	9.0	34.03	0.0001
3	9.11	0.2449	7.0	9.13	0.2433
4	0.74	0.6917	2.0	0.74	0.6915

CTSTP

Builds hierarchical log-linear models using forward selection, backward selection, or stepwise selection.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels or categories of the *i*-th classification variable. (Input)

TABLE — Vector of length NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR) containing the entries in the cells of the table to be fit. (Input)
See Comment 3 for comments on the ordering of the elements of TABLE.

ISTEP — Stepping option. (Input)

ISTEP	Action
-1	An attempt is made to remove an effect from the model (a backward step). An effect is removed if it has the largest *p*-value among all effects considered for removal with *p*-value exceeding POUT.
0	A backward step is attempted. If a variable is not removed, a forward step is attempted. This is a stepwise step.
1	An attempt is made to add an effect to the model (a forward step). An effect is added if it has the smallest *p*-value among all effects with *p*-value less than PIN.

NSTEP — Step length option. (Input)
For nonnegative NSTEP, NSTEP steps are taken. Less than NSTEPS are taken if no effect that can enter or leave the model meets the PIN or POUT criterion. Use NSTEP = -1 to indicate that stepping is to continue until no effect meets the PIN or POUT criterion to enter or leave the model.

NFORCE — The number of initial effects in the model that must be included in any model considered. (Input)
For NFORCE = *k*, the first *k* effects specified by NEF, NVEF, and INDEF will be included in all models considered.

NEF — Number of effects in the model. (Input/Output)
A marginal table is implied by each effect in the model. Lower order effects should not be included in the model specification since their inclusion is automatic (e.g., do not include effects *A* or *B* if effect *AB* is in the model). On input, NEF gives the number of effects in the initial model. On output, NEF gives the number of effects in the final model.

NVEF — Vector of length MAXNVF containing the number of classification variables associated with each effect. (Input/Output)
On input, NVEF contains the number of classification variables for each effect in the initial model. The final values are returned on output.

INDEF — Vector of length MAXIND containing, in consecutive positions, the indices of the variables that are included in each effect. (Input/Output)
The entries in INDEF are sequenced so that the first NVEF(1) elements contain the indices of the variables in effect 1, the next NVEF(2) elements of INDEF contain the indices of the variables in effect 2, etc. Each element of INDEF must be greater than zero. See Comment 4 for an example.

FIT — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) containing the model estimates of the cell counts. (Input/Output)
On input, FIT contains the initial estimates of the cell counts. Structural zeros in the model are specified by setting the corresponding element of FIT to 0.0. All other elements of FIT may be set to 1.0 if no other estimate of the expected cell counts is available. On output, FIT contains the fitted table. See Comment 3 for the ordering of the elements of FIT. If an element of FIT is positive but the corresponding element in TABLE is zero, the element is called a sampling zero. Sampling zeros may effect the number of parameters that can be estimated, but they will not affect the degrees of freedom in chi-squared tests. See the Description section of the manual document.

STAT — Vector of length 3 containing some output statistics for the final model fit during this invocation. (Output)

I	STAT(I)
1	Asymptotic chi-squared statistic based upon likelihood ratios for testing that the current model fits the observed data.
2	Degrees of freedom in chi-squared. This is the number of cells in the table minus the number of structural zeros minus the degrees of freedom for the model.
3	Probability of a greater chi-squared.

IEND — Completion indicator. (Output)

IEND	Meaning
0	Additional steps may be possible.
1	No additional steps are possible for the values of PIN and POUT.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of CTSTP for this table. If there are sampling zeros, set up for computing the degrees of freedom for each effect. Perform NSTEP steps (if ISTEP, POUT, and PIN allow it) and then release all workspace.
1	This is the first invocation, and additional calls to CTSTP will be made. Set up for computing the degrees of freedom for each effect and then perform NSTEP steps (if ISTEP, POUT, and PIN allow it).
2	This is an intermediate invocation of CTSTP. Perform NSTEP steps (if ISTEP, POUT, and PIN allow it).
3	This is the final invocation of this routine. Perform NSTEP steps (if ISTEP, POUT, and PIN allow it). Release all workspace.

NCLVAR — Number of classification variables. (Input)

A variable specifying a margin in the table is a classification variable. The first classification variable is named *A*, the second classification variable is named *B*, etc.

Default: NCLVAR = size (NCLVAL,1).

PIN — Largest *p*-value for entering variables. (Input)

Variables with *p*-values less than PIN may enter the model. The choice 0.05 is common.

Default: PIN = .05.

POUT — Smallest *p*-value for removing variables. (Input)

Variables with *p*-values greater than POUT may leave the model. POUT must be greater than or equal to PIN. The choice 0.10 is common.

Default: POUT = .10.

IPRINT — Printing option. (Input)

Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing of the initial and final model summary statistics and step summaries.
2	Printing of the input table is performed followed by printing of the initial and final model summary statistics and of the step summaries.

MAXNVF — The maximum length of NVEF as specified in the dimension statement in the calling program. (Input)

If the required length of NVEF becomes greater than MAXNVF, a type 4 error message is issued and the final model chosen is returned in NEF, NVEF, and INDEF. See Comment 2.

Default: MAXNVF = size (NVEF,1).

MAXIND — The maximum possible length of **INDEF** as specified in the dimension statement in the calling program. (Input)
If the required length of **INDEF** becomes greater than **MAXIND**, a type 4 error message is issued and the final model chosen is returned in **NEF**, **NVEF**, and **INDEF**. See Comment 2.
Default: **MAXIND** = size (**INDEF**,1).

FORTRAN 90 Interface

Generic: **CALL CTSTP** (**NCLVAL**, **TABLE**, **ISTEP**, **NSTEP**, **NFORCE**, **NEF**, **NVEF**, **INDEF**, **FIT**, **STAT**, **IEND** [, ...])

Specific: The specific interface names are **S_CTSTP** and **D_CTSTP**.

FORTRAN 77 Interface

Single: **CALL CTSTP** (**IDO**, **NCLVAR**, **NCLVAL**, **TABLE**, **PIN**, **POUT**, **ISTEP**, **NSTEP**, **NFORCE**, **IPRINT**, **NEF**, **NVEF**, **MAXNVF**, **INDEF**, **MAXIND**, **FIT**, **STAT**, **IEND**)

Double: The double precision name is **DCTSTP**.

Description

Routine **CTSTP** performs stepwise model building in hierarchical log-linear models. **CTSTP** handles structural and sampling zeros, and allows "downward," "upward," or "stepwise" stepping. For **NFORCE** > 0, the leading **NFORCE** effects in the initial model specified in **NEF**, **NVEF**, and **INDEF** are forced to remain in the model. A variable number (**NSTEP**) of steps from the input model are performed during a single invocation of **CTSTP**. Printing of the input table and intermediate results is performed if requested.

In hierarchical models, lower order effects are automatically included whenever a higher order effect containing the lower order effect is in the model. That is, the model (*AB*) automatically includes the mean and the main effects *A* and B, and the model (*AB*, *ACD*) automatically includes the lower order effects *A*, *B*, *C*, *D*, *AC*, *AD*, and *CD*.

The algorithm proceeds through the following steps during a single invocation when **IDO** = 0. For **IDO** > 0, these steps are still followed, but they may require more than one invocation of the routine.

1. The input model is fit. The current model is set to the input model.
2. If downward stepping is to be performed (**ISTEP** = **-**1 or **ISTEP** = 0), then each effect in the model is examined to determine if it can be deleted from the current model. An effect may be deleted from the current model if it is not a "forced effect" and if it must be included in the hierarchical specification of the model (in which lower order terms are not specified). Thus, for example, the effect *ABC* can be deleted from the model (*ABC*, *BCD*), yielding a model (*AB*, *AC*, *BCD*), but not from the model (*ABCD*) since *ABC* is not included in the hierarchical specification.

For each effect that can be deleted in a downward step, the usual chi-squared likelihood-ratio test statistic is computed as twice the difference of the log-likelihoods between the current model and the model in which the effect has been deleted. The degrees of freedom for the effect are determined (see below), and an asymptotic *p*-value is computed via the chi-squared distribution. After the *p*-values for all deleted models have been determined, the maximum *p*-value is selected. If it is greater than the *p*-value for deletion, **POUT**, the effect is deleted from the model, and the resulting model is fit.

3. If a downward step is not possible, either because all computed *p*-values are too small or because downward stepping is not to be performed, an upward step is attempted if requested (**ISTEP** = 0 or **ISTEP** = 1). For upward stepping, each effect in the full factorial analysis of variance specification of the table is examined to determine if the effect differs from the current model by exactly one term. For example, (*ABC*) differs by one term from the model (*AB*, *AC*, *BC*) and from the model (*ABD*, *ACD*, *BCD*), but it differs by more than one term from the model (*AB*, *BC*).

   For each effect that may be added to the model, a chi-squared likelihood-ratio test statistic is computed comparing the current model to the model with the added effect, its degrees of freedom are determined (see below), and an asymptotic *p*-value based upon the chi-squared distribution is computed. After all *p*-values for models with additive effects have been computed, the model with the minimum *p*-value is determined. If the minimum *p*-value is less than the *p*-value for addition, **PIN**, then the effect is added to the model, and the resulting model is fit.

4. If neither a step down, nor a step up can be performed, then **CTSTP** sets **IEND** = 1 and returns the original model to the user. Otherwise, if additional steps are to be made, execution continues at Step 2 above.

Degrees of Freedom

In **CTSTP**, structural zeros are considered to be a restriction of the parameter space. As such, they subtract from the degrees of freedom for an effect. Alternatively, sampling zeros are a result of sampling, and thus, they do not subtract for the degrees of freedom or restrict the parameter space. When computing degrees of freedom, sampling zeros are treated as if they were positive counts. If there are no structural zeros, then the degrees of freedom are computed as the product of the degrees of freedom for each variable in the effect where the degrees of freedom for the variable is the number of levels for the variable minus one. When structural zeros are present, there are restrictions on the parameter space, and the degrees of freedom for an effect are computed as the number of non-zero diagonal elements corresponding to the effect along the Cholesky factorization of the $X^T X$ matrix where X is the "design matrix" for the model. That is, each row of X contains the indicator variables for a cell in the table, with the indicator variables for the current model preceding the indicator variables for the effect for which degrees of freedom are desired. Because the degrees of freedom for an effect must be relative to the model, when there are structural zeros, it is possible for the degrees of freedom for an effect to change from one step to the next.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2STP/DC2STP. The reference is:

   CALL C2STP (IDO, NCLVAR, NCLVAL, TABLE, PIN, POUT, ISTEP, NSTEP, NFORCE, NEF, IPRINT, NVEF, MAXNVF, INDEF, MAXIND, FIT, STAT, IEND, MAXMAR, AMAR, INVEF, IINDEF, IDF, ZWK, RWK, IWK)

   The additional arguments are as follows:

   MAXMAR — The length of AMAR. (Input)
   When workspace is allocated by CTSTP, MAXMAR is equal to the number of workspace elements remaining after all other workspace is allocated. MAXMAR should be chosen as the maximum over all models considered of the sum over all marginal tables tables in the model of the number of elements in each marginal table.

   AMAR — Work vector of length MAXMAR used to store marginal means in the proportional fitting algorithm. (Output)

   INVEF — Work vector whose length is dependent on ISTEP, IPRINT, and
   z = the number of structural zeros in TABLE.

ISTEP	IPRINT		INVEF
-1, 0, 1	0, 1, 2	$z > 1$	$3v$
0, 1	0, 1, 2	$z \leq 1$	$3v$
-1	0	$z \leq 1$	$2v$

   Here, $v = 2^{\text{NCLVAR}} - 1$.

   IINDEF — Work vector whose length is dependent on ISTEP, IPRINT, and z = the number of structural zeros in TABLE.

ISTEP	IPRINT		IINDEF
-1, 0, 1	0, 1, 2	$z > 1$	$3d$
0, 1	0, 1, 2	$z \leq 1$	$3d$
-1	0	$z \leq 1$	$2d$

   Here, d = NCLVAR * $2^{\text{NCLVAR}-1}$.

   IDF — Vector of length $n + z$. (Output, for IDO = 0 or 1; input/output otherwise). Here, n = NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR). If there are no structural zeros in TABLE, IDF is not referenced and may be dimensioned of length 1 in the calling program. When using the IDO = 1, 2, or 3 option, the values stored in IDF should not be altered between calls to C2STP.

ZWK — Vector of length $n(n + 2)$. (Output, for IDO = 0 or 1; input/output otherwise). Here, n = NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR). If there are no structural zeros in TABLE, ZWK is not referenced and may be dimensioned of length 1 in the calling program. When using the IDO = 1, 2, or 3 option, the values stored in ZWK should not be altered between calls to C2STP.

RWK — Work vector whose length is dependent on IDO and z, the number of structural zeros in TABLE.

IDO		RWK
0, 1	$z > 1$	$2n + m$
0, 1	$z \leq 1$	n
2, 3	$z > 1$	n
2, 3	$z \leq 1$	n

Here, n = NCLVAL(1) * NCLVAL(2) * … * NCLVAL(NCLVAR) and m = NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR).

IWK — Work vector whose length is dependent on ISTEP and NSTEP.

ISTEP	NSTEP	Length of IWK
-1, 0	NSTEP = 0	3 * NCLVAR + NEF
-1, 0	NSTEP ≠ 0	3 * NCLVAR + v
1	NSTEP = 0	2 * NCLVAR + NEF
1	NSTEP ≠ 0	2 * NCLVAR + v

Here, $v = 2^{\text{NCLVAR}-1}$.

2. Informational errors

Type	Code	Description
3	1	The proportional fitting algorithm did not converge.
4	2	There is not enough workspace allocated for storing the marginal means.
4	3	The required length of NVEF to store the effects of the new model exceeds MAXNVF.
4	4	The required length of INDEF to store the effects of the new model exceeds MAXIND.

3. The cells of the vectors TABLE, and FIT are sequenced so that the first variable cycles from 1 to NCLVAL(1) the slowest, the second variable cycles from 1 to NCLVAL(2) the next slowest, etc., up to the NCLVAR-th variable, which cycles from 1 to NCLVAL(NCLVAR) the fastest.

Example: For NCLVAR = 3, NCLVAL(1) = 2, NCLVAL(2) = 3, and NCLVAL(3) = 2, the cells of table X(I, J, K) are entered into TABLE(1) through TABLE(12) in the following order. X(1, 1, 1), X(1, 1, 2), X(1, 2, 1), X(1, 2, 2), X(1, 3, 1), X(1, 3, 2), X(2, 1, 1), X(2, 1, 2), X(2, 2, 1), X(2, 2, 2), X(2, 3, 1), X(2, 3, 2). The elements of FIT are similarly sequenced.

4. INDEF is used to describe the marginal tables to be fit. For example, if NCLVAR = 3 and the first effect is to fit the marginal table for variables 1 and 3 and the second effect is to fit the marginal table for variable 2, then: NEF = 2, NVEF(1) = 2, and NVEF(2) = 1. Since the sum of the NVEF(I) is 3, then INDEF is a vector of length 3 with values: INDEF(1) = 1, INDEF(2) = 3, and INDEF(3) = 2.

Examples

Example 1

The following example is taken from Lee (1977). It involves a simple four-way table in which the first three factors have 2 levels, and the fourth factor has 3 levels. The data involves brand preference in different situations. In the example, the three-way interaction is removed, leaving 3 two-way interactions. In the new model, the three-way interaction is omitted.

```
      USE UMACH_INT
      USE CTSTP_INT
      USE WRIRN_INT
      USE WRRRN_INT
      USE ISUM_INT

      IMPLICIT  NONE
      INTEGER   IPRINT, LTAB, MAXIND, MAXNVF, NCLVAR
      PARAMETER  (IPRINT=2, LTAB=24, MAXIND=20, MAXNVF=10, NCLVAR=4)
!
      INTEGER    IEND, INDEF(MAXIND), ISTEP, LIND, NCLVAL(NCLVAR), &
                 NEF, NFORCE, NOUT, NSTEP, NVEF(MAXNVF)
      REAL       FIT(LTAB), STAT(3), TABLE(LTAB)
!
      DATA TABLE/19.0, 57.0, 29.0, 63.0, 29.0, 49.0, 27.0, 53.0, 23.0, &
          47.0, 33.0, 66.0, 47.0, 55.0, 23.0, 50.0, 24.0, 37.0, 42.0, &
          68.0, 43.0, 52.0, 30.0, 42.0/
      DATA NCLVAL/3, 2, 2, 2/, FIT/24*1.0/
      DATA NEF/1/
!
      CALL UMACH (2, NOUT)
!
      ISTEP    = 0
      NSTEP    = 1
      NFORCE   = 0
      NVEF(1)  = 3
      INDEF(1) = 1
      INDEF(2) = 2
      INDEF(3) = 4
!
      CALL CTSTP (NCLVAL, TABLE, ISTEP, NSTEP, NFORCE, NEF, &
```

```
                    NVEF, INDEF, FIT, STAT, IEND, IPRINT=IPRINT)
!
      WRITE (NOUT,99999) IEND, NEF
      CALL WRIRN ('NVEF', NVEF, 1, NEF, 1, 0)
      LIND = ISUM(NEF,NVEF,1)
      CALL WRIRN ('INDEF', INDEF, 1, LIND, 1, 0)
      CALL WRRRN ('FIT', FIT, 1, LTAB, 1, 0)
!
99999 FORMAT (/, ' IEND = ', I3, '   NEF = ', I3)
      END
```

Output

```
 Variable   Number of Levels
 1 A              3
 2 B              2
 3 C              2
 4 D              2

        ----------
    Table 1: B = 1 C = 1
   D (row) by A (column)
         1       2       3
 1   19.00   23.00   24.00
 2   57.00   47.00   37.00

        ----------
    Table 2: B = 1 C = 2
   D (row) by A (column)
         1       2       3
 1   29.00   33.00   42.00
 2   63.00   66.00   68.00

        ----------
    Table 3: B = 2 C = 1
   D (row) by A (column)
         1       2       3
 1   29.00   47.00   43.00
 2   49.00   55.00   52.00

        ----------
    Table 4: B = 2 C = 2
   D (row) by A (column)
         1       2       3
 1   27.00   23.00   30.00
 2   53.00   50.00   42.00

 ----------------------  Step: 0  ----------------------
 Input Model:  (A*B*D)
 Smallest p-value for removing effects      0.100
 Largest  p-value for entering effects      0.050
 Chi-squared                 33.92
 Degrees of Freedom            12.
 p-value                   0.0007
                                Degrees of
 Effect Tested      Chi-squared     Freedom     P-value
 A*B*D                    0.12           2      0.9408
 Effect Removed: A*B*D
```

```
 ----------------------  Step: 1  ----------------------
 Model:  (A*B, A*D, B*D)
 Chi-squared                 34.05
 Degrees of Freedom            14.
 p-value                   0.0020

 IEND =   0   NEF =   3

    NVEF
  1   2   3
  2   2   2

          INDEF
  1   2   3   4   5   6
  1   2   1   4   2   4


                                    FIT
    1       2       3       4       5       6       7       8       9      10
24.39   59.61   24.39   59.61   27.61   51.39   27.61   51.39   28.24   56.26

   11      12      13      14      15      16      17      18      19      20
28.24   56.26   34.76   52.74   34.76   52.74   32.38   53.12   32.38   53.12

   21      22      23      24
37.12   46.38   37.12   46.38
```

Example 2

Example two illustrates the use of CTSTP when sampling zeros are present. In this example, which is taken from Brown and Fuchs (1983), there are thirteen sampling zeros so that thirteen parameter estimates are infinite when the full model is fit. Here, we begin with the model fit by Brown and Fuchs, which, in CTSTP notation, is given as

$$(AC, AD, ABE, BCDE)$$

When this model is fit, there are five parameter estimates that are infinite. Note that these estimates have no effect on the degrees of freedom used in the tests computed here.

```
      USE UMACH_INT
      USE CTSTP_INT
      USE WRIRN_INT
      USE WRRRN_INT
      USE ISUM_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LTAB, MAXIND, MAXNVF, NCLVAR, I
      PARAMETER  (IPRINT=2, LTAB=32, MAXIND=30, MAXNVF=10, NCLVAR=5)
!
      INTEGER    IDO, IEND, INDEF(MAXIND), ISTEP, LIND, NCLVAL(NCLVAR), &
                 NEF, NFORCE, NOUT, NSTEP, NVEF(MAXNVF)
      REAL       FIT(LTAB), STAT(3), TABLE(LTAB)
!
      DATA TABLE/33.0, 32.0, 8.0, 8.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, &
          0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 2.0, 10.0, 3.0, 6.0, 1.0, &
          2.0, 0.0, 2.0, 0.0, 1.0, 0.0, 4.0, 0.0, 1.0, 0.0, 2.0/
      DATA NCLVAL/2, 2, 2, 2, 2/, FIT/32*1.0/, NEF/4/
```

```
      DATA (NVEF(I),I=1,4)/2, 2, 3, 4/
      DATA (INDEF(I),I=1,11)/1, 3, 1, 4, 1, 2, 5, 2, 3, 4, 5/
!
      CALL UMACH (2, NOUT)
!
      ISTEP  = -1
      NSTEP  = 2
      NFORCE = 0
!
      CALL CTSTP (NCLVAL, TABLE, ISTEP, NSTEP, NFORCE, NEF, &
                  NVEF, INDEF, FIT, STAT, IEND, IPRINT=IPRINT)
!
      WRITE (NOUT,99999) IEND, NEF
      CALL WRIRN ('NVEF', NVEF, 1, NEF, 1, 0)
      LIND = ISUM(NEF,NVEF,1)
      CALL WRIRN ('INDEF', INDEF, 1, LIND, 1, 0)
      CALL WRRRN ('FIT', FIT, 1, LTAB, 1, 0)
!
99999 FORMAT (/, ' IEND = ', I3, '   NEF = ', I3)
      END
```

Output

```
 Variable   Number of Levels
 1 A              2
 2 B              2
 3 C              2
 4 D              2
 5 E              2

          ----------
   Table 1: A = 1 B = 1 C = 1
     D (row) by E (column)
               1       2
     1   33.00   32.00
     2    8.00    8.00

          ----------
   Table 2: A = 1 B = 1 C = 2
     D (row) by E (column)
               1       2
     1   0.000   1.000
     2   1.000   0.000

          ----------
   Table 3: A = 1 B = 2 C = 1
     D (row) by E (column)
               1       2
     1   0.000   1.000
     2   0.000   0.000

          ----------
   Table 4: A = 1 B = 2 C = 2
     D (row) by E (column)
               1       2
     1   0.000   1.000
     2   0.000   0.000

          ----------
```

```
       Table 5: A = 2 B = 1 C = 1
         D (row) by E (column)
                   1       2
           1    2.00   10.00
           2    3.00    6.00

              ----------
       Table 6: A = 2 B = 1 C = 2
         D (row) by E (column)
                   1       2
           1   1.000   2.000
           2   0.000   2.000

              ----------
       Table 7: A = 2 B = 2 C = 1
         D (row) by E (column)
                   1       2
           1   0.000   1.000
           2   0.000   4.000

              ----------
       Table 8: A = 2 B = 2 C = 2
         D (row) by E (column)
                   1       2
           1   0.000   1.000
           2   0.000   2.000

     ----------------------  Step: 0  ----------------------
     Input Model:  (A*C, A*D, A*B*E, B*C*D*E)
     Smallest p-value for removing effects      0.100
     Chi-squared                   9.07
     Degrees of Freedom             10.
     p-value                     0.5251

                                    Degrees of
     Effect Tested      Chi-squared     Freedom     P-value
     A*C                       4.41           1      0.0358
     A*D                       6.56           1      0.0104
     A*B*E                     0.00           1      0.9912
     B*C*D*E                   0.00           1      0.9912
     Effect Removed: B*C*D*E

     ----------------------  Step: 1  ----------------------
     Model:  (A*C, A*D, A*B*E, B*C*D, B*C*E, B*D*E, C*D*E)
     Chi-squared                   9.07
     Degrees of Freedom             11.
     p-value                     0.6151

                                    Degrees of
     Effect Tested      Chi-squared     Freedom     P-value
     A*C                       4.41           1      0.0358
     A*D                       6.56           1      0.0104
     A*B*E                     0.00           1      1.0000
     B*C*D                     0.53           1      0.4673
     B*C*E                     0.00           1      1.0000
     B*D*E                     0.00           1      1.0000
     C*D*E                     0.10           1      0.7522
     Effect Removed: B*C*E

     ----------------------  Step: 2  ----------------------
```

```
 Model:  (A*C, A*D, A*B*E, B*C*D, B*D*E, C*D*E)
 Chi-squared                  9.07
 Degrees of Freedom             12.
 p-value                    0.6966
 IEND =   0   NEF =   6

          NVEF
1   2   3   4   5   6
2   2   3   3   3   3

                        INDEF
1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
1   3   1   4   1   2   5   2   3   4   2   4   5   3   4   5

                                FIT
    1      2      3      4      5      6      7      8      9     10
32.36  32.56   8.53   6.91   0.71   1.21   0.40   0.32   0.00   0.90

   11     12     13     14     15     16     17     18     19     20
0.00   0.75   0.00   0.27   0.00   0.09   2.64   9.44   2.47   7.09

   21     22     23     24     25     26     27     28     29     30
0.29   1.79   0.60   1.68   0.00   1.10   0.00   3.25   0.00   1.73

   31     32
0.00   1.91
```

CTRAN

Performs generalized Mantel-Haenszel tests in a stratified contingency table.

Required Arguments

NCLVAL — Vector of length NCLVAR containing, in its *i*-th element, the number of levels (categories) of the *i*-th classification variable. (Input)

TABLE — Vector of length NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR) containing the entries in the cells of the table to be fit. (Input)
See Comment 3 for comments on the ordering of the elements in TABLE. For the classification variables specified in INDROW and INDCOL, a series of two-dimensional contingency tables are obtained from the elements in TABLE. All other classification variables are stratification variables.

INDROW — Index of the classification variable to be used for the row variable in the stratified two-dimensional table. (Input)

INDCOL — Index of the classification variable to be used for the column variable in the stratified two-dimensional table. (Input)

ROWSCR — Vector of length NCLVAL(INDCOL) containing the scores associated with the column and used in each row. (Input, if IROWSC = 0; output, otherwise) ROWSCR is not used and can be dimensioned of length 1 in the calling program if ITYPE = 1. If IROWSC is 3, 4, or 5, then ROWSCR contains the scores used in the last contingency table analyzed.

COLSCR — Vector of length NCLVAL(INDROW) containing the scores associated with each row and used in each column. (Input, if ICOLSC = 0; output, otherwise) COLSCR is not used and can be dimensioned of length 1 in the calling program if ITYPE is not 3. If ICOLSC is 3, 4, or 5, then COLSCR contains the scores used in the last contingency table analyzed.

STAT — Table of size *m* by 3 containing the Mantel-Haenszel statistics. (Output)
Where *m* is one plus the number of stratified tables, i.e., *m* = 1 + NCLVAL(1) * NCLVAL(2) * ... * NCLVAL(NCLVAR)/(NCLVAL(INDROW) * NCLVAL(INDCOL)). The first column of STAT contains the chi-squared statistic for a test of partial association, the second column contains its degrees of freedom, and the third column contains the probability of a greater chi-squared. The first *m* - 1 rows of STAT contain the statistics computed for each of the stratified tables. The first row corresponds to the classification stratification variable levels (1, 1, ..., 1). The second row corresponds to levels (1, 1, ..., 2), etc., so that in row *m* - 1 all stratification variables are at their highest levels. The last row of STAT contains the same statistics pooled over all of the stratified tables.

Optional Arguments

NCLVAR — Number of classification variables. (Input)
Default: NCLVAR = size (NCLVAL,1).

ITYPE — The type of statistic to compute. (Input)
Default: ITYPE = 1.

ITYPE	Statistic
1	Generalized Mantel-Haenszel based upon the two-dimensional contingency tables.
2	Generalized Mantel-Haenszel based upon the row mean score in the two-dimensional table.
3	Generalized Mantel-Haenszel based upon the correlation score for the two-dimensional tables.

IROWSC — Option parameter giving the scores associated with the column index to be used when computing statistics in each row. (Input)
Default: IROWSC = 0.

IROWSC	Weights
0	User specified (or no) weights.
1	The digits 1, 2, …, NCLVAL(INDCOL).
2	Combined (over all tables) ridit-type scores.
3	Rank scores computed separately for each table.
4	Ridit-type scores computed separately for each table.
5	Logrank scores computed separately for each table.

IROWSC is not used if ITYPE = 1.

ICOLSC — Option parameter giving the scores associated with the row index to be used when computing statistics in each column. (Input)
Default: ICOLSC = 0.

ICOLSC	Weights
0	User specified (or no) weights.
1	The digits 1, 2, …, NCLVAL(INDROW).
2	Combined (over all tables) ridit-type scores.
3	Rank scores computed separately for each table.
4	Ridit-type scores computed separately for each table.
5	Logrank scores computed separately for each table.

ICOLSC is not used if ITYPE is not 3.

IPRINT — Print option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing.
1	Print the contents of the STAT array.
2	Print each stratified table followed by the contents of the STAT array.

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement of the calling program. (Input)
Default: LDSTAT = size (STAT,1).

FORTRAN 90 Interface

Generic: CALL CTRAN (NCLVAL, TABLE, INDROW, INDCOL, ROWSCR, COLSCR, STAT [, …])
Specific: The specific interface names are S_CTRAN and D_CTRAN.

FORTRAN 77 Interface

Single: CALL CTRAN (NCLVAR, NCLVAL, TABLE, INDROW, INDCOL, ITYPE, IROWSC, ICOLSC, IPRINT, ROWSCR, COLSCR, STAT, LDSTAT)
Double: The double precision name is DCTRAN.

Description

Routine CTRAN computes tests of partial association (a test of homogeneity, a test on means, and a test on correlations) in stratified two-dimensional contingency tables. The type of test computed depends upon parameter ITYPE. All tests are generalizations of the Mantel-Haenszel stratified 2 × 2 contingency table test statistic in the

sense that information is "pooled" over all tables without increasing the total degrees of freedom in the test. Like the Mantel-Haenszel test, if all tables violate the null hypothesis in the same direction, the tests computed here are more powerful than most other tests of the same null hypothesis.

While **CTRAN** allows for an arbitrary number of classification variables, only three are required to describe the test statistics since all stratification variables could be (if desired) lumped into a single classification variable. Because of this, only three classification variables are discussed here. Let f_{ijk} denote the frequency in cell *ij* of stratum *k* where *k* = 1, ..., *m*, *i* = 1, ..., *r*, and *j* = 1, ..., *c*. Then, the input data can be described as a series of contingency tables. For example, if *r* = *c* = 2, so that 2 × 2 tables are used, then we would have:

f_{111}	f_{121}		f_{112}	f_{122}	...	f_{11m}	f_{12m}
f_{211}	f_{221}		f_{212}	f_{222}	...	f_{21m}	f_{22m}

All tests are computed as follows: For each table, a test statistic vector xk with estimated covariance matrix

$$\hat{\sum}_k$$

is computed. The test statistic vector x_k represents the mean difference (from the null hypothesis) for the test being computed. Thus, if **ITYPE** = 1, x_k is a vector of cell frequencies minus their expected value under the hypothesis of homogeneity while if **ITYPE** = 2, x_k is a vector containing the row means (based upon the row scores) for the elements in a row of a table minus the estimated mean for the table (estimated under the assumption that all means are equal). Finally, if **ITYPE** = 3, x_k is a vector of length 1 containing an estimated correlation coefficient computed between the row and column scores.

Note that for nominal data in both the rows and columns, one would generally use **ITYPE** = 1 while if an ordering (and scores) make sense for each row of a table, **ITYPE** = 2 would be used. If an ordering (and scores) make sense for both the rows and the columns of a table, then a correlation measure (**ITYPE** = 3) is appropriate.

Test statistics for each table are computed as

$$\chi_k^2 = x_k^T \hat{\sum}_k^{-1} x_k$$

which has degrees of freedom (*r* - 1)(*c* - 1) when **ITYPE** = 1, *r* - 1 when **ITYPE** = 2, and 1 for **ITYPE** = 3. While these test statistics could be combined by summing them over all tables (yielding a $\mathbf{X}^2$ test with *m* times the of degrees of freedom in a single table), the Mantel-Haenszel test combines the scores in a different way. Let

$$x = \sum_k x_k, \text{ and let } \hat{\sum} = \sum_k \hat{\sum}_k$$

Then, an overall $\mathbf{X}^2$ may be computed as

$$x^T \hat{\sum}^{-1} x$$

This test statistic has the same degrees of freedom as the test statistic computed for a single stratum of the three-way table and is reported in the last row of **STAT**. Routine **CTRAN** uses simplified computational methods. See Landis, Cooper, Kennedy, and Koch (1979) for details.

Landis, Cooper, Kennedy, and Koch (1979, page 225) give the null hypothesis for a test of partial association as follows (paraphrased):

> H_0 : For each of the separate tables, the data in the respective rows of the table can be regarded as a successive set of simple random samples from a fixed population corresponding to the column marginal totals for the table.

All three tests above are tests of partial association.

For **ITYPE**= 2 and 3, different row and column (**ITYPE** = 3) scores are used in computing measures of location and association. The scores used by **CTRAN** for the rows are

1. For **IROWSC** = 0, the user supplies the scores to be used in each row of the table.
2. For **IROWSC** = 1, uniform scores are used. These scores consist of the digits 1, 2, …, *c* where *c* is the number of columns in each table.
3 For **IROWSC** = 2, combined ridit scores are used. A combined ridit score is computed by summing the column marginals over all tables. The combined row score for the *j*-th column is then computed as the sum of the initial *j* - 1 column marginals plus half of the *j*-th column marginal. The result is divided by the number of observations in all tables to yield the ridit score.
4. For **IROWSC** = 3, marginal rank scores are used. The *j*-th marginal rank score is computed for each table from the column marginals for that table as the sum of the initial *j* - 1 column marginals plus half the *j*-th column marginal.
5. For **IROWSC** = 4, marginal ridit scores are used. These are computed as the marginal rank scores divided by the total frequency in the table.
6. For **IROWSC** = 5, logrank scores are used. These are computed as

$$c_{jk} = 1 - \sum_{l=1}^{j} \frac{f_{+lk}}{\sum_{i=l}^{c} f_{+ik}}$$

where f_{+lk} is the column marginal for column l in table *k*.

Column scores are computed in a similar manner.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2RAN/DC2RAN. The reference is:

   CALL C2RAN (NCLVAR, NCLVAL, TABLE, INDROW, INDCOL, ITYPE, IROWSC, ICOLSC, IPRINT, ROWSCR, COLSCR, STAT, LDSTAT, IX, F, COLSUM, ROWSUM, DIFVEC, DIFSUM, COV, COVSUM, AWK, BWK)

   The additional arguments are as follows:

   IX — Work array of length NCLVAR.

   F — Work array of length NCLVAL(INDROW) * NCLVAL(INDCOL).

   COLSUM — Work array of length NCLVAL(INDCOL).

   ROWSUM — Work array of length NCLVAL(INDROW).

   DIFVEC — Work array. If ITYPE = 1, the length is (NCLVAL(INDROW) - 1) * (NCLVAL(INDCOL) - 1). For ITYPE = 2, the length is NCLVAL(INDROW). For ITYPE = 3, DIFVEC is not used and may be of length 1.

   DIFSUM — Work array. If ITYPE = 1, the length is (NCLVAL(INDROW) - 1) * (NCLVAL(INDCOL) - 1). DIFSUM contains the sum of the tables containing the observed minus expected frequencies (excluding the last row and column of each table). For ITYPE = 2, the length is NCLVAL(INDROW). DIFSUM contains the sum of the table row mean scores minus their expected value. For ITYPE = 3, the length is 1. DIFSUM contains the sum of the table correlations between the row and column mean scores. (Output)

   COV — Work array. If ITYPE = 1, the length is (NCLVAL(INDROW) - 1)^2 * (INCLVA(INDCOL) - 1)^2. For ITYPE = 2, the length is NCLVAL(INDROW)^2. For ITYPE = 3, COV is not used and may be of length 1.

   COVSUM — Work array. If ITYPE = 1, the length is (NCLVAL(INDROW) - 1)^2 * (INCLVA(INDCOL) - 1)^2. For ITYPE = 2, the length is NCLVAL(INDROW)^2. For ITYPE = 3, the length is 1.

   AWK — Work array. If ITYPE = 1, the length is (NCLVAL(INDROW) - 1)^2. For ITYPE = 2, the length is NCLVAL(INDROW). For ITYPE = 3, AWK is not used and may be of length 1.

   BWK — Work array. If ITYPE = 1, the length is (NCLVAL(INDCOL) - 1)^2. For ITYPE= 2 or 3, BWK is not used and may be of length 1.

2. Informational errors

Type	Code	Description
3	1	All frequencies of stratified table K are zero. This table will be excluded from the Mantel-Haenszel test statistic.
3	2	The elements of stratified table K sum to one. This table will be excluded from the Mantel-Haenszel test statistic.
3	3	The variance of the response variable for stratified table K is zero.
3	4	The variance of either the sub-population or the response variable is zero for stratified table K.
3	5	The label for table K exceeds the buffer limit of 72.

Here, **K** is an integer that is greater than or equal to one and less than or equal to the number of stratified contingency tables.

3. The cells of the vectors **TABLE** are sequenced so that the first variable cycles from 1 to **NCLVAL**(1) the slowest, the second variable cycles from 1 to **NCLVAL**(2) the next most slowly, and so on, up to the **NCLVAR**-th variable, which cycles from 1 to **NCLVAL**(**NCLVAR**) the fastest.

   Example: For **NCLVAR** = 3, **NCLVAL**(1) = 2, **NCLVAL**(2) = 3, and **NCLVAL**(3) = 2 the cells of table **X**(**I**, **J**, **K**) are entered into **TABLE**(1) through **TABLE**(12) in the following order: **X**(1, 1, 1), **X**(1, 1, 2), **X**(1, 2, 1), **X**(1, 2, 2), **X**(1, 3, 1), **X**(1, 3, 2), **X**(2, 1, 1), **X**(2, 1, 2), **X**(2, 2, 1), **X**(2, 2, 2), **X**(2, 3, 1), **X**(2, 3, 2).

Example

In the following example, all three values of **ITYPE** are used in computing the partial association statistics. This is accomplished via three calls to **CTRAN**. The value of **ITYPE** changes on each call. The example is taken from Landis, Cooper, Kennedy, and Koch (1979, page 241). Uniform scores are used in both the rows and column as required by the tests type. The results indicate the presence of association between the row and column variables.

```
      USE CTRAN_INT

      IMPLICIT   NONE
      INTEGER    ICOLSC, INDCOL, INDROW, IROWSC, LDSTAT, NCLVAR
      PARAMETER  (ICOLSC=1, INDCOL=1, INDROW=3, IROWSC=1, LDSTAT=5, &
                 NCLVAR=3)
!
      INTEGER    IPRINT, ITYPE, NCLVAL(NCLVAR)
      REAL       COLSCR(4), ROWSCR(3), STAT(LDSTAT,3), TABLE(48)
!
      DATA TABLE/23, 23, 20, 24, 18, 18, 13, 9, 8, 12, 11, 7, 12, 15, &
          14, 13, 7, 10, 13, 10, 6, 6, 13, 15, 6, 4, 6, 7, 9, 3, 8, &
          6, 2, 5, 5, 6, 1, 2, 2, 2, 3, 4, 2, 4, 1, 2, 3, 4/
      DATA NCLVAL/3, 4, 4/
!
      IPRINT = 2
```

```
      DO 10  ITYPE=1, 3
         CALL CTRAN (NCLVAL, TABLE, INDROW, INDCOL,  &
                     ROWSCR, COLSCR, STAT, ITYPE=ITYPE, &
                     IROWSC=IROWSC, ICOLSC=ICOLSC, IPRINT=IPRINT)
         IPRINT = 1
!
   10 CONTINUE
      END
```

Output

```
 Values for the class variables are defined to be:
 Variable   Number of Levels
 1 A              3
 2 B              4
 3 C              4

          ----------
       Strata 1: B = 1
     C (row) by A (column)
            1       2       3
 1   23.00    7.00    2.00
 2   23.00   10.00    5.00
 3   20.00   13.00    5.00
 4   24.00   10.00    6.00

          ----------
       Strata 2: B = 2
     C (row) by A (column)
            1       2       3
 1   18.00    6.00    1.00
 2   18.00    6.00    2.00
 3   13.00   13.00    2.00
 4    9.00   15.00    2.00

          ----------
       Strata 3: B = 3
     C (row) by A (column)
            1       2       3
 1    8.00    6.00    3.00
 2   12.00    4.00    4.00
 3   11.00    6.00    2.00
 4    7.00    7.00    4.00

          ----------
       Strata 4: B = 4
     C (row) by A (column)
            1       2       3
 1   12.00    9.00    1.00
 2   15.00    3.00    2.00
 3   14.00    8.00    3.00
 4   13.00    6.00    4.00
 Test of independence between row and column variables

                           Degrees of
       Strata  Chi-Squared     Freedom  Probability
            1         3.4           6       0.7575
            2        10.8           6       0.0942
```

```
          3          3.1           6       0.7987
          4          5.2           6       0.5177

                                Degrees of
                  Chi-Squared     Freedom  Probability
Mantel-Haenszel         10.6           6       0.1016
Test of equality of location for rows given column scores

                              Degrees of
      Strata  Chi-Squared     Freedom  Probability
           1         2.62           3       0.4536
           2         7.34           3       0.0617
           3         1.69           3       0.6381
           4         1.68           3       0.6420

                                Degrees of
                  Chi-Squared     Freedom  Probability
Mantel-Haenszel         6.59           3      0.08618
     Row Scores
    1       2       3
 1.000   2.000   3.000
Test of correlation given row and column scores

                         Degrees of
 Strata  Chi-Squared     Freedom  Probability
      1         1.57           1       0.2105
      2         7.06           1       0.0079
      3         0.16           1       0.6927
      4         0.66           1       0.4161

                                Degrees of
                  Chi-Squared     Freedom  Probability
Mantel-Haenszel         6.34           1       0.0118
     Row Scores
    1       2       3
 1.000   2.000   3.000

        Column Scores
    1       2       3       4
 1.000   2.000   3.000   4.000
```

CTGLM

Analyzes categorical data using logistic, Probit, Poisson, and other generalized linear models.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

MODEL — Model option parameter. (Input)

MODEL specifies the distribution of the response variable and the function used to model the distribution parameter. The lower-bound given in the following table is the minimum possible value of the response variable:

MODEL			
0	Poisson	Exponential	0
1	Neg. Binomial	Logistic	0
2	Logarithmic	Logistic	1
3	Binomial	Logistic	0
4	Binomial	Probit	0
5	Binomial	Log-log	0

Let γ be the dot product of a row in the design matrix with the parameters (plus the fixed parameter, if used). Then, the functions used to model the distribution parameter are given by:

Exponential	$\exp(\gamma)$
Logistic	$\exp(\gamma)/(1 + \exp(\gamma))$
Probit	Normal(γ) (normal cdf)
Log-log	$1 - \exp(-\gamma)$

NCOEF — Number of estimated coefficients in the model. (Output)

COEF — NCOEF by 4 matrix containing the parameter estimates and associated statistics. (Output, if INIT = 0; input, if INIT = 1 and MAXIT = 0; input/output, if INIT = 1 and MAXIT > 0)

	Statistic
1	Coefficient estimate.
2	Estimated standard deviation of the estimated coefficient.
3	Asymptotic normal score for testing that the coefficient is zero.
4	*p*-value associated with the normal score in column 3.

When INIT = 1, only the first column needs to be specified on input.

COV — NCOEF by NCOEF matrix containing the estimated asymptotic covariance matrix of the coefficients. (Output)
For MAXIT = 0, this is the Hessian computed at the initial parameter estimates.

XMEAN — Vector of length NCOEF containing the means of the design variables. (Output)

GR — Vector of length NCOEF containing the last parameter updates (excluding step halvings). (Output)
For MAXIT = 0, GR contains the inverse of the Hessian times the gradient vector, all computed at the initial parameter estimates.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ILT — For full-interval and left-interval observations, the column number in X that contains the upper endpoint of the observation interval. (Input)
See argument ICEN. If ILT = 0, left-interval and full-interval observations cannot be input.
Default: ILT = 0.

IRT — For full-interval and right-interval observations, the column number in X that contains the lower endpoint of the interval. (Input)
For point observations, X(*i*, IRT) contains the observation point. IRT must not be zero. See argument ICEN. In the usual case, all observations are "point" observations.
Default: IRT = size (X,2).

IFRQ — Column number in X containing the frequency of response for each observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

IFIX — Column number in X containing a fixed parameter for each observation that is added to the linear response prior to computing the model parameter. (Input)
The "fixed" parameter allows one to test hypothesis about the parameters via the log-likelihoods. If IFIX = 0, the fixed parameter is assumed to be 0.
Default: IFIX = 0.

IPAR — Column number in X containing an optional distribution parameter for each observation. (Input)
Default: IPAR = 0.
If IPAR = 0, the distribution parameter is assumed to be 1. The meaning of the distributional parameter depends upon MODEL as follows:

MODEL	Meaning of X(I, IPAR)
0	The Poisson parameter is given by X(*i*, IPAR) * exp(γ).
1	The number of successes required in the negative binomial is given by X(*i*, IPAR).
2	X(*i*, IPAR) is not used.
3 - 5	The number of trials in the binomial distribution is given by X(*i*, IPAR).

ICEN — Column number in X containing the interval-type for each observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a code of 0 is assumed. Valid codes are

X(i,ICEN)	Censoring
0	Point observation. The response is unique and is given by X(*i*, IRT).
1	Right-interval. The response is greater than or equal to X(*i*, IRT) and less than or equal to the upper bound, if any, of the distribution.
2	Left-interval. The response is less than or equal to X(*i*, ILT) and greater than or equal to the lower bound of the distribution.
3	Full-interval. The response is greater than or equal to X(*i*, IRT), but less than or equal to X(*i*, ILT).

INFIN — Method to be used for handling infinite estimates. (Input)
Default: INFIN = 1.

INFIN	Method
0	Remove a right or left-censored observation from the log-likelihood whenever the probability of the observation exceeds 0.995. At convergence, use linear programming to check that all removed observations actually have an estimated linear response that is infinite. Set IADD(*i*) for observation *i* to 2 if the linear response is infinite. If not all removed observations have infinite linear response, recompute the estimates based upon the observations with estimated linear response that is finite.This option is valid only for censoring codes (see ICEN) 1 and 2.
1	Iterate without checking for infinite estimates.

See the Description section for more discussion.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is usually sufficient. Use MAXIT = 0 to compute the Hessian, stored in COV, and the Newton step, stored in GR, at the initial estimates.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum relative change in any coefficient estimate is less than EPS from one iteration to the next or when the relative change in the log-likelihood, ALGL, from one iteration to the next is less than EPS/100. If EPS is negative, EPS = 0.001 is assumed.
Default: EPS = .001.

INTCEP — Intercept option. (Input)
Default: INTCEP = 1

INTCEP	Action
0	No intercept is in the model (unless otherwise provided for by the user).
1	Intercept is automatically included in the model.

NCLVAR — Number of classification variables. (Input)
Dummy or indicator variables are generated for classification variables using the IDUMMY = 2 option of IMSL routine GRGLM (Chapter 2, "Regression"). See Comment 3.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are classification variables. (Input, if NCLVAR is positive; not used otherwise). If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: NEF = 0.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input, if NEF is positive; not used otherwise).
If NEF is zero, NVEF is not used and can be dimensioned of length 1 in the calling program.

INDEF — Index vector of length NVEF(1) + NVEF(2) + … + NVEF(NEF) containing the column numbers in X associated with each effect.(Input, if NEF is positive, not used otherwise)
The first NVEF(1) elements of INDEF give the column numbers of the variables in the first effect. The next NVEF(2) elements give the column numbers for the second effect, etc. If NEF is zero, INDEF is not used and can be dimensioned of length 1 in the calling program.

INIT — Initialization option. (Input)
Default: INIT = 0.

INIT	Action
0	Unweighted linear regression is used to obtain initial estimates.
1	The NCOEF elements in the first column of COEF contain initial estimates of the parameters on input to CTGLM (requiring that the user know NCOEF prior to calling CTGLM).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed, but observational statistics are not printed.
2	All output statistics are printed.

MAXCL — An upper bound on the sum of the number distinct values taken on by each classification variable. (Input)
Default: If NCLVAR = 0, then MAXCL = 1, else MAXCL = 3 * NCLVAR.

NCLVAL — Vector of length NCLVAR containing the number of values taken by each classification variable. (Output, if NCLVAR is positive; not used otherwise)
NCLVAL(*i*) is the number of distinct values for the *i*-th classification variable. If NCLVAR is zero, NCLVAL is not used and can be dimensioned of length 1 in the calling program.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the distinct values of the classification variables in ascending order. (Output, if NCLVAR is positive; not used otherwise)
Since in general the length of CLVAL will not be known in advance, MAXCL (an upper bound for this length) should be used for purposes of dimensioning CLVAL. The first NCLVAL(1) elements of CLVAL contain the values for the first classification variables, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then CLVAL is not referenced and can be dimensioned of length 1 in the calling program.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

ALGL — Value of the log-likelihood evaluated at the final estimates. (Output)

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

CASE — NOBS by 5 vector containing the case analysis. (Output)

	Statistic
1	Predicted parameter.
2	The residual.
3	The estimated standard error of the residual.
4	The estimated influence of the observation.
5	The standardized residual.

Case statistics are computed for all observations except where missing values prevent their computation.

The predicted parameter in column 1 depends upon MODEL as follows.

MODEL	Parameter
0	The predicted mean for the observation.
1-5	The probability of a success on a single trial.

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)

IADDS — Vector of length NOBS indicating which observations are included in the extended likelihood. (Output, if MAXIT > 0; input/output, if MAXIT = 0)

Value	Status of Observation
0	Observation *i* is in the likelihood.
1	Observation *i* cannot be in the likelihood because it contains at least one missing value in X.
2	Observation *i* is not in the likelihood. Its estimated parameter is infinite. For MAXIT = 0, the IADDS array must be initialized prior to calling CTGLM.

In this case, some elements of **IADDS** may be set to 1, by **CTGLM**, but no check for infinite estimates performed.

NRMISS — Number of rows of data in **X** that contain missing values in one or more columns **ILT**, **IRT**, **IFRQ**, **IFIX**, **IPAR**, **ICEN**, **INDCL**, or **INDEF** of **X**. (Output)

FORTRAN 90 Interface

Generic: **CALL CTGLM** (**X**, **MODEL**, **NCOEF**, **COEF**, **COV**, **XMEAN**, **GR** [, ...])

Specific: The specific interface names are **S_CTGLM** and **D_CTGLM**.

FORTRAN 77 Interface

Single: **CALL CTGLM** (**NOBS**, **NCOL**, **X**, **LDX**, **MODEL**, **ILT**, **IRT**, **IFRQ**, **IFIX**, **IPAR**, **ICEN**, **INFIN**, **MAXIT**, **EPS**, **INTCEP**, **NCLVAR**, **INDCL**, **NEF**, **NVEF**, **INDEF**, **INIT**, **IPRINT**, **MAXCL**, **NCLVAL**, **CLVAL**, **NCOEF**, **COEF**, **LDCOEF**, **ALGL**, **COV**, **LDCOV**, **XMEAN**, **CASE**, **LDCASE**, **GR**, **IADDS**, **NRMISS**)

Double: The double precision name is **DCTGLM**.

Description

Routine **CTGLM** uses iteratively reweighted least squares to compute (extended) maximum likelihood estimates in some generalized linear models involving categorized data. One of several models, including the probit, logistic, Poisson, logarithmic, and negative binomial models, may be fit for input point or interval observations. (In the usual case, only point observations are observed.)

Let

$$\gamma_i = w_i + x_i^T \beta = w_i + \eta_i$$

be the linear response where x_i is a design column vector obtained from a row of *X*, β is the column vector of coefficients to be estimated, and w_i is a fixed parameter that may be input in **X**. When some of the γ_i are infinite at the supremum of the likelihood, then *extended maximum likelihood estimates* are computed. Extended maxi-

mum likelihood are computed as the finite (but nonunique) estimates $\hat{\beta}$ that optimize the likelihood containing only the observations with finite $\hat{\gamma}_i$. These estimates, when combined with the set of indices of the observations such that $\hat{\gamma}_i$ is infinite at the supremum of the likelihood, are called extended maximum estimates. When none of the optimal $\hat{\gamma}_i$ are infinite, extended maximum likelihood estimates are identical to maximum likelihood estimates. Extended maximum likelihood estimation is discussed in more detail by Clarkson and Jennrich (1991). In **CTGLM**, observations with potentially infinite

$$\hat{\eta}_i = x_i^T \hat{\beta}$$

are detected and removed from the likelihood if **INFIN** = 0. See below.

The models available in **CTGLM** are:

MODEL	Name	Parameterization	PDF
0	Poisson	$\lambda = N * \exp(w+\eta)$	$f(y) = \lambda^y \exp(-\lambda)/y!$
1	Neg. Binomial	$\theta = \frac{\exp\{w+\eta\}}{1+\exp\{w+\eta\}}$	$f(y) = \binom{S+y-1}{y-1}\theta^S(1-\theta)^y$
2	Logarithmic	$\theta = \frac{\exp\{w+\eta\}}{1+\exp\{w+\eta\}}$	$f(y) = (1-\theta)^y/(y\ln\theta)$
3	Logistic	$\theta = \frac{\exp\{w+\eta\}}{1+\exp\{w+\eta\}}$	$f(y) = \binom{N}{y}\theta^y(1-\theta)^{N-y}$
4	Probit	$\theta = \Phi(w+\eta)$	$f(y) = \binom{N}{y}\theta^y(1-\theta)^{N-y}$
5	Log-log	$\theta = 1-\exp\{-\exp(w+\eta)\}$	$f(y) = \binom{N}{y}\theta^y(1-\theta)^{N-y}$

Here, **Φ** denotes the cumulative normal distribution, *N* and *S* are known parameters specified for each observation via column **IPAR** of **X**, and *w* is an optional fixed parameter specified for each observation via column **IFIX** of **X**. (If **IPAR** = 0, then *N* is taken to be 1 for **MODEL** = 0, 3, 4 and 5 and *S* is taken to be 1 for **MODEL** = 1. If **IFIX** = 0, then *w* is taken to be 0.) Since the log-log model (**MODEL** = 5) probabilities are not symmetric with respect to 0.5, quantitatively, as well as qualitatively, different models result when the definitions of "success" and "failure" are interchanged in this distribution. In this model and all other models involving **θ**, **θ** is taken to be the probability of a "success."

Note that each row vector in the data matrix can represent a single observation; or, through the use of column **IFRQ** of the matrix **X**, each vector can represent several observations. Also note that classification variables and their products are easily incorporated into the models via the usual regression-type specifications.

Computational Details

For interval observations, the probability of the observation is computed by summing the probability distribution function over the range of values in the observation interval. For right-interval observations, Pr($Y \geq y$) is computed as a sum based upon the equality Pr($Y \geq y$) = 1 - Pr($Y < y$). Derivatives are computed similarly. **CTGLM** allows three types of interval observations. In full interval observations, both the lower and the upper endpoints of the interval must be specified. For right-interval observations, only the lower endpoint need be given while for left-interval observations, only the upper endpoint is given.

The computations proceed as follows:

1. The input parameters are checked for consistency and validity.
2. Estimates of the means of the "independent" or design variables are computed. The frequency of the observation in all but binomial distribution models is taken from column **IFRQ** of the data matrix **X**. In binomial distribution models, the frequency is taken as the product of n = **X**(**I**, **IPAR**) and **X**(**I**, **IFRQ**). In all cases, if **IFRQ** = 0, or **IPAR** = 0, these values default to 1. Means are computed as

$$\bar{x} = \frac{\sum_i f_i x_i}{\sum_i f_i}$$

3. If **INIT**= 0, initial estimates of the coefficients are obtained (based upon the observation intervals) as multiple regression estimates relating transformed observation probabilities to the observation design vector. For example, in the binomial distribution models, **θ** for point observations may be estimated as

$$\hat{\theta} = \mathrm{X}(\mathrm{I},\mathrm{IRT}) / \mathrm{X}(\mathrm{I},\mathrm{IPAR})$$

and, when MODEL = 3, the linear relationship is given by

$$\left(\ln\left(\hat{\theta} / \left(1 - \hat{\theta}\right)\right) \approx X\beta\right)$$

while if MODEL = 4,

$$\left(\Phi^{-1}\left(\hat{\theta}\right) = X\beta \right)$$

For bounded interval observations, the midpoint of the interval is used for **X(I,IRT)**. Right-interval observations are not used in obtaining initial estimates when the distribution has unbounded support (since the midpoint of the interval is not defined). When computing initial estimates, standard modifications are made to prevent illegal operations such as division by zero.

Regression estimates are obtained at this point, as well as later, by use of routine **RGIVN** (see Chapter 2, "Regression".)

4. Newton-Raphson iteration for the maximum likelihood estimates is implemented via iteratively reweighted least squares. Let

$$\Psi\left(x_i^T\beta\right)$$

denote the log of the probability of the *i*-th observation for coefficients **β**. In the least-squares model, the weight of the *i*-th observation is taken as the absolute value of the second derivative of

$$\Psi\left(x_i^T\beta\right)$$

with respect to

$$\gamma_i = x_i^T\beta$$

(times the frequency of the observation), and the dependent variable is taken as the first derivative **Ψ** with respect to γ_i, divided by the square root of the weight times the frequency. The Newton step is given by

$$\Delta\beta = \left(\sum_i \left| \Psi''\left(\gamma_i\right) \right| x_i x_i^T \right)^{-1} \sum_i \Psi'\left(\gamma_i\right) x_i$$

where all derivatives are evaluated at the current estimate of **γ**, and $\beta_{n+1} = \beta_n - \Delta\beta$. This step is computed as the estimated regression coefficients in the least-squares model. Step halving is used when necessary to ensure a decrease in the criterion.

5. Convergence is assumed when the maximum relative change in any coefficient update from one iteration to the next is less than **EPS** or when the relative change in the log--likelihood from one iteration to the next is less than **EPS**/100. Convergence is also assumed after **MAXIT** iterations or when step halving leads to a step size of less than .0001 with no increase in the log-likelihood.

6. For interval observations, the contribution to the log-likelihood is the log of the sum of the probabilities of each possible outcome in the interval. Because the distributions are discrete, the sum may involve many terms. The user should be aware that data with wide intervals can lead to expensive (in terms of computer time) computations.

7. If requested (**INFIN** = 0), then the methods of Clarkson and Jennrich (1991) are used to check for the existence of infinite estimates in

$$\eta_i = x_i^T \beta$$

As an example of a situation in which infinite estimates can occur, suppose that observation *j* is right censored with $t_j > 15$ in a logistic model. If design matrix *X* is such that $x_{jm} = 1$ and $x_{im} = 0$ for all $i \neq j$, then the optimal estimate of β_m occurs at

$$\hat{\beta}_m = \infty$$

leading to an infinite estimate of both β_m and η_j. In **CTGLM**, such estimates may be "computed."

In all models fit by **CTGLM**, infinite estimates can only occur when the optimal estimated probability associated with the left- or right-censored observation is 1. If **INFIN** = 0, left- or right- censored observations that have estimated probability greater than 0.995 at some point during the iterations are excluded from the log-likelihood, and the iterations proceed with a log-likelihood based upon the remaining observations. This allows convergence of the algorithm when the maximum relative change in the estimated coefficients is small and also allows for the determination of observations with infinite

$$\eta_i = x_i^T \beta$$

At convergence, linear programming is used to ensure that the eliminated observations have infinite η_i. If some (or all) of the removed observations should not have been removed (because their estimated η_i's must be finite), then the iterations are restarted with a log-likelihood based upon the finite η_i observations. See Clarkson and Jennrich (1991) for more details.

When **INFIN** = 1, no observations are eliminated during the iterations. In this case, when infinite estimates occur, some (or all) of the coefficient estimates $\hat{\beta}$ will become large, and it is likely that the Hessian will become (numerically) singular prior to convergence.

When infinite estimates for the $\hat{\eta}_j$ are detected, routine **RGIVN** (see Chapter 2, "Regression") is used at the convergence of the algorithm to obtain unique estimates $\hat{\beta}$. This is accomplished by regressing the optimal $\hat{\eta}_j$ or the observations with finite η against $X\beta$, yielding a unique $\hat{\beta}$ (by setting coefficients $\hat{\beta}$ that are linearly related to previous coefficients in the model to zero). All of the final statistics relating to $\hat{\beta}$ are based upon these estimates.

8. Residuals are computed according to methods discussed by Pregibon (1981). Let $l_i(\gamma_i)$ denote the log-likelihood of the *i*-th observation evaluated at γ_i. Then, the standardized residual is computed as

$$r_i = \frac{\ell_i'\left(\hat{\gamma}_i\right)}{\sqrt{\ell_i''\left(\hat{\gamma}_i\right)}}$$

where $\hat{\gamma}_i$ is the value of γ_i when evaluated at the optimal $\hat{\beta}$ and the derivatives here (and only here) are with respect to γ rather than with respect to β. The denominator of this expression is used as the "standard error of the residual" while the numerator is the "raw" residual.

Following Cook and Weisberg (1982), we take the influence of the *i*-th observation to be

$$\ell_i'\left(\hat{\gamma}_i\right)^T \ell''\left(\hat{\gamma}\right)^{-1} \ell'\left(\hat{\gamma}_i\right)$$

This quantity is a one-step approximation to the change in the estimates when the *i*-th observation is deleted. Here, the partial derivatives are with respect to β.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2GLM/DC2GLM**. The reference is:

```
CALL C2GLM (NOBS, NCOL, X, LDX, MODEL, ILT, IRT, IFRQ, IFIX, IPAR, ICEN, INFIN,
   MAXIT, EPS, INTCEP, NCLVAR, INDCL, NEF, NVEF, INDEF, INIT, IPRINT, MAXCL,
   NCLVAL, CLVAL, NCOEF, COEF, LDCOEF, ALGL, COV, LDCOV, XMEAN, CASE, LDCASE,
   GR, IADD, NRMISS, NMAX, OBS, ADDX, XD, WK, KBASIS)
```

The additional arguments are as follows:

NMAX — Maximum number of observations that can be handled in the linear programming. (Input)
If workspace is not explicitly provided, **NMAX** is set to **NMAX** = (*n* - 8)/(7 + **NCOEF**) in **S_CTGLM** and **NMAX** = (*n* - 16)/(11 + 2 * **NCOEF**) in **D_CTGLM** where *n* is the maximum number of units of workspace available after allocating space for **OBS**. In the typical problem,

no linear programming is performed so that NMAX = 1 is sufficient. NMAX = NOBS is always sufficient. Even when extended maximum likelihood estimates are computed, NMAX = 30 will usually suffice. If INFIN = 1, set NMAX = 0.

OBS — Work vector of length NCOEF + 1.

ADDX — Logical work vector of length NMAX. ADDX is not needed and can be a array of length 1 in the calling program if NMAX = 0.

XD — Work vector of length NMAX * NCOEF. XD is not needed and can be a array of length 1 in the calling program if NMAX = 0.

WK — Work vector of length 4 * NMAX. WK is not needed and can be a array of length 1 in the calling program if NMAX = 0.

KBASIS — Work vector of length 2 * NMAX. KBASIS is not needed and can be a array of length 1 in the calling program if NMAX = 0.

2 Informational errors

Type	Code	Description
3	1	There were too many iterations required. Convergence is assumed.
3	2	There were too many step halvings. Convergence is assumed.
4	3	The number of distinct values of the classification variables exceeds MAXCL.
4	4	The number of distinct values of a classification must be greater than one.
4	5	LDCOEF or LDCOV must be greater than or equal to NCOEF.
4	6	The number of observations to be deleted has exceeded NMAX. Rerun with a different model or increase the workspace.

3. Dummy variables are generated for the classification variables as follows: An ascending list of all distinct values of each classification variable is obtained and stored in CLVAL. Dummy variables are then generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case the dummy variable is one. See argument IDUMMY for IDUMMY = 2 in routine GRGLM in Chapter 2.

4. The "product" of a classification variable with a covariate yields dummy variables equal to the product of the covariate with each of the dummy variables associated with the classification variable.

5. The "product" of two classification variables yields dummy variables in the usual manner. Each dummy variable associated with the first classification variable multiplies each dummy variable associated with the second classification variable. The resulting dummy variables are such that the index of the second classification variable varies fastest.

Programming Notes

1. Classification variables are specified via arguments **NCLVAR** and **INDCL**. Indicator or dummy variables are created for the classification variables using routine **GRGLM** (see Chapter 2, "Regression") with **IDUMMY** = 2.

2. To enhance precision "centering" of covariates is performed if **INTCEP** = 1 and **NOBS - NRMISS** > 1. In doing so, the sample means of the design variables are subtracted from each observation prior to its inclusion in the model. On convergence the intercept, its variance and its covariance with the remaining estimates are transformed to the uncentered estimate values.

3 Two methods for specifying a binomial distribution model are possible. In the first method, **X**(**I**, **IFRQ**) contains the frequency of the observation while **X**(**I**, **IRT**) is 0 or 1 depending upon whether the observation is a success or failure. In this case, *N* = **X**(**I**, **IPAR**) is always 1. The model is treated as repeated Bernoulli trials, and interval observations are not possible.

   A second method for specifying binomial models is to use **X**(**I**, **IRT**) to represent the number of successes in the **X**(**I**, **IPAR**) trials. In this case, **X**(**I**, **IFRQ**) will usually be 1, but it may be greater than 1, in which case interval observations are possible.

   The estimated coefficients will be identical between the two methods, but the log-likelihood will differ by a constant term determined by the binomial coefficient. Specifically, if the model is treated as binomial trials, each observation with

$$y_i = \text{X(I, IRT)}, N_i = \text{X(I, IPAR)}, f_i = \text{X(I, IFRQ)}$$

   contributes

$$f_i \log\binom{N_i}{y_i} = f_i \log\frac{N_i!}{y_i!(N_i - y_i)!}$$

   to the log-likelihood. For Bernoulli data, this term evaluates to 0 for every observation.

Examples

Example

The first example is from Prentice (1976) and involves the mortality of beetles after exposure to various concentrations of carbon disulphide. Both a logit and a probit fit are produced for linear model

$$\mu + \beta x$$

The data is given as:

1.690	59	6
1.724	60	13
1.755	62	18
1.784	56	28
1.811	63	52
1.836	59	53
1.861	62	61
1.883	60	60

```
      USE UMACH_INT
      USE CTGLM_INT

      IMPLICIT   NONE
      INTEGER    IPAR, IRT, LDCASE, LDCOEF, LDCOV, LDX, NCOL, &
                 NEF, NOBS, NOUT, MAXCL
      REAL       EPS
      PARAMETER  (EPS=0.0001, IPAR=2, IRT=3, LDCASE=8, LDCOEF=2, LDCOV=2, &
                 LDX=8, MAXCL=1, NCOL=3, NEF=1, NOBS=8)
!
      INTEGER    INDCL(MAXCL), INDEF(1), IPRINT, MODEL, NCLVAL(1), &
                 NCOEF, NRMISS, NVEF(1)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(1), COEF(LDCOEF,4), &
                 COV(LDCOV,4), GR(2), X(LDX,NCOL), XMEAN(2)
!
      DATA NVEF/1/, INDEF/1/
      DATA X/1.690, 1.724, 1.755, 1.784, 1.811, 1.836, 1.861, 1.883, &
          59, 60, 62, 56, 63, 59, 62, 60, 6, 13, 18, 28, 52, 53, 61, &
          60/
!
      IPRINT = 2
      CALL UMACH(2, NOUT)
      DO 10  MODEL=3, 4
         WRITE(NOUT, *) 'MODEL=', MODEL
         CALL CTGLM (X, MODEL, NCOEF, COEF, COV, XMEAN, GR, IRT=IRT, &
         IPAR=IPAR, EPS=EPS, NEF=NEF, NVEF=NVEF, INDEF=INDEF, IPRINT=IPRINT)
      IPRINT = 1
   10 CONTINUE
!
      END
```

Output

```
 Model =  3

 Initial Estimates
      1       2
 -63.27   35.84
```

```
Method  Iteration  Step size  Maximum scaled        Log
                              coef. update      likelihood
   Q-N        0                                -20.31
   Q-N        1      1.0000      0.1387        -19.25
   N-R        2      1.0000      0.6112E-01    -18.89
   N-R        3      1.0000      0.7221E-01    -18.78
   N-R        4      1.0000      0.6362E-03    -18.78
   N-R        5      1.0000      0.3044E-06    -18.78

Log-likelihood       -18.77818

                  Coefficient Statistics
                      Standard    Asymptotic    Asymptotic
     Coefficient         Error   Z-statistic       P-value
 1        -60.76          5.19        -11.66          0.00
 2         34.30          2.92         11.76          0.00

 Asymptotic Coefficient Covariance
                 1             2
   1    0.2691E+02   -0.1512E+02
   2                  0.8505E+01

                            Case Analysis
                                    Residual                Standardized
      Predicted      Residual    Std. Error      Leverage      Residual
1         0.058         2.593         1.792         0.267         1.448
2         0.164         3.139         2.871         0.347         1.093
3         0.363        -4.498         3.786         0.311        -1.188
4         0.606        -5.952         3.656         0.232        -1.628
5         0.795         1.890         3.202         0.269         0.590
6         0.902        -0.195         2.288         0.238        -0.085
7         0.956         1.743         1.619         0.198         1.077
8         0.979         1.278         1.119         0.138         1.143

Last Coefficient Update
        1           2
1.104E-07  -2.295E-07

Covariate Means
    1.793


      Observation Codes
1   2   3   4   5   6   7   8
0   0   0   0   0   0   0   0

Number of Missing Values          0

Model =  4

Log-likelihood       -18.23232

                 Coefficient Statistics
                     Standard    Asymptotic    Asymptotic
    Coefficient         Error   Z-statistic       P-value
1        -34.94          2.64        -13.23          0.00
2         19.74          1.49         13.29          0.00
```

Note that the probit model yields a slightly smaller absolute log-likelihood and, thus, is preferred. For this data, a model based upon the log-log transformation function is even better. See Prentice (1976) for details.

Example 2

As a second example, the following data illustrate the Poisson model when all types of interval data are present. The example also illustrates the use of classification variables and the detection of potentially infinite estimates (which turn out here to be finite). These potential estimates lead to the two iteration summaries. The input data is

ILT	**IRT**	**ICEN**		
0	5	0	1	0
9	4	3	0	0
0	4	1	0	0
9	0	2	1	1
0	1	0	0	1

A linear model

$$\mu + \beta_1 x_1 + \beta_2 x_2$$

is fit where $x_1 = 1$ if the Class 1 variable is 0, $x_1 = 0$, otherwise, and the x_2 variable is similarly defined.

```
      USE CTGLM_INT

      IMPLICIT   NONE
      INTEGER    ICEN, IFIX, ILT, INFIN, IPAR, IPRINT, IRT,  &
                 LDCASE, LDCOEF, LDCOV, LDX, MAXCL, MAXIT, MODEL, &
                 NCLVAR, NCOL, NEF, NOBS
      REAL       EPS
      PARAMETER  (ICEN=3, ILT=1, INFIN=0, IPAR=2, IPRINT=2, &
                  IRT=2, LDCASE=5, LDCOEF=4, LDCOV=4, LDX=5, MAXCL=4, &
                  MODEL=0, NCLVAR=2, NCOL=5, NEF=2, NOBS=5)
!
      INTEGER    IADD(NOBS), INDCL(NCLVAR), INDEF(2), NCLVAL(MAXCL), &
                NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(4), COEF(LDCOEF,4), &
                COV(LDCOV,4), GR(5), X(LDX,NCOL), XMEAN(3)
!
      DATA INDCL/4, 5/, NVEF/1, 1/, INDEF/4, 5/
      DATA X/0, 9, 0, 9, 0, 5, 4, 4, 0, 1, 0, 3, 1, 2, 0, 1, 0, 0, 1, &
           0, 0, 0, 0, 1, 1/
!
      CALL CTGLM (X, MODEL, NCOEF, COEF, COV, XMEAN, GR, ILT=ILT, &
      IRT=IRT, IPAR=IPAR, ICEN=ICEN, INFIN=INFIN, &
      NCLVAR=NCLVAR, NCLVAL=NCLVAL, CLVAL=CLVAL, INDCL=INDCL, &
      NEF=NEF, NVEF=NVEF, INDEF=INDEF, IPRINT=IPRINT, MAXCL=MAXCL)
!
      END
```

Output

```
Initial Estimates
     1        2        3
0.2469   0.4463  -0.0645

Method  Iteration  Step size  Maximum scaled        Log
                              coef. update      likelihood
  Q-N         0                                -3.529
  Q-N         1      0.2500       5.168        -3.262
  N-R         2      0.0625       183.4        -3.134
  N-R         3      1.0000      0.7438        -3.006
  N-R         4      1.0000      0.2108        -3.005
  N-R         5      1.0000      0.5559E-02    -3.005

Method  Iteration  Step size  Maximum scaled        Log
                              coef. update      likelihood
  Q-N         0                                -3.529
  Q-N         1      0.2500       5.168        -3.262
  N-R         2      0.0625       183.4        -3.217
  N-R         3      1.0000       1.128        -3.116
  N-R         4      1.0000      0.1673        -3.115
  N-R         5      1.0000      0.4418E-02    -3.115

Log-likelihood       -3.114638

                 Coefficient Statistics
                    Standard    Asymptotic    Asymptotic
    Coefficient        Error   Z-statistic       P-value
1        -0.549        1.171        -0.517         0.605
2         0.549        0.610         0.900         0.368
3         0.549        1.083         0.507         0.612

     Asymptotic Coefficient Covariance
               1             2             3
1    0.1372E+01   -0.3719E+00   -0.1172E+01
2                  0.3719E+00    0.1719E+00
3                                0.1172E+01

                            Case Analysis
                                  Residual                Standardized
      Predicted      Residual    Std. Error      Leverage      Residual
1         5.000         0.000         2.236         1.000         0.000
2         6.925        -0.412         2.108         0.764        -0.196
3         6.925         0.412         1.173         0.236         0.351
4         0.000         0.000         0.000         0.000           NaN
5         1.000         0.000         1.000         1.000         0.000

      Last Coefficient Update
         1           2           3
-2.924E-07  -1.131E-08   7.075E-07

Covariate Means
     1       2
0.6000   0.6000

Distinct Values For Each Class Variable
Variable  1:      0.         1.0
Variable  2:      0.         1.0
```

```
    Observation Codes
    1   2   3   4   5
    0   0   0   0   0

    Number of Missing Values           0
```

CTWLS

more...

Performs a generalized linear least-squares analysis of transformed probabilities in a two-dimensional contingency table.

Required Arguments

TABLE — NRESP by NPOP matrix containing the frequency count in each cell of each population. (Input) The *i*-th column of TABLE contains the counts for the *i*-th population.

ITRAN — Vector of length NTRAN containing the transformation code for each of the NTRAN transformations to be applied. (Input)
ITRAN is not referenced and can be a vector of length 1 in the calling program if NTRAN = 0. Let a "response" denote a transformed cell probability. Then, ITRAN(1) contains the first transformation to be applied to the cell probabilities, ITRAN(2) contains the second transformation, which is to be applied to the responses obtained after ITRAN(1) is performed, etc. Note that the *k*-th transformation takes the ISIZE(*k* - 1) responses at step *k* into ISIZE(*k*) responses, where ISIZE(0) is taken to be NPOP * NRESP. Let *y* denote the vector result of a transformation, *x* denote the responses before the transformation is applied, *A* denote a matrix of constants, and *v* denote a vector of constants. Then, the possible transformations are

ITRAN	Transformation
1	Linear, defined over all populations ($y = Ax$)
2	Logarithmic ($y(i, j) = \ln(x(i, j))$)
3	Exponential ($y(i, j) = \exp(x(i, j))$)
4	Additive ($y(i, j) = y(i, j) + v(i, j)$)
5	Linear, defined for one population and, identically, applied over all populations ($y(i) = Ax(i)$)

where $y(i)$ and $x(i)$ are the subvectors for the *i*-th population, $y(i, j)$ and $x(i, j)$ denote the *j*-th response in the *i*-th population, and $v(i, j)$ denotes the corresponding element of the vector "*v*". Transformation type 5 is the same as transformation type 1 when the same linear transformation is applied in each

population (i.e., the type 1 matrix is block diagonal with identical blocks). Because the size of the type 5 transformation matrix *A* is NPOP^2 times smaller than the type 1 transformation matrix, the type 5 transformation is usually preferred where it can be used.

ISIZE — Vector of length NTRAN containing the number of response functions defined by the *k*-th transformation. (Input)
Transformation types 2, 3, and 4 have the same number of output responses as are input, and elements of ISIZE corresponding to transformations of these types should reflect this fact. Transformation types 1 and 5 can either increase or, more commonly, decrease the number of responses. For transformation type 5, if *m* linear transformations are defined for each population, the corresponding element of ISIZE should be *m* * NPOP.

AMATS — Vector containing the transformation constants. (Input)
AMATS contains the transformation matrices and vectors needed in the type 1, 4 and 5 transformations. While AMATS is a vector, its elements may be treated as a number of matrices or vectors where the number of structures depends upon the transformation types as follows:

ITRAN	Type	Dimension	Length
1	Matrix	*m* by *n*	*M* * *n*
2, 3	Not referenced		0
4	Vector	*M*	*M*
5	Matrix	*m*/NPOP by *n*/NPOP	*M* * *n*/(NPOP * NPOP)

Here, *m* = ISIZE(*i*) and *n* = ISIZE(*i* - 1), and ISIZE(0) is not input (in ISIZE) but is taken to be NPOP * NRESP. Matrices and vectors are stored consecutively in AMATS with column elements for matrices stored consecutively as is standard in FORTRAN. Thus, if ITRAN(1) = 5 and ITRAN(2) = 4, with NREP = 3, NPOP= 2, and ISIZE(1) = ISIZE(2) = 2, then the vector AMATS would contain in consecutive positions *A*(1, 1), *A*(2, 1), *A*(1, 2), *A*(2, 2), *A*(1, 3), *A*(2, 3), *v*(1), *v*(2), *v*(3), *v*(4) where *A* is the matrix for transformation type 5 and *v* is the vector for transformation type 4.

X — Design matrix of size ISIZE(NTRAN) by NCOEF. (Input, if NCOEF > 0)
X contains the design matrix for predicting the transformed cell probabilities F from the covariates stored in X. If NCOEF = 0, X is not referenced and can be a 1 by 1 matrix in the calling program.

NH — Vector of length NUMH. (Input, if NCOEF > 0)
NH(*i*) contains the number of consecutive rows in H used to specify hypothesis *i*. If NCOEF = 0, NH is not referenced and can be a vector of length 1 in the calling program.

H — Matrix of size *m* by NCOEF containing the constants to be used in the multivariate hypothesis tests. (Input, if NCOEF > 0)
Here, *m* is the sum of the elements in NH. Each hypothesis is of the form
H_0 : C * COEF = 0, where C for the *i*-th hypothesis is NH(*i*) rows of H, and COEF is estimated in the linear model. The first NH(1) rows of H make up the first hypothesis, the next NH(2) rows make up the second hypothesis, etc. If NCOEF = 0, H is not referenced and can be a 1 by 1 matrix in the calling program.

CHSQ — NUMH + 1 by 3 matrix containing the results of the hypothesis tests. (Output, if NCOEF > 0)
The first row of CHSQ contains the results for test 1, the next row contains the results for test 2, etc. The last row of CHSQ contains a test of the adequacy of the model. Within each row, the first column contains the chi-squared statistic, the second column contains its degrees of freedom, and the last column contains the probability of a larger chi-squared. If NCOEF = 0, CHSQ is not referenced and can be a 1 by 1 matrix in the calling program.

COEF — NCOEF by 4 matrix containing the coefficient estimates and related statistics. (Output, if NCOEF > 0)
The columns of coefficient are as follows:

Col	Statistic
1	Coefficient estimate
2	Estimated standard error of the coefficient
3	*z*-statistic for a test that the coefficient equals 0 versus the Two-sided alternative
4	*p*-value corresponding to the *z*-statistic

If NCOEF = 0, COEF is not referenced and can be a 1 by 1 matrix in the calling program.

COVCF — NCOEF by NCOEF matrix containing the estimated variances and covariances of COEF. (Output, if NCOEF > 0)
If NCOEF = 0, COVCF is not referenced and can be a 1 by 1 matrix in the calling program.

F — Vector of length ISIZE(NTRAN) containing the transformed probabilities, the responses. (Output)

COVF — Matrix of size ISIZE(NTRAN) by ISIZE(NTRAN) containing the estimated variances and covariances of F. (Output)

RESID — ISIZE(NTRAN) by 4 matrix containing a case analysis for the transformed probabilities as estimated by the linear model. (Output, if NCOEF > 0)
The linear model gives F = X * BETA. The columns of RESID are as follows:

Col	Description
1	Residual
2	Standard error
3	Leverage
4	Standardized residual

If NCOEF = 0, RESID is not referenced and can be a 1 by 1 matrix in the calling program.

Optional Arguments

NRESP — Number of cells in each population. (Input)
Default: NRESP = size (TABLE,1).

NPOP — Number of populations. (Input)
Default: NPOP = size (TABLE,2).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

NTRAN — Number of transformations to be applied to the cell probabilities. (Input)
Cell probabilities are computed as the frequency count for the cell divided by the population sample size. Set NTRAN = 0 if a linear model predicting the cell probabilities is to be used.
Default: NTRAN = size (ITRAN,1).

NCOEF — Number of coefficients in the linear model relating the transformed probabilities F to the design matrix X. (Input)
Let F denote the vector result of applying the NTRAN transformations, and assume that the model gives F = X * COEF. Then, NCOEF is the length of COEF.
Default: NCOEF = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NUMH — Number of multivariate hypotheses to be tested on the coefficients in COEF. (Input, if NCOEF > 0)
If NCOEF = 0, NUMH is not referenced.
Default: NUMH = size (NH,1).

LDH — Leading dimension of **H** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDH** = size(**H**,1).

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	Print all output arrays and vectors.
2	Print all output arrays and vectors as well as the matrices and vectors in AMATS.

LDCHSQ — Leading dimension of **CHSQ** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCHSQ** = size (**CHSQ**,1).

LDCOEF — Leading dimension of **COEF** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOEF** = size (**COEF**,1).

LDCOVC — Leading dimension of **COVCF** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOVC** = size (**COVCF**,1)

LDCOVF — Leading dimension of **COVF** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOVF** = size (**COVF**,1).

LDRESI — Leading dimension of **RESID** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDRESI** = size (**RESID**,1).

FORTRAN 90 Interface

Generic: **CALL CTWLS** (**TABLE**, **ITRAN**, **ISIZE**, **AMATS**, **X**, **NH**, **H**, **CHSQ**, **COEF**, **COVCF**, **F**, **COVF**, **RESID** [, ...])

Specific: The specific interface names are **S_CTWLS** and **D_CTWLS**.

FORTRAN 77 Interface

Single: CALL CTWLS (NRESP, NPOP, TABLE, LDTABL, NTRAN, ITRAN, ISIZE, AMATS, NCOEF, X, LDX, NUMH, NH, H, LDH, IPRINT, CHSQ, LDCHSQ, COEF, LDCOEF, COVCF, LDCOVC, F, COVF, LDCOVF, RESID, LDRESI)

Double: The double precision name is DCTWLS.

Description

Routine CTWLS performs weighted least-squares analysis of a general p = NPOP population by r = NRESP response categories per population contingency table. After division by the sample size, there are $n = pr$ cell probabilities.

Define s = ISIZE(NTRAN) responses f_i such that each response is obtained from the cell probabilities as $f_i = g_i(p_1, p_2, \ldots, p_n)$, for $i = 1, \ldots, s$. Call the functions g_i the response functions. Then, if

$$\hat{\Sigma}_f$$

is the asymptotic covariance matrix of the responses, and **X** is a design matrix for a linear model predicting $f = X\,\boldsymbol{\beta}$ with q = NCOEF coefficients **β** = COEF, then CTWLS performs a weighted least-squares analysis of the model $f = X\,\boldsymbol{\beta}$ where the generalized weights are given by

$$\hat{\Sigma}_f = \mathrm{COVF}$$

Estimates obtained in this way are best asymptotic normal estimates of **β**.

Let

$$\hat{\Sigma}_p$$

denote the estimated variance-covariance matrix of the estimated cell probabilities, and let $(\partial g_i/\partial p_j)$ denote the matrix of partial derivatives of g_i with respect to p_j. Then,

$$\hat{\Sigma}_f$$

is given by

$$\hat{\Sigma}_f = \left(\frac{\partial g_i}{\partial p_j}\right)\hat{\Sigma}_p\left(\frac{\partial g_i}{\partial p_j}\right)^T$$

where the (i, j)-th element in

$$\hat{\Sigma}_p$$

is computed as

$$p_i(\delta_{ij} - p_j)$$

Here, $\delta_{ij} = 1$ if $i = j$ and is zero otherwise.

In **CTWLS**, the transformations g_i are defined by successive application of one of five types of simpler transformations. Let $p_i = h_{0,j}$ for $j = 1, \ldots, n$ denote the n cell probabilities, and let $h_{i,j}$ denote the **ISIZE**(i) responses obtained after i simple transformations have been performed with h_i denoting the corresponding vector of estimates. Then, the simple transformations are defined by:

1. Linear: $h_{i+1} = A_i h_i$ where A_i is a matrix of coefficients specified via the vector **AMATS** in **CTWLS**.
2. Logarithmic: $h_{i+1,j} = \ln(h_{i,j})$ where $j = 1, \ldots,$ **ISIZE**(i). That is, take the logarithm of each of the responses.
3. Exponential: $h_{i+1,j} = \exp(h_{i,j})$ where $j = 1, \ldots,$ **ISIZE**(i). That is, take the exponential of each of the responses.
4. Additive: $h_{i+1,j} = h_{i,j} + v_j$, where $j = 1, \ldots,$ **ISIZE**(i), and v_j is specified via the vector **AMATS** in **CTWLS**. Additive transformations are generally used to adjust for zero cells or to apply a continuity correction to the cell probabilities.
5. Linear (by population):

$$h_{i+1}^j = A_i h_i^j \text{ where } h_i^j$$

is the vector of responses at stage i in the j-th population, and A_i is a matrix of coefficients specified via **AMATS**.

Given the responses f_i and their covariances

$$\hat{\Sigma}_f$$

estimates for β are computed via generalized least squares as

$$\hat{\beta} = \left(X^T \hat{\Sigma}_f^{-1} X \right)^{-1} X^T \hat{\Sigma}_f^{-1} f$$

Let Σ_β denote the asymptotic covariance matrix of β. Then, Σ_β is estimated by

$$\hat{\Sigma}_{\beta} = \left(X^T \hat{\Sigma}_f^{-1} X \right)^{-1}$$

Hypothesis tests of the form $H_0 : C_i \boldsymbol{\beta} = 0$ are performed when requested. Here, C_i is a matrix of coefficients specified via a submatrix of the matrix **H**. Results are returned in the vector **CHSQ**. The asymptotic chi-squared test for testing the null hypothesis is given by

$$\chi^2 = \left(C_i \beta \right)^T \left(C_i \hat{\Sigma} \beta \right)^{-1} C_i \beta$$

This test has q_i = rank(C_i) degrees of freedom. If zero degrees of freedom are returned, the hypothesis cannot be tested in the original parameterization.

A test of the model checks that the residuals obtained from the model $f = X \boldsymbol{\beta}$ are not too large. This test, which has $s - q$ degrees of freedom, is an asymptotic chi-squared test and is computed as

$$Q = \left(f - X\hat{\beta} \right)^T \left(\hat{\Sigma}_f \right)^{-1} \left(f - X\hat{\beta} \right)$$

Residuals from the generalized linear model are easily computed as

$$r_i = f_i - x_i \hat{\beta}$$

where x_i is the row of the design matrix X corresponding to the i-th observation. This residual has the asymptotic variance

$$\hat{\sigma}_i^2 = \left(\hat{\Sigma}_f \right)_{ii} \left(1 - \left(X \left(X^T \hat{\Sigma}_f X \right)^{-1} X^T \right)_{ii} \right)$$

where $(A)_{ii}$ denotes the i-th diagonal element of matrix A. A standardized residual is then computed as

$$z = r_i / \hat{\sigma}_i$$

which has an asymptotic standard normal distribution if the model is correct.

The leverage of observation i, v_i, is computed as

$$v_i = \left(X \left(X^T \hat{\Sigma}_f^{-1} X \right)^{-1} X^T \hat{\Sigma}_f^{-1} \right)_{ii}$$

It is a measure of the importance of the observation in the predicted values. Values greater than $2q/s$ are large.

Because the tests performed by **CTWLS** are asymptotic ones, the user should treat the results with caution. The reported asymptotic p-values are most likely to be exact when the number of counts in each cell is large (say 5 or more), and less exact for smaller cell counts. Care should also be taken to avoid illegal operations. For example,

the routine returns an error message when the log of a negative or zero value is attempted. When this occurs, the user should either use a continuity correction (i.e. modify the transformations used by adding a constant to all cells or to the cell resulting in the illegal operation) or abandon the model.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2WLS/DC2WLS**. The reference is:

   ```
   CALL C2WLS (NRESP, NPOP, TABLE, LDTABL, NTRAN, ITRAN, ISIZE, AMATS, NCOEF, X,
      LDX, NUMH, NH, H, LDH, IPRINT, CHSQ, LDCHSQ, COEF, LDCOEF, COVCF, LDCOVC, F,
      COVF, LDCOVF, RESID, LDRESI, PDER, FRQ, EST, XX, WK, IWK, WWK)
   ```

   The additional arguments are as follows:

   PDER — Work vector of length **ISIZE**(**NTRAN**) * max(**NPOP** * **NRESP**, **ISIZE**(*i*)) if **NTRAN** is greater than zero. **PDER** is not used and can be dimensioned of length 1 if **NTRAN** = 0.

   FRQ — Work vector of length **NPOP**.

   EST — Work vector of length **NPOP** * **NRESP** + **ISIZE**(1) + … + **ISIZE**(**NTRAN**).

   XX — Work vector of length (**NCOEF** + 1) * **ISIZE**(**NTRAN**) if **NCOEF** is greater than zero. If **NCOEF** = 0, **XX** is not referenced and can be a vector of length 1 in the calling program.

   WK — Work vector of length 3(max(**NPOP** * **NRESP**, **ISIZE**(*i*))) + **NCOEF** + 1.

   IWK — Work vector of length max(**NH**(*i*)) if **NUMH** is greater than 0. If **NCOEF** = 0, **IWK** is not referenced and can be a vector of length 1 in the calling program.

   WWK — Work vector of length max(**NH**(*i*)) * (4 + **NCOEF** + max(**NCOEF**, max(**NH**(*i*))) if **NUMH** is greater than 0. If **NUMH** = 0, **WWK** is not referenced and can be a vector of length 1 in the calling program.

2. Informational error

Type	Code	Description
4	1	A negative response occurred while performing a logarithmic transformation. The logarithm of a negative number is not allowed.

Examples

Example 1

This example is taken from Landis, Stanish, Freeman, and Koch (1976), pages 213-217. Generalized kappa statistics are computed via vector functions of the form:

$$F(p) = \exp(A_4 \ln(A_3 \exp(A_2 \ln(A_1 p))))$$

where p is the cell probabilities. The raw frequencies are given as two 4 × 4 contingency tables. These tables are reorganized as a single 16 × 2 table for input into **CTWLS**. The input tables are

$$\begin{pmatrix} 38 & 5 & 0 & 1 \\ 33 & 11 & 3 & 0 \\ 10 & 14 & 5 & 6 \\ 3 & 7 & 3 & 10 \end{pmatrix} \begin{pmatrix} 5 & 3 & 0 & 0 \\ 3 & 11 & 4 & 0 \\ 2 & 13 & 3 & 4 \\ 1 & 2 & 4 & 14 \end{pmatrix}$$

Two generalized kappa statistics using two different sets of weights are computed for each population. Hypothesis tests are then performed on the four resulting generalized kappa statistics. In this example, the matrix of covariates is an identity matrix so that tests on the responses are performed.

```
      USE CTWLS_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCHSQ, LDCOEF, LDCOVC, LDCOVF, LDH, LDRESI, &
                 LDTABL, LDX, NCOEF, NPOP, NTRAN
      PARAMETER  (IPRINT=2, LDCHSQ=10, LDCOEF=4, LDCOVC=4, LDCOVF=4, &
                 LDH=10, LDRESI=4, LDTABL=16, LDX=4, NCOEF=4, NPOP=2, &
                 NTRAN=8)
!
      INTEGER    ISIZE(NTRAN), ITRAN(NTRAN), NH(9)
      REAL       A1(10,16), A2(18,10), A3(4,18), A4(2,4), AMATS(420), &
                 CHSQ(LDCHSQ,3), COEF(LDCOEF,4), COVCF(LDCOVC,NCOEF), &
                 COVF(LDCOVF,LDCOVF), F(LDX), H(LDH,4), &
                 RESID(LDRESI,4), TABLE(LDTABL,NPOP), X(LDX,NCOEF)
!
      EQUIVALENCE (A1, AMATS(1)), (A2, AMATS(161)), (A3, AMATS(341)), &
                (A4, AMATS(413))
!
      DATA TABLE/38, 5, 0, 1, 33, 11, 3, 0, 10, 14, 5, 6, 3, 7, 3, 10, &
          5, 3, 0, 0, 3, 11, 4, 0, 2, 13, 3, 4, 1, 2, 4, 14/
      DATA X/1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1/
      DATA NH/1, 1, 1, 1, 1, 1, 2, 1, 1/
      DATA H/1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, -1, 0, 0, 0, 0, 1, 0, &
          1, 0, 0, 0, 1, 0, 1, -1, 0, -1, 0, 0, 0, 0, 0, 1, -1, 0, &
          -1, 0, -1/
      DATA ITRAN/5, 2, 5, 3, 5, 2, 5, 3/
      DATA ISIZE/20, 20, 36, 36, 8, 8, 4, 4/
      DATA A1/1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, &
          .5, 1, 0, 0, 0, 0, 0, 1, 0, 0, .25, 1, 0, 0, 0, 0, 0, 0, 1,&
          0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, .5, 0, 1, 0, 0, 0, 1, 0, &
          0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, .5, 0, 1, 0, 0, 0, 0, &
          0, 1, 0, .25, 0, 0, 1, 0, 1, 0, 0, 0, 0, .25, 0, 0, 1, 0, &
          0, 1, 0, 0, 0, .5, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, &
          0, 0, 0, 0, 1, 0, .5, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, &
          0, 1, 0, 1, 0, 0, 0, .25, 0, 0, 0, 1, 0, 0, 1, 0, 0, .5, 0, &
          0, 0, 1, 0, 0, 0, 1, 1, 1/
      DATA A2/1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,&
          0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
          0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
          0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, &
          0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, &
          1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, &
          0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, &
          0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
          0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, &
          1/
      DATA A3/-1, -1, 0, 0, 0, -.5, 1, .5, 0, -.25, 1, .75, 0, 0, 1, &
```

```
           1, 0, -.5, 1, .5, -1, -1, 0, 0, 0, -.5, 1, .5, 0, -.25, 1, &
           .75, 0, -.25, 1, .75, 0, -.5, 1, .5, -1, -1, 0, 0, 0, -.5, &
           1, .5, 0, 0, 1, 1, 0, -.25, 1, .75, 0, -.5, 1, .5, -1, -1, &
           0, 0, 1, 0, 0, 0, 0, 1, 0, 0/
      DATA A4/1, 0, 0, 1, -1, 0, 0, -1/
!
      CALL CTWLS (TABLE, ITRAN, ISIZE, AMATS, X, NH, H, &
                  CHSQ, COEF, COVCF, F, COVF, RESID, IPRINT=IPRINT)
!
      END
```

Output

```
 Hypothesis Tests on Coefficients
 H-1
         1          0          0          0
 H-2
         0          1          0          0
 H-3
         1         -1          0          0
 H-4
         0          0          1          0
 H-5
         0          0          0          1
 H-6
         0          0          1         -1
 H-7
         1          0         -1          0
         0          1          0         -1
 H-8
         1          0         -1          0
 H-9
         0          1          0         -1

          Hypothesis Chi-Squared Statistics
                            Degrees of
 Hypothesis  Chi-Squared     freedom       p-value
          1        16.99           1        0.0000
          2        39.70           1        0.0000
          3        39.54           1        0.0000
          4        14.27           1        0.0002
          5        30.07           1        0.0000
          6        28.76           1        0.0000
          7         1.07           2        0.5850
          8         0.90           1        0.3425
          9         1.06           1        0.3040

                         Degrees of
             Chi-Squared     freedom       p-value
 Model Test         0.00           0           NaN

                 Coefficient Statistics
     Coefficient  Standard Error   Statistic       p-value
 1        0.2079            0.05        4.12        0.0000
 2        0.3150            0.05        6.30        0.0000
 3        0.2965            0.08        3.78        0.0002
 4        0.4069            0.07        5.48        0.0000

            Asymptotic Coefficient Covariance
```

```
                1             2             3             4
1    2.5457E-03    2.3774E-03        0.            0.
2                  2.4988E-03        0.            0.
3                                 6.1629E-03    5.6229E-03
4                                               5.5069E-03

                    Residual Analysis
                     Standard                 Standardized
       Residual         Error      Leverage      Residual
1        0.0000        0.0000        1.0000           NaN
2        0.0000        0.0000        1.0000           NaN
3        0.0000        0.0000        1.0000           NaN
4        0.0000        0.0000        1.0000           NaN

Transformed Probabilities
       1   0.2079
       2   0.3150
       3   0.2965
       4   0.4069

Asymptotic Covariance of the Transformed Probabilities
                1             2             3             4
1    2.5457E-03    2.3774E-03        0.            0.
2                  2.4988E-03        0.            0.
3                                 6.1629E-03    5.6229E-03
4                                               5.5069E-03

     Linear transformation matrix, by population, for transformation 5
          1       2       3       4       5       6       7       8       9
 1   1.000   1.000   1.000   1.000   0.000   0.000   0.000   0.000   0.000
 2   0.000   0.000   0.000   0.000   1.000   1.000   1.000   1.000   0.000
 3   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   1.000
 4   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000
 5   1.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000   1.000
 6   0.000   1.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000
 7   0.000   0.000   1.000   0.000   0.000   0.000   1.000   0.000   0.000
 8   0.000   0.000   0.000   1.000   0.000   0.000   0.000   1.000   0.000
 9   1.000   0.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000
10   1.000   0.500   0.250   0.000   0.500   1.000   0.500   0.250   0.250

         10      11      12      13      14      15      16
 1   0.000   0.000   0.000   0.000   0.000   0.000   0.000
 2   0.000   0.000   0.000   0.000   0.000   0.000   0.000
 3   1.000   1.000   1.000   0.000   0.000   0.000   0.000
 4   0.000   0.000   0.000   1.000   1.000   1.000   1.000
 5   0.000   0.000   0.000   1.000   0.000   0.000   0.000
 6   1.000   0.000   0.000   0.000   1.000   0.000   0.000
 7   0.000   1.000   0.000   0.000   0.000   1.000   0.000
 8   0.000   0.000   1.000   0.000   0.000   0.000   1.000
 9   0.000   1.000   0.000   0.000   0.000   0.000   1.000
10   0.500   1.000   0.500   0.000   0.250   0.500   1.000

     Linear transformation matrix, by population, for transformation 5
          1       2       3       4       5       6       7       8       9
 1   1.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000   0.000
 2   1.000   0.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000
 3   1.000   0.000   0.000   0.000   0.000   0.000   1.000   0.000   0.000
 4   1.000   0.000   0.000   0.000   0.000   0.000   0.000   1.000   0.000
 5   0.000   1.000   0.000   0.000   1.000   0.000   0.000   0.000   0.000
 6   0.000   1.000   0.000   0.000   0.000   1.000   0.000   0.000   0.000
```

```
      7   0.000   1.000   0.000   0.000   0.000   0.000   1.000   0.000   0.000
      8   0.000   1.000   0.000   0.000   0.000   0.000   0.000   1.000   0.000
      9   0.000   0.000   1.000   0.000   1.000   0.000   0.000   0.000   0.000
     10   0.000   0.000   1.000   0.000   0.000   1.000   0.000   0.000   0.000
     11   0.000   0.000   1.000   0.000   0.000   0.000   1.000   0.000   0.000
     12   0.000   0.000   1.000   0.000   0.000   0.000   0.000   1.000   0.000
     13   0.000   0.000   0.000   1.000   1.000   0.000   0.000   0.000   0.000
     14   0.000   0.000   0.000   1.000   0.000   1.000   0.000   0.000   0.000
     15   0.000   0.000   0.000   1.000   0.000   0.000   1.000   0.000   0.000
     16   0.000   0.000   0.000   1.000   0.000   0.000   0.000   1.000   0.000
     17   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   1.000
     18   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000   0.000

           10
      1   0.000
      2   0.000
      3   0.000
      4   0.000
      5   0.000
      6   0.000
      7   0.000
      8   0.000
      9   0.000
     10   0.000
     11   0.000
     12   0.000
     13   0.000
     14   0.000
     15   0.000
     16   0.000
     17   0.000
     18   1.000

     Linear transformation matrix, by population, for transformation 5
            1       2       3       4       5       6       7       8       9
  1  -1.000   0.000   0.000   0.000   0.000  -1.000   0.000   0.000   0.000
  2  -1.000  -0.500  -0.250   0.000  -0.500  -1.000  -0.500  -0.250  -0.250
  3   0.000   1.000   1.000   1.000   1.000   0.000   1.000   1.000   1.000
  4   0.000   0.500   0.750   1.000   0.500   0.000   0.500   0.750   0.750

           10      11      12      13      14      15      16      17      18
  1   0.000  -1.000   0.000   0.000   0.000   0.000  -1.000   1.000   0.000
  2  -0.500  -1.000  -0.500   0.000  -0.250  -0.500  -1.000   0.000   1.000
  3   1.000   0.000   1.000   1.000   1.000   1.000   0.000   0.000   0.000
  4   0.500   0.000   0.500   1.000   0.750   0.500   0.000   0.000   0.000

  Linear transformation matrix, by population, for transformation 5
                        1       2       3       4
                  1   1.000   0.000  -1.000   0.000
                  2   0.000   1.000   0.000  -1.000
```

Example 2

The second example is taken from Prentice (1976) and involves a logistic fit to the mortality of beetles after exposure to various concentrations of carbon disulphide. Because one of the cells on input has a count of zero and it is not possible to take the logarithm of zero, a constant 0.5 is added to each cell prior to calling **CTWLS**. The model can be expressed as

$$\ln \frac{p_{i1}}{p_{i2}} = \mu + \beta_1 x$$

where *i* indexes the 8 populations. The data is given as:

1.690	6	53
1.724	13	47
1.755	18	44
1.784	28	28
1.811	52	11
1.836	53	6
1.861	61	1
1.883	60	0

For comparison, a maximum fit yields

$$\hat{\mu} = .74 \text{ and } \hat{\beta} = 34.3$$

(see routine **CTGLM**).

```
      USE CTWLS_INT

      IMPLICIT     NONE
      INTEGER    IPRINT, LDCHSQ, LDCOEF, LDCOVC, LDCOVF, LDH, LDRESI, &
                 LDTABL, LDX, NCOEF, NPOP, NRESP, NTRAN, NUMH
      PARAMETER  (IPRINT=2, LDCOVF=8, LDH=1, LDX=8, NCOEF=2, NPOP=8,  &
                 NRESP=2, NTRAN=2, NUMH=0, LDCHSQ=NUMH+1, &
                 LDCOEF=NCOEF, LDCOVC=NCOEF, LDRESI=LDX, LDTABL=NRESP)
!
      INTEGER    ISIZE(NTRAN), ITRAN(NTRAN), NH(1)
      REAL       AMATS(2), CHSQ(LDCHSQ,3), COEF(LDCOEF,4), &
                 COVCF(LDCOVC,NCOEF), COVF(LDCOVF,LDCOVF), F(LDX),  &
                 H(LDH,4), RESID(LDRESI,4), TABLE(LDTABL,NPOP),  &
                 X(LDX,NCOEF)
!
      DATA TABLE/6, 53, 13, 47, 18, 44, 28, 28, 52, 11, 53, 6, 61, 1, &
          60, 0/, ITRAN/2, 5/, ISIZE/16, 8/, AMATS/1, -1/
      DATA X/8*1, 1.690, 1.724, 1.755, 1.784, 1.811, 1.836, 1.861, &
          1.883/
!
      TABLE = TABLE + 0.5
!
      CALL CTWLS (TABLE, ITRAN, ISIZE, AMATS, X, NH, H,  &
                  CHSQ, COEF, COVCF, F, COVF, RESID, NUMH=NUMH, &
                  IPRINT=IPRINT)
!
      END
```

Output

```
         Test of the Model
               Degrees of
Chi-Squared     freedom       p-value
       8.43           6        0.2081

                 Coefficient Statistics
    Coefficient  Standard Error   Statistic       p-value
1      -55.6590            5.02      -11.10        0.0000
2       31.4177            2.83       11.09        0.0000

 Asymptotic Coefficient Covariance
           1             2
1     25.16        -14.20
2                   8.024

                     Residual Analysis
                      Standard                Standardized
        Residual         Error      Leverage      Residual
 1        0.4552        0.3232        0.6052        1.4086
 2        0.2368        0.2480        0.6468        0.9548
 3       -0.3568        0.2413        0.7608       -1.4787
 4       -0.3902        0.2285        0.7440       -1.7076
 5        0.2800        0.2761        0.7192        1.0141
 6        0.0840        0.3484        0.7036        0.2410
 7        0.9042        0.7749        0.8791        1.1670
 8        1.2953        1.3777        0.9413        0.9402

 Transformed Probabilities
         1  -2.108
         2  -1.258
         3  -0.878
         4   0.000
         5   1.518
         6   2.108
         7   3.714
         8   4.796

          Asymptotic Covariance of the Transformed Probabilities
           1                 2            3             4             5
1    0.1725            0.            0.            0.            0.
2                  9.5127E-02        0.            0.            0.
3                                 7.6526E-02        0.            0.
4                                                7.0175E-02        0.
5                                                              0.1060

             6             7             8
1        0.            0.            0.
2        0.            0.            0.
3        0.            0.            0.
4        0.            0.            0.
5        0.            0.            0.
6    0.1725            0.            0.
7                  0.6829            0.
8                                 2.017

Linear transformation matrix, by population, for transformation 5
                              1       2
```

```
                                  1.000  -1.000
```

Nonparametric Statistics

Routines

Usage Notes

Other Chapters

Much of what is considered nonparametric statistics is included in other chapters. Topics of possible interest in other chapters are: nonparametric measures of location and scale (Chapter 1, "Basic Statistics"), quantile estimation (Chapter 1, "Basic Statistics"), nonparametric measures in a contingency table (Chapter 5, "Categorical and Discrete Data Analysis"), measures of correlation in a contingency table (Chapter 3, "Correlation"), tests of goodness of fit and randomness (Chapter 7, "Tests of Goodness of Fit and Randomness"), and nonparametric routines for density and hazard estimation (Chapter 15, "Density and Hazard Estimation").

Other Methods

Many of the tests described in this chapter may be computed using the routines described in other chapters after first substituting ranks (or some other score) for the observed values. (Routine **RANKS**(see Chapter 1, "Basic Statistics") may be used to compute ranks.) This method for computing nonparametric test statistics is recommended for cases such as unbalanced one-way ANOVA designs for which no nonparametric subroutine is provided.

Missing Values

Most routines described in this chapter automatically handle missing values (NaN, "not a number"; see the Reference Material section of this manual). In these routines, observations that are missing are ignored; the variable **NMISS** is incremented by one for each missing observation. The user should be aware, however, that some routines described in this chapter do not handle missing values. Missing values input to such routines may result in erroneous results.

Tied Observations

Many of the routines described in this chapter contain an argument **FUZZ** in the input. Observations that are within **FUZZ** of each other in absolute value are said to be tied. Moreover, in some routines, an observation within **FUZZ** of some value is said to be equal to that value. In routine **SNRNK**, for example, such observations are eliminated from the analysis. If **FUZZ** = 0.0, observations must be identically equal before they are considered to be tied. Other positive values of **FUZZ** allow for numerical imprecision or roundoff error.

SIGNT

Performs a sign test of the hypothesis that a given value is a specified quantile of a distribution.

Required Arguments

X — Vector of length NOBS containing the input data. (Input)

Q — Hypothesized percentile of the population from which X was drawn. (Input)

P — Value in the range (0, 1). (Input)
Q is the 100 * P percentile of the population.

NPOS — Number of positive differences X(*j*) - Q, for *j* = 1, 2, ..., NOBS. (Output)

NTIE — Number of zero differences (ties) X(*j*) - Q, for *j* = 1, 2, ..., NOBS. (Output)

PROB — Binomial probability of NPOS or more positive differences in NOBS - NTIE - NMISS trials. (Output)

NMISS — Number of missing values in X. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

FORTRAN 90 Interface

Generic: CALL SIGNT (X, Q, P, NPOS, NTIE, PROB, NMISS [, ...])
Specific: The specific interface names are S_SIGNT and D_SIGNT.

FORTRAN 77 Interface

Single: CALL SIGNT (NOBS, X, Q, P, NPOS, NTIE, PROB, NMISS)
Double: The double precision name is DSIGNT.

Description

Routine **SIGNT** tests hypotheses about the proportion **P** of a population that lies below a value **Q**. In continuous distributions, this can be a test that **Q** is the 100**P**-th percentile of the population from which **X** was obtained. To carry out testing, **SIGNT** tallies the number of values above **Q** in **NPOS**. The binomial probability of **NPOS** or more values above **Q** is then computed using the proportion **P** and the sample size **NOBS** (adjusted for the missing observations [**NMISS**] and ties [**NTIE**]).

Hypothesis testing is performed as follows for the usual null and alternative hypotheses.

- H_0 : Pr(**X** ≤ **Q**) ≤ **P** (the **P**-th quantile is at least **Q**)
  H_1: Pr(**X** ≤ **Q**) > **P**
  Reject H_0 if **PROB** is less than or equal to the significance level.
- H_0 : Pr(**X** ≤ **Q**) ≥ **P** (the **P**-th quantile is no greater than **Q**)
  H_1 : Pr(**X** ≤ **Q**) < **P**
  Reject H_0 if **PROB** is greater than or equal to one minus the significance level.
- H_0 : Pr(**X** = **Q**) = **P**(the **P**-th quantile is **Q**)
  H_1 : Pr(**X** ≤ **Q**) < **P** or Pr(**X** ≤ **Q**) > **P**
  Reject H_0 if **PROB** is less than or equal to half the significance level or greater than or equal to one minus half the significance level.

The assumptions are as follows:

1. The $\mathbf{X}_i$ are a random sample; i.e., they are independent and identically distributed.
2. The measurement scale is at least ordinal; i.e, an ordering less than, greater than, and equal to exists in the observations.

Many uses for the sign test are possible with various values of **P** and **Q**. For example, to perform a matched sample test that the difference of the medians of *Y* and *Z* is 0.0, let *P* = 0.5, *q* = 0.0, and $X_i = Y_i - Z_i$ in matched observations *Y* and *Z*. To test that the median difference is *C*, let *Q* = *C*.

Comments

Other probabilities that may be of interest can be computed via routine **BINDF** (see Chapter 17, "Probability Distribution Functions and Inverses").

Example

We wish to test the hypothesis that at least 75% of a population is negative. Because 0.923 < 0.95, we fail to reject the null hypothesis at the 5 percent level of significance.

```
      USE SIGNT_INT
      USE UMACH_INT
      INTEGER    NOBS
      REAL       P, Q
      PARAMETER  (NOBS=19, P=0.75, Q=0.0)
!
      INTEGER    NMISS, NOUT, NPOS, NTIE
      REAL       PROB, X(NOBS)
!
      DATA X/92.0, 139.0, -6.0, 10.0, 81.0, -11.0, 45.0, -25.0, -4.0, &
           22.0, 2.0, 41.0, 13.0, 8.0, 33.0, 45.0, -33.0, -45.0, -12.0/
!                                Perform sign test
      CALL SIGNT (X, Q, P, NPOS, NTIE, PROB, NMISS)
!                                Print output
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99996) NPOS
      WRITE (NOUT,99997) NTIE
      WRITE (NOUT,99998) PROB
      WRITE (NOUT,99999) NMISS
!
99996 FORMAT (' Number of positive  differences = ', I2)
99997 FORMAT (' Number of ties                  = ', I2)
99998 FORMAT (' PROB                            = ', F6.3)
99999 FORMAT (' Number of missing values        = ', I2)
      END
```

Output

```
 Number of positive  differences = 12
 Number of ties                  =  0
 PROB                            =  0.923
 Number of missing values        =  0
```

SNRNK

Performs a Wilcoxon signed rank test.

Required Arguments

Y — Vector of length **NOBS** containing the data. (Input)

FUZZ — Constant used to determine when a value is 0.0 or when two values are tied. (Input)
When |**Y**(*i*)| or |**Y**(*i*) **-** **Y**(*j*)| is less than or equal to **FUZZ**, then the *i*-th observation is taken to be zero, or the *i*-th and *j*-th observations are said to be tied, respectively.

STAT — Vector of length 10 containing the computed statistics. (Output)
Statistics are computed in two ways. In method 1, the average rank of tied observations is used, and observations equal to zero are not counted. In method 2, ties are randomly broken, and observations equal to zero are randomly assigned to the positive or negative half line.

I	**STAT(I)**
1	The positive rank sum, *W*+, using method 1.
2	The absolute value of the negative rank sum, *W*-, using method 1.
3	The standardized (to an asymptotic variance of 1.0) minimum of (*W*+, *W*-) using method 1.
4	The asymptotic probability of not exceeding STAT(3) under the null hypothesis that the distribution is symmetric about 0.0.
5	The positive rank sum, *W*+, using method 2.
6	The absolute value of the negative rank sum, *W*-, using method 2.
7	The standardized (to an asymptotic variance of 1.0) minimum of (*W*+, *W*-) using method 2.
8	The asymptotic probability of not exceeding STAT(7) under the null hypothesis that the distribution is symmetric about 0.0.
9	The number of zero observations.
10	The total number of observations that are tied, and that are not within FUZZ of zero.

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**Y**,1).

NMISS — Number of missing values in **Y**. (Output)

FORTRAN 90 Interface

Generic: **CALL SNRNK** (**Y**, **FUZZ**, **STAT** [, ...])
Specific: The specific interface names are **S_SNRNK** and **D_SNRNK**.

FORTRAN 77 Interface

Single: **CALL SNRNK** (**NOBS**, **Y**, **FUZZ**, **STAT**, **NMISS**)
Double: The double precision name is **DSNRNK**.

Description

Routine **SNRNK** performs a Wilcoxon signed rank test of symmetry about zero. In one sample, this test can be viewed as a test that the population median is zero. In matched samples, a test that the medians of the two populations are equal can be computed by first computing difference scores. These difference scores would then be used as input to **SNRNK**. A general reference for the methods used is Conover (1980).

Routine **SNRNK** computes statistics for two methods for handling zero and tied observations. In the first method, observations within **FUZZ** of zero are not counted, and the average rank of tied observations is used. (Observations within **FUZZ** of each other are said to be tied.) In the second method, observations within **FUZZ** of zero are randomly assigned a positive or negative sign, and the ranks of tied observations are randomly permuted.

The *W*+ and *W*- statistics are computed as the sums of the ranks of the positive observations and the sum of the ranks of the negative observations, respectively. Asymptotic probabilities are computed using standard methods (see, e.g., Conover 1980, page 282).

The W+ and W- statistics may be used to test the following hypotheses about the median, M. In deciding whether to reject the null hypothesis, use the bracketed statistic if method 2 for handling ties is preferred. Possible null hypotheses and alternatives are given as follows:

- $H_0 : M \leq 0$ $H_1 : M > 0$
  Reject if **STAT**(1) [or **STAT**(5)] is too large.
- $H_0 : M \geq 0$ $H_1 : M < 0$
  Reject if **STAT**(2) [or **STAT**(6)] is too large.
- $H_0 : M = 0$ $H_1 : M \neq 0$
  Reject if **STAT**(3) [or **STAT**(7)] is too small. Alternatively, if an asymptotic test is desired, reject if 2 * **STAT**(4) [or 2 * **STAT**(8)] is less than the significance level.

- Tabled values of the test statistic can be found in the references. If possible, tabled values should be used. If the number of nonzero observations is too large, then the asymptotic probabilities computed by SNRNK can be used.

  The assumptions required for the hypothesis tests are as follows:

  c. The distribution of each X_i is symmetric.

  d. The X_i are mutually independent.

  e. All X_i's have the same median.

  f. An ordering of the observations exists (i.e., $X_1 > X_2$ and $X_2 > X_3$ implies that $X_1 > X_3$).

- If other assumptions are made, related hypotheses that are more (or less) restrictive can be tested.

Comments

1. Workspace may be explicitly provided, if desired, by use of S2RNK/DS2RNK. The reference is:

   ```
   CALL S2RNK (NOBS, Y, FUZZ, ISEED, STAT, NMISS, IR, YRANK)
   ```

   The additional arguments are as follows:

   IR — Work vector of length NOBS.

   YRANK — Work vector of length NOBS.

   If Y is not needed, Y and YRANK can share the same storage locations.

2. Informational errors

Type	Code	Description
3	4	NOBS is less than 50 and exact tables should be referenced for probabilities.
3	5	Each element of Y is within FUZZ of 0. STAT(1) through STAT(8) are set to NaN (not a number).

3. The signed rank statistic provides a test of the hypothesis that the population median is equal to zero. To test that the median is equal to some other value, say, 10.0, use the routine SADD (IMSL MATH/LIBRARY) to subtract 10.0 from each observation prior to calling SNRNK.

4. The signed rank test can be used to test that the medians of two matched random variables are equal. This is the nonparametric equivalent of the paired *t*-test. To use SNRNK to perform this test, use the routine SAXPY (IMSL MATH/LIBRARY) prior to calling SNRNK to compute the differences, Y(*i*) - X(*i*). Then, call SNRNK with these differences.

5. The routine **RNUN** (see Chapter 18, "Random Number Generation") is used to randomly break ties. The routine **RNSET** in *Chapter 18* can be used to initialize the seed of the random number generator. The routine **RNOPT** (also in *Chapter 18*) can be used to select the form of the generator.

Example

This example illustrates the application of the Wilcoxon signed rank test to a test on two matched samples (matched pairs). A test that the median difference is 10.0 (rather than 0.0) is performed by subtracting 10.0 from each of the differences prior to calling **SNRNK**. The routine **RNSET** (Chapter 18, "Random Number Generation") is used to set the seed. As can be seen from the output, the null hypothesis is rejected. The warning error will always be printed when the number of observations is 50 or less unless printing is turned off for warning errors. See routine **ERSET** in the Reference Material.

```
      USE WRRRN_INT
      USE RNSET_INT
      USE SNRNK_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      REAL       FUZZ
      PARAMETER  (FUZZ=0.0001, NOBS=7)
!
      INTEGER    I, NMISS, NOUT
      REAL       STAT(10), W(NOBS), X(NOBS), Y(NOBS)
!
      DATA W/223, 216, 211, 212, 209, 205, 201/
      DATA X/208, 205, 202, 207, 206, 204, 203/
!
      DO 10  I=1, NOBS
         Y(I) = X(I) - W(I) - 10.0
   10 CONTINUE
!                                Print Y prior to calling SNRNK
      CALL WRRRN ('Y', Y, 1, NOBS, 1, 0)
!                                Initialize the seed
      CALL RNSET (123457)
!
      CALL SNRNK (Y, FUZZ, STAT, NMISS=NMISS)
!                                Print output
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT(1), STAT(5), STAT(2), STAT(6), STAT(3), &
                         STAT(7), STAT(4), STAT(8), STAT(9), STAT(10), &
                         NMISS
!
99999 FORMAT (' Statistic                    Method 1     Method 2', &
            /, ' W+.....................', F9.0, 4X, F9.0, /, &
            ' W-.....................', F9.0, 4X, F9.0, /, &
            ' Standardized Minimum.....', F9.4, 4X, F9.4, /, &
            ' p-value.................', F9.4, 4X, F9.4, //, &
            ' Number of zeros.........', F9.0, /, ' Number of ', &
            'ties..........', F9.0, /, ' Number of missing........', &
            I5)
!
```

```
END
```

Output

```
                          Y
   1       2       3       4       5       6       7
-25.00  -21.00  -19.00  -15.00  -13.00  -11.00   -8.00

*** WARNING  ERROR 4 from SNRNK.  NOBS = 7.  The number of
***          observations, NOBS, is less than 50, and exact
***          tables should be referenced for probabilities.

Statistic                    Method 1     Method 2
W+.......................       0.           0.
W-.......................      28.          28.
Standardized Minimum.....  -2.3664      -2.3664
p-value..................   0.0090       0.0090

Number of zeros..........       0.
Number of ties...........       0.
Number of missing........    0
```

NCTRD

Performs the Noether test for cyclical trend.

Required Arguments

X — Vector of length NOBS containing the observations in chronological order. (Input)

FUZZ — Value to be used in determining when consecutive observations in X are tied. (Input)
If |X(*i* + 1) - X(*i*)| is less than or equal to FUZZ, then X(*i* + 1) and X(*i*) are said to be tied.

NSTAT — Vector of length 6 containing output statistics. (Output)

I	NSTAT(I)
1	The number of consecutive sequences of length three used to detect cyclical trend when tying middle elements are eliminated from the sequence, and the next consecutive observation is used.
2	The number of monotonic sequences of length three in the set defined by NSTAT(1).
3	The number of monotonic sequences where tied threesomes are counted as nonmonotonic.
4	The number of monotonic sequences where tied threesomes are counted as monotonic.
5	The number of middle observations eliminated because they were tied in forming the NSTAT(1) sequences.
6	The number of tied sequences found in forming the NSTAT(3) and NSTAT(4) sequences. A sequence is called a tied sequence if the middle element is tied with either of the two other elements.

P — Vector of length 3 containing the probabilities of NSTAT(2) or more, NSTAT(3) or more, or NSTAT(4) or more monotonic sequences. (Output)
If NSTAT(1) is less than 1, P(1) is set to NaN (not a number).

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than or equal to 3.
Default: NOBS = size (X,1).

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: CALL NCTRD (X, FUZZ, NSTAT, P [, ...])

Specific: The specific interface names are S_NCTRD and D_NCTRD.

FORTRAN 77 Interface

Single: CALL NCTRD (NOBS, X, FUZZ, NSTAT, P, NMISS)

Double: The double precision name is DNCTRD.

Description

Routine NCTRD performs the Noether test for cyclical trend (Noether 1956) for a sequence of measurements. In this test, the observations are first divided into sets of three consecutive observations. Each set is then inspected, and if the set is monotonically increasing or decreasing, the count variable is incremented.

The count variables, NSTAT(2), NSTAT(3), and NSTAT(4), differ in the manner in which ties are handled. A tie can occur in a set (of size three) only if the middle element is tied with either of the two ending elements. Tied ending elements are not considered. In NSTAT(2), tied middle observations are eliminated, and a new set of size 3 is obtained by using the next observation in the sample. In NSTAT(3), the original set of size three is used, and tied middle observations are counted as nonmonotonic. In NSTAT(4), tied middle observations are counted as monotonic.

The probabilities of occurrence of the counts are obtained from the binomial distribution with $p = 1/3$, where p is the probability that a random sample of size three from a continuous distribution is monotonic. The binomial sample size is, of course, the number of sequences of size three found (adjusted for ties).

Hypothesis test:

$H_0 : q = \Pr(X_i > X_{i-1} > X_{i-2}) + \Pr(X_i < X_{i-1} < X_{i-2}) \leq 1/3 \qquad H_1 : q > 1/3$

Reject if P(1) (or P(2) or P(3) depending on the method used for handling ties) is less than the significance level of the test.

Assumption: The observations are independent and are from a continuous distribution.

Comments

1. Informational errors

Type	Code	Description
3	3	NSTAT(1), which is used to determine NSTAT(3) and NSTAT(4), is less than 8. The asymptotic probabilities will not be exact.
3	4	At least one tie was detected in X.

2. If **NOBS** is greater than or equal to 3 but **NSTAT**(1) is less than one, **P**(1) will be set to NaN. The remaining statistics and associated probabilities will be determined and returned as described.

Example

A test for cyclical trend in a sequence of 1000 randomly generated observations is performed. Because of the sample used, there are no ties and all three test statistics yield the same result.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE IMSL_LIBRARIES

      IMPLICIT   NONE

      INTEGER    NOBS
      REAL       FUZZ
      PARAMETER  (FUZZ=0.0, NOBS=1000)
!
      INTEGER    ISEED, NSTAT(6)
      REAL       P(3), X(NOBS)
!
      DATA ISEED/123457/
!
      CALL RNSET (ISEED)
      CALL RNUN (X)
!                                  Noether test
      CALL NCTRD (X, FUZZ, NSTAT, P)
!                                  Print results
      CALL WRIRN ('NSTAT', NSTAT, 1, 6, 1, 0)
      CALL WRRRN ('P', P, 1, 3, 1, 0)
!
      END
```

Output

```
                NSTAT
 1     2     3     4     5     6
333   107   107   107     0     0

             P
 1         2         3
0.6979    0.6979    0.6979
```

SDPLC

Performs the Cox and Stuart sign test for trends in dispersion and location.

Required Arguments

X — Vector of length **NOBS** containing the observations in chronological order. (Input)

K — Number of consecutive **X** elements to be used to measure dispersion. (Input)
Not required if **IOPT** is different from zero.

IDS — Dispersion measure option. (Input)
If **IDS** is zero, the range is used as a measure of dispersion. Otherwise, the centered sum of squares is used. Not required if **IOPT** is different from zero.

FUZZ — Value used to determine when elements in **X** are tied. (Input)
If |**X**(*i*) **-** **X**(*j*)| is less than or equal to **FUZZ**, **X**(*i*) and **X**(*j*) are said to be tied. **FUZZ** must be nonnegative.

NSTAT — Vector of length 8. (Output)
The first 4 elements of **NSTAT** are the output statistics when the observations are divided into two groups. The last 4 elements are the output statistics when the observations are divided into three groups.

I	NSTAT(I)
1	Number of negative differences (two groups)
2	Number of positive differences (two groups)
3	Number of zero differences (two groups)
4	Number of differences used to calculate PSTAT(1) through PSTAT(4) (two groups).
5	Number of negative differences (three groups)
6	Number of positive differences (three groups)
7	Number of zero differences (three groups)
8	Number of differences used to calculate PSTAT(5) through PSTAT(8) (three groups).

PSTAT — Vector of length 8 containing probabilities. (Output)
The first four elements of **PSTAT** are computed from two groups of observations.

I	PSTAT(I)
1	Probability of NSTAT(1) + NSTAT(3) or more negative signs (ties are considered negative).
2	Probability of obtaining NSTAT(2) or more positive signs (ties are considered negative).
3	Probability of NSTAT(1) + NSTAT(3) or more negative signs (ties are considered positive).
4	Probability of obtaining NSTAT(2) or more positive signs (ties are considered positive).

The last four elements of PSTAT are computed from three groups of observations.

I	PSTAT(I)
5	Probability of NSTAT(1) + NSTAT(3) or more negative signs (ties are considered negative).
6	Probability of obtaining NSTAT(2) or more positive signs (ties are considered negative).
7	Probability of NSTAT(1) + NSTAT(3) or more negative signs (ties are considered positive).
8	Probability of obtaining NSTAT(2) or more positive signs (ties are considered positive).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

IOPT — Statistic option parameter. (Input)
If IOPT = 0, the Cox and Stuart tests for trends in dispersion are computed. Otherwise, the Cox and Stuart tests for trends in location are computed.
Default: IOPT = 0.

NMISS — Number of missing values in X. (Output)

FORTRAN 90 Interface

Generic: CALL SDPLC (X, K, IDS, FUZZ, NSTAT, PSTAT [, ...])
Specific: The specific interface names are S_SDPLC and D_SDPLC.

FORTRAN 77 Interface

Single: CALL SDPLC (NOBS, X, IOPT, K, IDS, FUZZ, NSTAT, PSTAT, NMISS)
Double: The double precision name is DSDPLC.

Description

Routine SDPLC tests for trends in dispersion or location in a sequence of random variables depending upon the value of the input variable IOPT. A derivative of the sign test is used (see Cox and Stuart 1955).

Location Test

For the location test (IOPT = 1) with two groups, the observations are first divided into two groups with the middle observation thrown out if there are an odd number of observations. Each observation in group one is then compared with the observation in group two that has the same lexicographical order. A count is made of the number of times a group-one observation is less than (NSTAT(1)), greater than (NSTAT(2)), or equal to (NSTAT(3)), its counterpart in group two. Two observations are counted as equal if they are within FUZZ of one another.

In the three-group test, the observations are divided into three groups, with the center group losing observations if the division is not exact. The first and third groups are then compared as in the two-group case, and the counts are stored in NSTAT(5) through NSTAT(7).

Probabilities in PSTAT are computed using the binomial distribution with sample size equal to the number of observations in the first group (NSTAT(4) or NSTAT(8)), and binomial probability $p = 0.5$.

Dispersion Test

The dispersion tests proceed exactly as with the tests for location, but using one of two derived dispersion measures. The input value K is used to define NOBS/K groups of consecutive observations starting with observation 1. The first K observations define the first group, the next K observations define the second group, etc., with the last observations omitted if NOBS is not evenly divisible by K. A dispersion score is then computed for each group as either the range (IDS = 0), or a multiple of the variance (IDS ≠ 0) of the observations in the group. The dispersion scores form a derived sample. The tests proceed on the derived sample as above.

Ties

Ties are defined as occurring when a group one observation is within **FUZZ** of its last group counterpart. Ties imply that the probability distribution of **X** is not strictly continuous, which means that $Pr(X_1 > X_2) \neq 0.5$ under the null hypothesis of no trend (and the assumption of independent identically distributed observations). When ties are present, the computed binomial probabilities are not exact, and the hypothesis tests will be conservative.

Hypothesis tests

In the following, *i* indexes an observation from group 1, while *j* indexes the corresponding observation in group 2 (two groups) or group 3 (three groups).

- $H_0 : Pr(X_i > X_j) = Pr(X_i < X_j) = 0.5$
  $H_1 : Pr(X_i > X_j) < Pr(X_i < X_j)$
  Hypothesis of upward trend. Reject if **PSTAT**(3) (or **PSTAT**(7)) is less than the significance level.
- $H_0 : Pr(X_i > X_j) = Pr(X_i < X_j) = 0.5$
  $H_1 : Pr(X_i > X_j) > Pr(X_i < X_j)$
  Hypothesis of downward trend. Reject if **PSTAT**(2) (or **PSTAT**(6)) is less than the significance level.
- $H_0 : Pr(X_i > X_j) = Pr(X_i < X_j) = 0.5$
  $H_1 : Pr(X_i > X_j) \neq Pr(X_i < X_j)$
  Two tailed test. Reject if 2 max(**PSTAT** (2), **PSTAT**(3)) (or 2 max(**PSTAT** (6), **PSTAT**(7)) is less than the significance level.

Assumptions

1. The observations are a random sample; i.e., the observations are independently and identically distributed.
2. The distribution is continuous.

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2PLC/DS2PLC**. The reference is:

   ```
   CALL S2PLC (NOBS, X, IOPT, K, IDS, FUZZ, NSTAT, PSTAT, NMISS, XWK)
   ```

   The additional argument is:

   XWK — Work vector of length **NOBS**.

   If **X** is not needed, **X** and **XWK** can share the same storage location.

2. Informational errors

Type	Code	Description
4	4	NSTAT(4) is too small to continue with a dispersion test.
3	5	At least one tie is detected in X.

Example

This example illustrates both the location and dispersion tests. The data, which are taken from Bradley (1968), page 176, give the closing price of AT&T on the New York stock exchange for 36 days in 1965. Tests for trends in location (`IOPT` = 1), and for trends in dispersion (`IOPT` = 0) are performed. Trends in location are found.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    IDS, K, NOBS, IS
      REAL       FUZZ
      PARAMETER  (FUZZ=0.001, IDS=0, K=2, NOBS=36)
!
      INTEGER    IOPT, NSTAT(8)
      REAL       PSTAT(8), X(NOBS)
!
      DATA X/9.5, 9.875, 9.25, 9.5, 9.375, 9.0, 8.75, 8.625, 8.0, &
          8.25, 8.25, 8.375, 8.125, 7.875, 7.5, 7.875, 7.875, 7.75, &
          7.75, 7.75, 8.0, 7.5, 7.5, 7.125, 7.25, 7.25, 7.125, 6.75, &
          6.5, 7.0, 7.0, 6.75, 6.625, 6.625, 7.125, 7.75/
!                                Tests for trends in location
      IOPT = 1
      IS = 1
      CALL SDPLC (X, K, IDS, FUZZ, NSTAT, PSTAT, IOPT=IOPT)
!                                Print results
      CALL WROPT (-6, IS, 1)
      CALL WRIRN ('NSTAT', NSTAT, 1, 8, 1, 0)
      CALL WRRRN ('PSTAT', PSTAT, 1, 8, 1, 0)
!                                Tests for trends in dispersion
      IOPT = 0
      CALL SDPLC (X, K, IDS, FUZZ, NSTAT, PSTAT)
!                                Print results
      CALL WRIRN ('NSTAT', NSTAT, 1, 8, 1, 0)
      CALL WRRRN ('PSTAT', PSTAT, 1, 8, 1, 0)
!
      END
```

Output

```
*** WARNING  ERROR 5 from SDPLC.  At least one tie is detected in X.

            NSTAT
1    2    3    4    5    6    7    8
0   17    1   18    0   12    0   12
```

```
              PSTAT
        1             2             3             4             5
  1.00000       0.00007       1.00000       0.00000       1.00000

        6             7             8
  0.00024       1.00000       0.00024

  *** WARNING  ERROR 5 from SDPLC.  At least one tie is detected in X.

              NSTAT
  1   2   3   4   5   6   7   8
  4   3   2   9   4   2   0   6

                      PSTAT
        1             2             3             4             5
  0.253906      0.910156      0.746094      0.500000      0.343750
        6             7             8
  0.890625      0.343750      0.890625
```

NTIES

Computes tie statistics for a sample of observations.

Required Arguments

X — Vector of length **NOBS** containing the observations. (Input)
X must be ordered monotonically increasing with all missing values removed.

FUZZ — Value used to determine ties. (Input)
Observations *i* and *j* are tied if the successive differences **X**(*k* + 1) **-** **X**(*k*) between observations *i* and *j*, inclusive, are all less than **FUZZ**. **FUZZ** must be nonnegative.

TIES — Vector of length 4 containing the tie statistics. (Output)
The tie statistics are returned in **TIES** and are computed as follows:

$$TIES(1) = \sum_{j=1}^{\tau} \left[t_j \left(t_j - 1 \right) \right] / 2$$

$$TIES(2) = \sum_{j=1}^{\tau} \left[t_j \left(t_j - 1 \right) \left(t_j + 1 \right) \right] / 12$$

$$TIES(3) = \sum_{j=1}^{\tau} t_j \left(t_j - 1 \right) \left(2t_j + 5 \right)$$

$$TIES(4) = \sum_{j=1}^{\tau} t_j \left(t_j - 1 \right) \left(t_j - 2 \right)$$

where t_j is the number of ties in the *j*-th group (rank) of ties, and τ is the number of tie groups in the sample.

Optional Arguments

NOBS — The number of observations. (Input)
Default: **NOBS** = size (**X**,1).

FORTRAN 90 Interface

Generic: **CALL NTIES** (X, **FUZZ**, **TIES** [, ...])
Specific: The specific interface names are **S_NTIES** and **D_NTIES**.

FORTRAN 77 Interface

Single: CALL NTIES (NOBS, X, FUZZ, TIES)

Double: The double precision name is DNTIES.

Description

Routine NTIES computes tie statistics for a monotonically increasing sample of observations. "Tie statistics" are statistics that may be used to correct a continuous distribution theory nonparametric test for tied observations in the data. Observations *i* and *j* are tied if the successive differences X(*k* + 1) - X(*k*), inclusive, are all less than FUZZ. Note that if each of the monotonically increasing observations is equal to its predecessor plus a constant, if that constant is less than FUZZ, then all observations are contained in one tie group. For example, if FUZZ = 0.11, then the following observations are all in one tie group.

0.0, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00

Example

We want to compute tie statistics for a sample of length 7.

```
      USE NTIES_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      REAL       FUZZ
      PARAMETER  (FUZZ=0.001, NOBS=7)
!
      REAL       TIES(4), X(NOBS)
!
      DATA X/1.0, 1.0001, 1.0002, 2.0, 3.0, 3.0, 4.0/
!                                Compute tie statistics
      CALL NTIES (X, FUZZ, TIES)
!                                Print results
      CALL WRRRN ('TIES', TIES, 1, 4, 1, 0)
!
      END
```

Output

```
             TIES
     1       2       3       4
  4.00    2.50   84.00    6.00
```

RNKSM

Performs the Wilcoxon rank sum test.

Required Arguments

X — Vector of length NOBSX containing the first sample. (Input)

Y — Vector of length NOBSY containing the second sample. (Input)

FUZZ — Constant used to determine ties in X and Y. (Input)
If $|z_i - z_j| \leq$ FUZZ, then z_i and z_j are said to be tied, where z_i is the i-th element of X or Y. FUZZ must be nonnegative.

STAT — Vector of length 10 containing the output statistics. (Output)

I	STAT(I)
1	Wilcoxon *W* statistic (the sum of the ranks of the X observations) adjusted for ties in such a manner that *W* is as small as possible.
2	2 * *E*(*W*) - *W*, where *E*(*W*) is the expected value of *W*.
3	Probability of obtaining a statistic less than or equal to the minimum of (*W*, 2*E*(*W*) - *W*).
4	*W* statistic adjusted for ties in such a manner that *W* is as large as is possible.
5	STAT(2); but adjusted for ties as in 4.
6	STAT(3); but adjusted for ties as in 4.
7	W statistic with average ranks used in place of tied ranks.
8	Estimated standard error of STAT(7) under the null hypothesis of no difference.
9	Standard normal score associated with STAT(7).
10	Two-sided *p*-value associated with STAT(9).

Optional Arguments

NOBSX — Number of observations in X. (Input)
Default: NOBSX = size (X,1).

NOBSY — Number of observations in Y. (Input)
Default: NOBSY = size (Y,1).

NMISSX — Number of missing (NaN, not a number) observations in **X**. (Output)

NMISSY — Number of missing (NaN, not a number) observations in **Y**. (Output)

FORTRAN 90 Interface

Generic: CALL RNKSM (X, Y, FUZZ, STAT [, ...])
Specific: The specific interface names are S_RNKSM and D_RNKSM.

FORTRAN 77 Interface

Single: CALL RNKSM (NOBSX, X, NOBSY, Y, FUZZ, STAT, NMISSX, NMISSY)
Double: The double precision name is DRNKSM.

Description

Routine **RNKSM** performs the Wilcoxon rank sum test for identical population distribution functions. The Wilcoxon test is a linear transformation of the Mann-Whitney *U* test. If the difference between the two populations can be attributed solely to a difference in location, then the Wilcoxon test becomes a test of equality of the population means (or medians) and is the nonparametric equivalent of the two-sample *t*-test.

Routine **RNKSM** obtains ranks in the combined sample after first eliminating missing values from the data. The rank sum statistic is then computed as the sum of the ranks in the **X** sample. Three methods for handling ties are used. (A tie is counted when two observations are within **FUZZ** of each other.) The first method uses the largest possible rank for tied observations in the smallest sample, while the second method uses the smallest possible rank for these observations. Thus, the range of possible rank sums is obtained. The third, method for handling tied observations between samples uses the average rank of the tied observations.

Asymptotic standard normal scores are computed for the *W* score (based upon a variance that has been adjusted for ties) when average ranks are used (see Conover 1980, page 217), and the probability associated with the two-sided alternative is computed.

Hypothesis Tests

In each test following, the first line gives the hypothesis (and its alternative) under the assumptions 1 to 3 below, while the second line gives the hypothesis when assumption 4 is also true. The rejection region is the same for both hypotheses and is given in terms of method 3 for handling ties. Another output statistic should be used (**STAT**(1) or **STAT**(4)) if another method for handling ties is desired.

- H_0 : Pr(**X** < **Y**) = 0.5 H_1 : Pr(**X** < **Y**) ≠ 0.5
  H_0 : E(**X**) = E(**Y**) H_1 : E(**X**) ≠ E(**Y**)
  Reject if **STAT**(10) is less than the significance level of the test. Alternatively, reject H_0 if **STAT**(7) is too large or too small.
- H_0 : Pr(**X** < **Y**) ≤ 0.5 H_1 : Pr(**X** < **Y**) > 0.5
  H_0 : E(**X**) ≥ E(**Y**) H_1 : E(**X**) < E(**Y**)
  Reject if **STAT**(7) is too small.
- H_0 : Pr(**X** < **Y**) ≥ 0.5 H_1 : Pr(**X** < **Y**) < 0.5
  H_0 : E(**X**) ≤ E(**Y**) H_1 : E(**X**) > E(**Y**)
  Reject if **STAT**(7) is too large.

Assumptions

1. **X** and **Y** are a random sample from their respective populations.
2. All observations are mutually independent.
3. The measurement scale is at least ordinal (i.e., an ordering less than, greater than, or equal to exists among the observations).
4. If $F(X)$ and $G(Y)$ are the distribution functions of X and Y, respectively, then $G(Y) = F(X + c)$ for some constant c (i.e., the distribution of Y is at worst a translation of the distribution of X).

Tables of critical values of the W statistic are given in the references for small samples.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2KSM/DR2KSM**. The reference is:

   ```
   CALL R2KSM (NOBSX, X, NOBSY, Y, FUZZ, STAT, NMISSX, NMISSY, IWK, YWK)
   ```

   The additional arguments are as follows:

   IWK — Integer work vector of length **NOBSX** + **NOBSY**

   YWK — Work vector of length **NOBSX** + **NOBSY**.

2. Informational errors

Type	Code	Description
3	4	Both NOBSX and NOBSY are less than 25. Tabled critical values for *W* should be used.
3	5	Tied observations occurred between the samples.
4	6	Each element of X and/or Y is a missing (NaN, not a number) value.

3. The Mann-Whitney *U* statistic is given in terms of *W* as *U* = *W* - *K* * (*K* + 1)/2, where *K* = **NOBSX**, and *W* = **STAT**(1) (or **STAT**(4)). Tables of critical values for *W* are available in the references given in the manual document.

4. For greatest efficiency in computing *W*, the **X** sample should be the smallest sample.

Example

The following example is taken from Conover (1980, page 224). It involves the mixing time of 2 mixing machines using a total of 10 batches of a certain kind of batter, 5 batches for each machine. The null hypothesis is not rejected at the 5 percent level of significance. The warning error is always printed when one or more ties are detected unless printing for warning errors is turned off. See routine **ERSET** in the Reference Material.

```
      USE RNKSM_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBSX, NOBSY
      REAL       FUZZ
      PARAMETER  (FUZZ=0.001, NOBSX=5, NOBSY=5)
!
      INTEGER    I, NMISSX, NMISSY, NOUT
      REAL       STAT(10), X(NOBSX), Y(NOBSY)
!
      DATA X/7.3, 6.9, 7.2, 7.8, 7.2/
      DATA Y/7.4, 6.8, 6.9, 6.7, 7.1/
!
      CALL RNKSM (X, Y, FUZZ, STAT, NMISSX=NMISSX, NMISSY=NMISSY)
!                                Print the results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,10), REAL(NMISSX), REAL(NMISSY)
!
99999 FORMAT (' Wilcoxon W statistic ......................', F5.1, &
            /, ' 2*WBAR - W ..............................', &
            F5.1, /, ' p-value ...................................' &
            , F7.3, /, ' Adjusted Wilcoxon statistic ', '............' &
            , '.....', F5.1, /, ' Adjusted 2*WBAR - W ', '...........', &
            '..', '............', F5.1, /, ' Adjusted p-value ', &
            '.........................', F7.3, /, ' W statistic ', &
            'for averaged ranks ..............', F5.1, /, ' Standard ' &
            , 'error of W (averaged ranks) ........', F7.3, /, &
            ' Standard normal score of W (averaged ranks) .', F7.3, &
            /, ' Two-sided p-value of W (averaged ranks) .....', &
            F7.3, //, ' Number of missing for X .....................' &
```

```
              , F5.1, /, ' Number of missing for Y ', '.............' &
              , '.....', F5.1)
!
      END
```

Output

```
 *** WARNING  ERROR 5 from RNKSM.  At least one tie is detected between the
 ***          samples.
 Wilcoxon W statistic ..................... 34.0
 2*WBAR - W ............................... 21.0
 p-value ..................................  0.110
 Adjusted Wilcoxon statistic .............. 35.0
 Adjusted 2*WBAR - W ...................... 20.0
 Adjusted p-value ..........................  0.075
 W statistic for averaged ranks ............ 34.5
 Standard error of W (averaged ranks) ......  4.758
 Standard normal score of W (averaged ranks) .  1.471
 Two-sided p-value of W (averaged ranks) .....  0.141

 Number of missing for X ..................  0.0
 Number of missing for Y ..................  0.0
```

INCLD

Performs an includance test.

Required Arguments

X — Vector of length NOBSX containing the data for the first sample. (Input)

Y — Vector of length NOBSY containing the data for the second sample. (Input)

ILX — Index of the element in the ordered first sample to be used as the low endpoint of the range considered. (Input)
ILX must be greater than zero and less than IHX.

IHX — Index of the element in the ordered first sample to be used as the high endpoint of the range considered. (Input)
IHX must be greater than ILX and less than or equal to NOBSX.

FUZZ — Value used to determine ties. (Input)
If a second sample element is within FUZZ of the ILX or IHX order statistics in the first sample, a tie will be counted.

STAT — Vector of length 4 containing the statistics. (Output)
In the description below, (X(ILX), X(IHX)) is the interval from the ILX ordered first sample value to the IHX ordered first sample value (i.e., from the ILX to the IHX order statistics in the first sample).

I	STAT(I)
1	Number of ties detected.
2	Number of untied elements in the second sample that are outside the interval (X(ILX), X(IHX)).
3	Probability of STAT(2) or more second sample elements lying outside (X(ILX), X(IHX)).
4	Probability of STAT(1) + STAT(2) or more elements in the second sample lying outside (X(ILX), X(IHX)).

Optional Arguments

NOBSX — Number of observations in the first sample. (Input)
Default: NOBSX = size (X,1).

NOBSY — Number of observations in the second sample. (Input)
Default: NOBSY = size (Y,1).

NMISSX — Number of missing (NaN, not a number) values in X. (Output)

NMISSY — Number of missing (NaN, not a number) values in Y. (Output)

FORTRAN 90 Interface

Generic: CALL INCLD (X, Y, ILX, IHX, FUZZ, STAT [, ...])
Specific: The specific interface names are S_INCLD and D_INCLD.

FORTRAN 77 Interface

Single: CALL INCLD (NOBSX, X, NOBSY, Y, ILX, IHX, FUZZ, STAT, NMISSX, NMISSY)
Double: The double precision name is DINCLD.

Description

Routine INCLD tests that an equal proportion of two populations lies between the ILX and IHX order statistics of the first sample, and that the densities are equal at the two points. Let X_{il} and X_{ih} denote the two order statistics in the first sample, where l = ILX, and h = IHX. Then, the probability of exactly i observations in the second sample being outside of the interval (X_{il} , X_{ih}) is hypergeometric and is given by

$$\Pr_i = \frac{\binom{M-b+(ih-il+1)}{M-i}\binom{1}{0}\binom{N-(ih-il+1)}{b}}{\binom{N+M}{M}}$$

where M is the sample size in the first sample (NOBSX - NMISSX), N is the sample size in the second sample (NOBSY - NMISSY), and

$$\binom{n}{x}$$

denotes a binomial coefficient. The probability of b or fewer observations in the second sample being outside the interval is given by

$$\Pr = \sum_{i=0}^{b} \Pr_i$$

Use of this test requires that the population samples sizes, **ILX** and **IHX**, be set prior to sampling or viewing the data. Ties do not present any special problems except when they occur at the interval endpoints X_{il} and X_{ih}. When this occurs for the first sample, no action is taken, but an informative warning message is issued. When a second sample observation is within **FUZZ** of an endpoint, then a tie is counted in **STAT**(1), and once more, a warning message is issued. In this case, **STAT**(3) and **STAT**(4) can be considered as bounds for the actual probability.

Comments

1. Workspace may be explicitly provided, if desired, by use of **I2CLD/DI2CLD**. The reference is:

   ```
   CALL I2CLD (NOBSX, X, NOBSY, Y, ILX, IHX, FUZZ, STAT, NMISSX, NMISSY, WK)
   ```

   The additional argument is:

   WK — Work vector of length **NOBSX**. If **X** is not needed, **X** and **WK** can share the same storage locations.

2. If **ILX** = 1 and **IHX** = **NOBSX**, **INCLD** tests the hypothesis that the second population lies in equal proportion to the first population, between the endpoints of the first sample.

3. If **ILX** = (**NOBSX** + 1)/4 and **IHX** = 3 * (**NOBSX** + 1)/4, the first and the third quartile estimates of the first population are being considered. The null hypothesis may be that the first and second samples are drawn from the same population.

Example

The following example, which is an adaptation of a problem in Bradley (1968, page 234) illustrates the use of **INCLD** to test that equal proportions of two populations lie between the endpoints of the first sample.

```
      USE INCLD_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    IHX, ILX, NOBSX, NOBSY
      REAL       FUZZ
      PARAMETER  (FUZZ=0.0001, IHX=12, ILX=1, NOBSX=12, NOBSY=15)
!
      REAL       STAT(4), X(NOBSX), Y(NOBSY)
      CHARACTER  CLABEL(5)*30, RLABEL(1)*4
!
      DATA X/1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12/
      DATA Y/0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2/
      DATA RLABEL/'NONE'/
      DATA CLABEL/' ', '%/Number of ties', '%/Number outside', &
```

```
            'p-value%/untied', 'p-value%/both'/
!                                 Perform includance test
      CALL INCLD (X, Y, ILX, IHX, FUZZ, STAT)
!                                 Print results
      CALL WRRRL ('STAT', STAT, RLABEL, CLABEL, 1, 4, 1)
!
      END
```

Output

```
                            STAT
                                          p-value     p-value
Number of ties  Number outside             untied        both
         0.000           7.000              0.038       0.038
```

KRSKL

Performs a Kruskal-Wallis test for identical population medians.

Required Arguments

NI — Vector of length NGROUP containing the number of responses for each of the NGROUP groups. (Input)

Y — Vector of length NI(1) + … + NI(NGROUP) that contains the responses for each of the NGROUP groups. (Input)
Y must be sorted by group, with the NI(1) observations in group 1 coming first, the NI(2) observations in group two coming second, and so on.

FUZZ — Constant used to determine ties in Y. (Input)
If (after sorting) |Y(*i*) - Y(*i* + 1)| is less than or equal to FUZZ, then a tie is counted. FUZZ must be nonnegative.

STAT — Vector of length 4 containing the Kruskal-Wallis statistics. (Output)

I	STAT(I)
1	Kruskal-Wallis *H* statistic.
2	Asymptotic probability of a larger *H* under the null hypothesis of identical population medians.
3	*H* corrected for ties.
4	Asymptotic probability of a larger *H* (corrected for ties) under the null hypothesis of identical populations.

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (NI,1).

FORTRAN 90 Interface

Generic: CALL KRSKL (NI, Y, FUZZ, STAT [, …])
Specific: The specific interface names are S_KRSKL and D_KRSKL.

FORTRAN 77 Interface

Single: CALL KRSKL (NGROUP, NI, Y, FUZZ, STAT)

Double: The double precision name is DKRSKL.

Description

The routine KRSKL generalizes the Wilcoxon two-sample test computed by routine RNKSM to more than two populations. It computes a test statistic for testing that the population distribution functions in each of *K* populations are identical. Under appropriate assumptions, this is a nonparametric analogue of the one-way analysis of variance. Since more than two samples are involved, the alternative is taken as the analogue of the usual analysis of variance alternative, namely that the populations are not identical.

The calculations proceed as follows: All observations are ranked regardless of the population to which they belong. Average ranks are used for tied observations (observations within FUZZ of each other). Missing observations (observations equal to NaN, not a number) are not included in the ranking. Let R_i denote the sum of the ranks in the *i*-th population. The test statistic *H* is defined as:

$$H = \frac{1}{s^2}\sum_{i=1}^{K}\left(\frac{R_i^2}{n_i} - \frac{N(N+1)^2}{4}\right)$$

where *N* is the total of the sample sizes, n_i is the number of observations in the *i*-th sample, and S^2 is computed as the (bias corrected) sample variance of the R_i.

The null hypothesis is rejected when STAT(4) (or STAT(2)) is less than the significance level of the test. If the null hypothesis is rejected, then the procedures given in Conover (1980, page 231) may be used for multiple comparisons. The routine KRSKL computes asymptotic probabilities using the chi-squared distribution when the number of groups is 6 or greater, and a Beta approximation (see Wallace 1959) when the number of groups is 5 or less. Tables yielding exact probabilities in small samples may be obtained from Owen (1962).

Comments

1. Workspace may be explicitly provided, if desired, by use of K2SKL/DK2SKL. The reference is:

   ```
   CALL K2SKL (NGROUP, NI, Y, FUZZ, STAT, IWK, WK, YRNK)
   ```

   The additional arguments are as follows:

   IWK — Integer work vector of length *m*.

   WK — Work vector of length *m*.

   YRNK — Work vector of length *m*.

2. Informational errors

Type	Code	Description
3	4	At least one tie was detected in Y.
3	5	All elements of Y are tied. STAT is set to -1.0.
3	6	The chi-squared degrees of freedom are less than 5, so the Beta approximation is used.

Example

The following example is taken from Conover (1980, page 231). The data represents the yields per acre of four different methods for raising corn. Since $H = 25.5$, the four methods are clearly different. The warning error is always printed when the Beta approximation is used, unless printing for warning errors is turned off. See IMSL routine **ERSET** in the Reference Material.

```
      USE KRSKL_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NGROUP
      REAL       FUZZ
      PARAMETER  (FUZZ=0.001, NGROUP=4)
!
      INTEGER    NI(NGROUP), NOUT
      REAL       STAT(4), Y(34)
!
      DATA NI/9, 10, 7, 8/
      DATA Y/83, 91, 94, 89, 89, 96, 91, 92, 90, 91, 90, 81, 83, 84, &
          83, 88, 91, 89, 84, 101, 100, 91, 93, 96, 95, 94, 78, 82, &
          81, 77, 79, 81, 80, 81/
!                                Perform Kruskal-Wallis test
      CALL KRSKL (NI, Y, FUZZ, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
!
99999 FORMAT (' H (no ties)    = ', F8.1, /, ' Prob (no ties) = ', &
            F11.4, /, ' H (ties)       = ', F8.1, /, ' Prob (ties)   ' &
            , ' = ', F11.4)
!
      END
```

Output

```
*** WARNING  ERROR 6 from KRSKL.  The chi-squared degrees of freedom are
***          less than 5, so the Beta approximation is used.
H (no ties)    =     25.5
Prob (no ties) =      0.0000
H (ties)       =     25.6
Prob (ties)    =      0.0000
```

BHAKV

Performs a Bhapkar *V* test.

Required Arguments

NI — Vector of length NGROUP containing the number of responses for each of the NGROUP groups. (Input)

Y — Vector of length NI(1) + NI(2) + … + NI(NGROUP) containing the responses for each of the NGROUP groups. (Input)
Y must be sorted by group with the NI(1) observations for group 1 coming first.

V — Bhapkar *V* statistic. (Output)

PROB — Asymptotic probability of exceeding V under the null hypothesis that the populations are equal. (Output)
Asymptotically, *V* follows a chi-squared distribution with NGROUP - 1 degrees of freedom.

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (NI,1).

FORTRAN 90 Interface

Generic: CALL BHAKV (NI, Y, V, PROB [, …])

Specific: The specific interface names are S_BHAKV and D_BHAKV.

FORTRAN 77 Interface

Single: CALL BHAKV (NGROUP, NI, Y, V, PROB)

Double: The double precision name is DBHAKV.

Description

Routine **BHAKV** tests the hypothesis that several samples are from the same population using the Bhapkar *V* statistic. Let the number of samples be denoted by *K* = **NGROUP**. To compute the Bhapkar *V* statistic, one first computes, for each group *i*, the statistic t_i = the number of *K*-tuples that can be formed with one observation from each sample such that the element from population *i* is the smallest. The sample variance of the ratio of t_i to the total number of such *k*-tuples is then computed. The Bhapkar *V* statistic is then a constant *c* multiplied by this variance, where $c = n(2m - 1)$, m = **NGROUP**, and *n* is the sum of the sample sizes (after missing values are eliminated).

Comments

Workspace may be explicitly provided, if desired, by use of **B2AKV**/**DB2AKV**. The reference is

```
CALL B2AKV (NGROUP, NI, Y, V, PROB, IWK, WK, YWK)
```

The additional arguments are as follows:

IWK — Integer work vector of length **NI**(1) + … + **NI**(**NGROUP**) + **NGROUP**

WK — Work vector of length **NGROUP**

YWK — Work vector of length **NI**(1) + … + **NI**(**NGROUP**). If **Y** is not needed, **Y** and **YWK** can share the same storage locations.

Example

We want to test the null hypothesis that three samples of size 3, 2, and 4, respectively, are from the same population using the Bhapkar *V* statistic.

```
      USE BHAKV_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NGROUP
      PARAMETER  (NGROUP=3)
!
      INTEGER    NI(NGROUP), NOUT
      REAL       PROB, V, Y(9)
!
      DATA NI/3, 2, 4/
      DATA Y/1, 3, 2, -1, 5, 4, 7, 2, 9/
!                                Perform Bhapkar V test
      CALL BHAKV (NI, Y, V, PROB)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998) V
      WRITE (NOUT,99999) PROB
!
99998 FORMAT (' V    = ', F12.5)
```

```
    99999 FORMAT (' Prob = ', F12.5)
    !
          END
```

Output

```
    V    =      1.89429
    Prob =      0.38785
```

FRDMN

Performs Friedman's test for a randomized complete block design.

Required Arguments

NB — Number of blocks. (Input)

NT — Number of treatments. (Input)

Y — Vector of length NB * NT containing the observations. (Input)
The first NT positions of Y contain the observations on treatments 1, 2, …, NT in the first block. The second NT positions contain the observations in the second block, etc., and so on.

FUZZ — Constant used to determine ties. (Input)
In the ordered observations, if |Y(*i*) - Y(*i* + 1)| is less than or equal to FUZZ, then Y(*i*) and Y(*i* + 1) are said to be tied.

ALPHA — Critical level for multiple comparisons. (Input)
ALPHA should be between 0 and 1 exclusive.

STAT — Vector of length 6 containing the Friedman statistics. (Output)
Probabilities reported are computed under the appropriate null hypothesis.

I	STAT(I)
1	Friedman two-sided test statistic.
2	Approximate *F* value for STAT(1).
3	Page test statistic for testing the ordered alternative that the median of treatment *i* is less than or equal to the median of treatment *i* + 1, with strict inequality holding for some *i*.
4	Asymptotic *p*-value for STAT(1). Chi-squared approximation.
5	Asymptotic *p*-value for STAT(2). *F* approximation.
6	Asymptotic *p*-value for STAT(3). Normal approximation.

SMRNK — Vector of length NT containing the sum of the ranks of each treatment. (Output)

D — Minimum absolute difference in two elements of SMRNK to infer at the alpha level of significance that the medians of the corresponding treatments are different. (Output)

FORTRAN 90 Interface

Generic: CALL FRDMN (NB, NT, Y, FUZZ, ALPHA, STAT, SMRNK, D)

Specific: The specific interface names are S_FRDMN and D_FRDMN.

FORTRAN 77 Interface

Single: CALL FRDMN (NB, NT, Y, FUZZ, ALPHA, STAT, SMRNK, D)

Double: The double precision name is DFRDMN.

Description

Routine FRDMN may be used to test the hypothesis of equality of treatment effects within each block in a randomized block design. No missing values are allowed. Ties are handled by using the average ranks. The test statistic is the nonparametric analogue of an analysis of variance *F* test statistic.

The test proceeds by first ranking the observations within each block. Let *A* denote the sum of the squared ranks, i.e., let

$$A = \sum_{i=1}^{k}\sum_{j=1}^{b}\text{Rank}\left(Y_{ij}\right)^2$$

where Rank(Y_{ij}) is the rank of the *i*-th observation within the *j*-th block, *b* = NB is the number of blocks, and *k* = NT is the number of treatments. Let

$$B = \frac{1}{b}\sum_{i=1}^{k}R_i^2$$

where

$$R_i = \sum_{j=1}^{b}\text{Rank}\left(Y_{ij}\right)$$

The Friedman test statistic (STAT(1)) is given by:

$$T = \frac{\left(k-1\right)\left(bB - b^2k\left(k+1\right)^2/4\right)}{A - bk\left(k+1\right)^2/4}$$

that, under the null hypothesis, has an approximate chi-squared distribution with *k* - 1 degrees of freedom. The asymptotic probability of obtaining a larger chi-squared random variable is returned in STAT(4).

If the *F* distribution is used in place of the chi-squared distribution, then the usual oneway analysis of variance *F*-statistic computed on the ranks is used. This statistic, reported in **STAT**(2), is given by

$$F = \frac{(b-1)T}{b(k-1)-T}$$

and asymptotically follows an *F* distribution with (*k* - 1) and (*b* - 1)(*k* - 1) degrees of freedom under the null hypothesis. **STAT**(5) is the asymptotic probability of obtaining a larger *F* random variable. (If *A* = *B*, **STAT**(1) and **STAT**(2) are set to machine infinity, and the significance levels are reported as $k!/(k!)^b$, unless this computation would cause underflow, in which case the significance levels are reported as zero.) Iman and Davenport (1980) discuss the relative advantages of the chi-squared and *F* approximations. In general, the *F* approximation is considered best.

The Friedman *T* statistic is related both to the Kendall coefficient of concordance and to the Spearman rank correlation coefficient. See Conover (1980) for a discussion of the relationships.

If, at the **α** = **ALPHA** level of significance, the Friedman test results in rejection of the null hypothesis, then an asymptotic test that treatments *i* and *j* are different is given by:
reject H_0 if $|R_i - R_j| >$ **D**, where

$$D = t_{1-\alpha/2}\sqrt{2b(A-B)/((b-1)(k-1))}$$

where *t* has (*b* - 1)(*k* - 1) degrees of freedom. Page's statistic (**STAT**(3)) is used to test the same null hypothesis as the Friedman test but is sensitive to a monotonic increasing alternative. The Page test statistic is given by

$$Q = \sum_{i=1}^{k} jR_i$$

It is largest (and thus most likely to reject) when the R_i are monotonically increasing.

Assumptions

The assumptions in the Friedman test are as follows:

1. The *k*-vectors of responses within each of the *b* blocks are mutually independent (i.e., the results within one block have no effect on the results within another block).
2. Within each block, the observations may be ranked.

The hypothesis tested is that each ranking of the random variables within each block is equally likely. The alternative is that at least one of the treatments tends to have larger values than one or more of the other treatments. The Friedman test is a test for the equality of treatment means or medians.

Comments

1. Workspace may be explicitly provided, if desired, by use of F2DMN/DF2DMN. The reference is:

   ```
   CALL F2DMN (NB, NT, Y, FUZZ, ALPHA, STAT, SMRNK, D, IWK, WK)
   ```

   The additional arguments are as follows:

   IWK — Integer work vector of length NT.

   WK — Work vector of length 2 * NT.

2. Informational errors

Type	Code	Description
4	5	At least one missing value was detected in Y. No missing values are permitted in this routine since it assumes a complete block design.
3	6	At least one tie was detected within a block.
3	7	The ranks of the treatments were exactly the same in all the blocks.

Example

The following example is taken from Bradley (1968), page 127, and tests the hypothesis that 4 drugs have the same effects upon a person's visual acuity. Five subjects were used.

```
      USE FRDMN_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NB, NT
      REAL       ALPHA, FUZZ
      PARAMETER  (ALPHA=0.05, FUZZ=0.001, NB=5, NT=4)
!
      INTEGER    NOUT
      REAL       D, SMRNK(NT), STAT(6), Y(NB*NT)
!
      DATA Y/.39, .55, .33, .41, .21, .28, .19, .16, .73, .69, .64, &
          .62, .41, .57, .28, .35, .65, .57, .53, .60/
!                                Perform Friedman's test
      CALL FRDMN (NB, NT, Y, FUZZ, ALPHA, STAT, SMRNK, D)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT, SMRNK, D
!
99999 FORMAT (' Friedman T.........', F8.2, /, ' Friedman F.........', &
            F8.2, /, ' Page test..........', F8.2, /, ' Prob ', &
            'Friedman T....', F11.5, /, ' Prob Friedman F....', &
            F11.5, /, ' Prob Page test.....', F11.5, /, ' Sum of ', &
            'Ranks.......', 4F8.2, /, ' D..................', F11.5)
!
      END
```

Output

```
    Friedman T.........    8.28
    Friedman F.........    4.93
    Page test..........  111.00
    Prob Friedman T....    0.04057
    Prob Friedman F....    0.01859
    Prob Page test.....    0.98495
    Sum of Ranks.......   16.00   17.00    7.00   10.00
    D..................    6.65638
```

The Friedman null hypothesis is rejected at the **α** = .05 while the Page null hypothesis is not. (A Page test with a monotonic decreasing alternative would be rejected, however.) Using **SMRNK** and **D**, one can conclude that treatment 3 is different from treatments 1 and 2, and that treatment 4 is different from treatment 2, all at the **α** = .05 level of significance.

QTEST

Performs a Cochran *Q* test for related observations.

Required Arguments

X — NOBS by NVAR matrix of dichotomized data, containing NOBS readings of zero or one on each of NVAR treatments. (Input)

Q — Cochran's *Q* statistic. (Output)

PQ — Asymptotic probability of exceeding Q under the null hypothesis of equality of the underlying populations. (Output)

Optional Arguments

NOBS — Number of blocks for each treatment. (Input)
Default: NOBS = size (X,1).

NVAR — Number of treatments. (Input)
Default: NVAR = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

FORTRAN 90 Interface

Generic: CALL QTEST (X, Q, PQ [, ...])
Specific: The specific interface names are S_QTEST and D_QTEST.

FORTRAN 77 Interface

Single: CALL QTEST (NOBS, NVAR, X, LDX, Q, PQ)
Double: The double precision name is DQTEST.

Description

Routine **QTEST** computes the Cochran *Q* test statistic that may be used to determine whether or not *M* matched sets of responses differ significantly among themselves. The data may be thought of as arising out of a randomized block design in which the outcome variable must be success (= 1.0) or failure (= 0.0). Within each block a multivariate vector of 1's or 0's is observed. The hypothesis is that the probability of success within a block does not depend upon the treatment.

Assumptions

1. The blocks are a random sample from the population of all possible blocks.
2. The outcome of each treatment is dichotomous.

Hypothesis

The hypothesis being tested may be stated in at least two ways.

1. H_0: All treatments have the same effect.
   H_1: The treatments do not all have the same effect.
2. Let p_{ij} denote the probability of outcome 1.0 in block *i*, treatment *j*.
   $H_0 : p_{i1} = p_{i2} = \ldots = p_{ic}$ for each *i*.
   $H_1 : p_{ij} \neq p_{ik}$ for some *i*, and some $j \neq k$
   where *c*(= **NVAR**) is the number of treatments.

The null hypothesis is rejected if Cochran's *Q* statistic is too large.

Comments

1. Informational errors

Type	Code	Description
4	5	X must consist of zeros and ones only.
3	6	X consists of either all ones or all zeros. Q is set to NaN (not a number). PQ is set to 1.0.

2. The input data must consist of zeros and ones only. For example, the data may be passfail information on **NVAR** questions asked of **NOBS** people or the test responses of **NOBS** individuals to **NVAR** different conditions.

3. The resulting statistic is distributed approximately as chi-squared with NVAR - 1 degrees of freedom if NOBS is not too small. NOBS greater than or equal to 5 * NVAR is a conservative recommendation.

Example

The following example is taken from Siegel (1956, page 164). It measures the responses of 18 housewives to 3 types of interviews.

```
      USE QTEST_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LDX, NVAR
      PARAMETER  (NVAR=3, LDX=18)
!
      INTEGER    NOUT
      REAL       PQ, Q, X(LDX,NVAR)
!
      DATA X/0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, &
          1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, &
          0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0/
!                                 Perform Cochran Q test
      CALL QTEST (X, Q, PQ)
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) Q, PQ
!
99999 FORMAT ('  Q =  ', F6.3, /, ' PQ = ', F9.5)
!
      END
```

Output

```
 Q =  16.667
PQ =   0.00024
```

KTRND

Performs *k*-sample trends test against ordered alternatives.

Required Arguments

NI — Vector of length NGROUP that contains the number of responses for each of the NGROUP groups. (Input)

X — Vector of length NI(1) + NI(2) + … + NI(NGROUP) containing the responses for each of the NGROUP groups. (Input)
All of the responses for group 1 come first, followed by group 2, and so on.

STAT — Vector of length 17 containing the test results. (Output)

I	STAT(I)
1	Test statistic (ties are randomized).
2	Conservative test statistic with ties counted in favor of the null hypothesis.
3	*p*-value associated with STAT(1).
4	*p*-value associated with STAT(2).
5	Continuity corrected STAT(3).
6	Continuity corrected STAT(4).
7	Expected mean of the statistic.
8	Expected kurtosis of the statistic. (The expected skewness is zero.
9	Total sample size.
10	Coefficient of rank correlation based upon STAT(1).
11	Coefficient of rank correlation based upon STAT(2).
12	Total number of ties between samples.
13	The *t*-statistic associated with STAT(3).
14	The *t*-statistic associated with STAT(4).
15	The *t*-statistic associated with STAT(5).
16	The *t*-statistic associated with STAT(6).
17	Degrees of freedom for each *t*-statistic.

Optional Arguments

NGROUP — Number of groups. (Input)
NGROUP must be greater than or equal to 3.
Default: **NGROUP** = size (**NI**,1).

FORTRAN 90 Interface

Generic: **CALL KTRND** (**NI**, **X**, **STAT** [, ...])
Specific: The specific interface names are **S_KTRND** and **D_KTRND**.

FORTRAN 77 Interface

Single: **CALL KTRND** (**NGROUP**, **NI**, **X**, **STAT**)
Double: The double precision name is **DKTRND**.

Description

Routine **KTRND** performs a *k*-sample trends test against ordered alternatives. The alternative to the null hypothesis of equality is that $F_1(\mathbf{X}) < F_2(\mathbf{X}) < \ldots F_k(\mathbf{X})$, where F_1, F_2, etc., are cumulative distribution functions, and the operator < implies that the less than relationship holds for all values of *X*. While the trends test used in **KTRND** requires that the background populations be continuous, ties occurring within a sample have no effect on the test statistic or associated probabilities. Ties between samples are important, however. Two methods for handling ties between samples are used. These are:

1. Ties are randomly split (**STAT**(1)).
2. Ties are counted in a manner that is unfavorable to the alternative hypothesis (**STAT** (2)).

Computational Procedure

Consider the matrices

$$M^{km} = \left(m_{ij}^{km}\right) = \begin{pmatrix} 2 & \text{if } X_{ki} < X_{mj} \\ 0 & \text{otherwise} \end{pmatrix}$$

where X_{ki} is the *i*-th observation in the *k*-th population, X_{mj} is the *j*-th observation in the *m*-th population, and each matrix M^{km} is n_k by n_m where n_i = **NI**(*i*). Let S_{km} denote the sum of all elements in M^{km}. Then, **STAT**(2) is computed as the sum over all elements in S_{km}, minus the expected value of this sum (computed as

$$\sum_{k<m} n_k n_m$$

when there are no ties and the distributions in all populations are equal). In **STAT**(1), ties are broken randomly, and the element in the summation is taken as 2.0 or 0.0 depending upon the result of breaking the tie.

STAT(3) and **STAT**(4) are computed using the *t* distribution. The probabilities reported are asymptotic approximations based upon the *t* statistics in **STAT**(13) and **STAT**(14), which are computed as in Jonckheere (1954, page 141). Similarly, **STAT**(5) and **STAT**(6) give the probabilities for **STAT**(15) and **STAT**(16), the continuity corrected versions of **STAT**(3) and **STAT**(4). The degrees of freedom for each *t* statistic (**STAT**(17)) are computed so as to make the *t* distribution selected as close as possible to the actual distribution of the statistic (see Jonckheere 1954, page 141).

STAT(7), the variance of the test statistic **STAT**(1), and **STAT**(8), the kurtosis of the test statistic, are computed as in Jonckheere (1954, page 138). The coefficients of rank correlation in **STAT**(9) and **STAT**(10) reduce to the Kendall τ statistic when there are just two groups.

Exact probabilities in small samples can be obtained from tables in Jonckheere (1954). Note, however, that the *t* approximation appears to be a good one.

Assumptions

1. The X_{mi} for each sample are independently and identically distributed according to a single continuous distribution.
2. The samples are independent.

Hypothesis tests

$H_0 : F_1(\mathbf{X}) \geq F_2(\mathbf{X}) \geq \ldots \geq F_k(\mathbf{X})$

$H_1 : F_1(\mathbf{X}) < F_2(\mathbf{X}) < \ldots < F_k(\mathbf{X})$

Reject if **STAT**(3) (or **STAT**(4), or **STAT**(5) or **STAT**(6), depending upon the method used) is too large.

Comments

1. Informational errors

Type	Code	Description
3	4	At least one tie is detected in X. Randomization is used to break all ties.
3	5	There are no degrees of freedom for the *t*-statistics. STAT(3) to STAT(6) are set to 0.

2. The closer **STAT**(10) and **STAT**(11) are to unity, the more one would be inclined to reject the hypothesis of randomness.

3. Routine **RNUN** (see Chapter 18, "Random Number Generation") is used to randomly break ties. Routine **RNSET** (see *Chapter 18*) can be used to initialize the seed of the random number generator. The routine **RNOPT** (see *Chapter 18*) can be used to select the form of the generator.

Example

The following example is taken from Jonckheere (1954, page 135). It involves four observations in four independent samples.

```
      USE RNSET_INT
      USE KTRND_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NGROUP
      PARAMETER  (NGROUP=4)
!
      INTEGER    NI(NGROUP), NOUT
      REAL       STAT(17), X(16)
!
      DATA NI/4, 4, 4, 4/
      DATA X/19, 20, 60, 130, 21, 61, 80, 129, 40, 99, 100, 149, 49, &
          110, 151, 160/
!
      CALL RNSET (123457)
!                                Get the statistics
      CALL KTRND (NI, X, STAT)
!                                Print the results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
!
99999 FORMAT (' STAT(1) - Test statistic (random) ..........', F8.1, &
            /, ' STAT(2) - Test statistic (null hypothesis) ..', &
            F8.1, /, ' STAT(3) - p-value for STAT(1) ............' &
            , F12.5, /, ' STAT(4) - p-value for STAT(2) ', &
            '............', F12.5, /, ' STAT(5) - Continuity ', &
            'corrected STAT(3) ......', F12.5, /, ' STAT(6) - ', &
            'Continuity corrected STAT(4) ......', F12.5, /, &
            ' STAT(7) - Expected mean ..................', F8.1, &
            /, ' STAT(8) - Expected kurtosis ..............', &
            F12.5, /, ' STAT(9) - Total sample size ..............' &
            , F8.1, /, ' STAT(10)- Rank corr. coef. based on STAT(1) ' &
            , '.', F12.5, /, ' STAT(11)- Rank corr. coef. based on ', &
```

```
                'STAT(2) .', F12.5, /, ' STAT(12)- Total number of ties ' &
                , '.............', F8.1, /, ' STAT(13)- t-statistic ', &
                'associated w/STAT(3) ..', F10.3, /, ' STAT(14)- ', &
                't-statistic associated w/STAT(4) ..', F10.3, /, &
                ' STAT(15)- t-statistic associated w/STAT(5) ..', F10.3, &
                /, ' STAT(16)- t-statistic associated w/STAT(6) ..', &
                F10.3, /, ' STAT(17)- Degrees of freedom ..............' &
                , F10.3)
    !
          END
```

Output

```

    STAT(1) - Test statistic (random) ..........    46.0
    STAT(2) - Test statistic (null hypothesis) ..    46.0
    STAT(3) - p-value for STAT(1) ..............     0.01483
    STAT(4) - p-value for STAT(2) ..............     0.01483
    STAT(5) - Continuity corrected STAT(3) .....     0.01683
    STAT(6) - Continuity corrected STAT(4) .....     0.01683
    STAT(7) - Expected mean ....................   458.7
    STAT(8) - Expected kurtosis ................    -0.15365
    STAT(9) - Total sample size ................    16.0
    STAT(10)- Rank corr. coef. based on STAT(1) .     0.47917
    STAT(11)- Rank corr. coef. based on STAT(2) .     0.47917
    STAT(12)- Total number of ties .............     0.0
    STAT(13)- t-statistic associated w/STAT(3) ..     2.264
    STAT(14)- t-statistic associated w/STAT(4) ..     2.264
    STAT(15)- t-statistic associated w/STAT(5) ..     2.208
    STAT(16)- t-statistic associated w/STAT(6) ..     2.208
    STAT(17)- Degrees of freedom ...............    36.050
```

Tests of Goodness of Fit and Randomness

Routines

Usage Notes

The routines in this chapter are used to test for goodness of fit and randomness. The goodness-of-fit tests are described in Conover (1980). There are two goodness-of-fit tests for general distributions, a Kolmogorov-Smirnov test and a chi-squared test. The user supplies the hypothesized cumulative distribution function for these two tests. There is one routine (Lilliefors) that can be used to test specifically for exponential distributions and five routines (Shapiro-Wilk, Lilliefors, Mardia, Anderson-Darling, and Cramer-von Mises) that can be used to test specifically for normal distributions.

The tests for randomness are often used to evaluate the adequacy of pseudorandom number generators. These tests are discussed in Knuth (1981).

The Kolmogorov-Smirnov routines in this chapter compute exact probabilities in small to moderate sample sizes. The chi-squared goodness-of-fit test may be used with discrete as well as continuous distributions.

The Kolmogorov-Smirnov and chi-squared goodness-of-fit test routines allow for missing values (NaN, not a number) in the input data. The routines that test for randomness do not allow for missing values.

KSONE

Performs a Kolmogorov-Smirnov one-sample test for continuous distributions.

Required Arguments

CDF — User-supplied **FUNCTION** to compute the cumulative distribution function (CDF) at a given value. The form is **CDF**(**Y**), where

Y – Value at which **CDF** is to be evaluated. (Input)
CDF – Value of **CDF** at **Y**. (Output)

CDF must be declared **EXTERNAL** in the calling program.

X — Vector of length **NOBS** containing the observations. (Input)

PDIF — Vector of length 6 containing the output statistics. (Output)

I	**PDIF(I)**
1	D_n = Maximum of $\left(D_n^+, D_n^-\right)$
2	D_n^+ Maximum difference between the theoretical and empirical CDF's
3	D_n^- Maximum difference between the empirical and theoretical CDF's
4	$Z = \sqrt{NOBS} \ * \ (PDIF(1))$
5	Probability of the statistic exceeding D_n under the null hypothesis of equality and against the one-sided alternative. An exact probability is computed for NOBS $\leq$ 80, and an approximate probability is computed for NOBS > 80. See function AKS1DF (Chapter 17, "Probability Distribution Functions and Inverses").
6	Probability of the statistic exceeding D_n under the null hypothesis of equality and against the two-sided alternative. This probability is twice the probability reported in PDIF(5), (or 1.0 if 2 * PDIF(5) is greater than 1.0). This approximation is nearly exact when PDIF(5) is less than 0.10.

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**X**,1).

NMISS — Number of missing (NaN, not a number) values. (Output)

FORTRAN 90 Interface

Generic: CALL KSONE (CDF, X, PDIF [, ...])
Specific: The specific interface names are S_KSONE and D_KSONE.

FORTRAN 77 Interface

Single: CALL KSONE (CDF, NOBS, X, PDIF, NMISS)
Double: The double precision name is DKSONE.

Description

The routine KSONE performs a Kolmogorov-Smirnov goodness-of-fit test in one sample. The hypotheses tested follow:

- $H_0 : F(x) = F^*(x) \quad H_1 : F(x) \neq F^*(x)$
- $H_0 : F(x) \geq F^*(x) \quad H_1 : F(x) < F^*(x)$
- $H_0 : F(x) \leq F^*(x) \quad H_1 : F(x) > F^*(x)$

where *F* is the cumulative distribution function (CDF) of the random variable, and the theoretical CDF, *F**, is specified via the user-supplied FUNCTION CDF. Let *n* = NOBS - NMISS. The test statistics for both one-sided alternatives

$$D_n^+ = \text{PDIF}(2)$$

and

$$D_n^- = \text{PDIF}(3)$$

and the two-sided (D_n = PDIF(1)) alternative are computed as well as an asymptotic *z*-score (PDIF(4)) and *p*-values associated with the one-sided (PDIF(5)) and two-sided (PDIF(6)) hypotheses. For $n > 80$, asymptotic *p*-values are used (see Gibbons 1971). For $n \leq 80$, exact one-sided *p*-values are computed according to a method given by Conover (1980, page 350). An approximate two-sided test *p*-value is obtained as twice the one-sided *p*-value. The approximation is very close for one-sided *p*-values less than 0.10 and becomes very bad as the one-sided *p*-values get larger.

Comments

1. Workspace may be explicitly provided, if desired, by use of K2ONE/DK2ONE. The reference is:

   CALL K2ONE (CDF, NOBS, X, PDIF, NMISS, XWK)

The additional argument is:

XWK — Work vector of length 3 * (NOBS + 1) if NOBS ≤ 80, or of length NOBS if NOBS > 80.

2. Informational errors

Type	Code	Description
4	2	PDIF, the output cumulative distribution value from CDF, must be greater than or equal to 0.0 and less than or equal to 1.0 (by definition of a probability distribution function).
4	3	At least one tie is detected in X. Ties are not allowed in KSONE.
4	4	PDIF, the output cumulative distribution value from CDF, cannot decrease with increasing X (by the definition of a cumulative distribution function).
4	6	All the elements of X are missing (NaN, not a number) values.

3. No check is made for the validity of the input data. Thus, although one or more of the X(I) may be inconsistent with the distribution in that an observation may be outside of the range of the distribution, KSONE will not detect the anomaly (unless the user causes it to be detected via the function CDF).

Programming Notes

1. The theoretical CDF is assumed to be continuous. If the CDF is not continuous, the statistics

$$D_n^*$$

will not be computed correctly.

2. Estimation of parameters in the theoretical CDF from the sample data will tend to make the *p*-values associated with the test statistics too liberal. The empirical CDF will tend to be closer to the theoretical CDF than it should be.

3. No attempt is made to check that all points in the sample are in the support of the theoretical CDF. If all sample points are not in the support of the CDF, the null hypothesis must be rejected.

4. The user must supply an external FUNCTION that calculates the theoretical CDF for a given abscissa. The calling program must contain an EXTERNAL statement with the name of this routine. Often, IMSL functions in Chapter 17, "Probability Distribution Functions and Inverses" may be used. Examples of possible user-supplied routines follow. Each FORTRAN function would be preceded by the statement

```
REAL FUNCTION CDF(X)
```

and ended by a RETURN and an END statement.

a. Normal (μ, σ^2) $Z = (X - \mu)/\sigma$
CDF = ANORDF(*Z*)

b. Uniform[*a*, *b*]
```
If(X .LT. a) THEN
 CDF = 0.0
ELSE IF(X .GT. b) THEN
 CDF = 1.0
ELSE
 CDF = (X - a)/(b - a)
END IF
```

c. Minimum of *n* CDF = 1.0 - (1.0 - *X*)**n
Uniform(0, 1) random numbers

Example

In this example, a random sample of size 100 is generated via routine **RNUN** (see Chapter 18, "Random Number Generation") for the uniform (0, 1) distribution. We want to test the null hypothesis that the CDF is the standard normal distribution with a mean of 0.5 and a variance equal to the uniform (0, 1) variance (1/12).

```
      USE RNSET_INT
      USE RNUN_INT
      USE KSONE_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    ISEED, NOBS
      PARAMETER  (ISEED=123457, NOBS=100)
!
      INTEGER    NMISS, NOUT
      REAL       CDF, PDIF(6), X(100)
      EXTERNAL   CDF
!                                  Generate the sample
      CALL RNSET (ISEED)
      CALL RNUN (X)
!
      CALL KSONE (CDF, X, PDIF, NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) NMISS, PDIF
99999 FORMAT ('NMISS = ', I4/' D     = ', F8.4/' D+    = ', F8.4/ &
            ' D-    = ', F8.4/' Z     = ', F8.4/' Prob greater D', &
            ' one-sided = ', F8.4/' Prob greater D two-sided = ', &
            F8.4)
      END
!
!                                   The CDF
!
      REAL FUNCTION CDF (X)
      REAL       X
!
```

```
      REAL       AMEAN, STD
      PARAMETER  (AMEAN=0.50, STD=0.2886751)
!
      REAL       ANORDF, Z
      EXTERNAL   ANORDF
!                                     Standardize
      Z = (X-AMEAN)/STD
!                                  Get the probability
      CDF = ANORDF(Z)
!
      RETURN
      END
```

Output

```
 NMISS =    0
 D     =   0.1471
 D+    =   0.0810
 D-    =   0.1471
 Z     =   1.4708
 Prob greater D one-sided =   0.0132
 Prob greater D two-sided =   0.0264
```

CHIGF

Performs a chi-squared goodness-of-fit test.

Required Arguments

CDF — User-supplied **FUNCTION** to compute the cumulative distribution function (CDF) at a given value. The form is **CDF**(**Y**), where

Y – Value at which the CDF is to be evaluated. (Input)
CDF – Value of the CDF at Y. (Output)

CDF must be declared EXTERNAL in the calling program.

NELM — The absolute value of **NELM** is the number of data elements currently input in **X**. (Input)
NELM may be positive, zero, or negative. Negative **NELM** means delete the **-NELM** data elements from the analysis.

X — Vector of length |**NELM**| containing the data elements for this call. (Input)
If the data element is missing (NaN, not a number), then the observation is ignored.

NCAT — The absolute value of **NCAT** is the number of cells into which the observations are to be tallied. (Input)
If **NCAT** is negative, then **CHIGF** chooses the cutpoints in **CUTP** so that the cells are equiprobable in continuous distributions. **NCAT** should not be negative in discrete distributions. The user must be careful to define cutpoints in discrete distributions since no error message can be generated in this situation if **NCAT** is negative.

RNGE — Vector of length 2 containing the lower and upper endpoints of the range of the distribution, respectively. (Input)
If the lower and upper endpoints are equal, a range on the whole real line is used. If the lower and upper endpoints are different, points outside of the range are ignored so that distributions conditional on the range can be used. In this case, the point **RNGE**(1) is excluded from the first interval, but the point **RNGE**(2) is included in the last interval.

NDFEST — Number of parameters estimated in computing the CDF. (Input)

CUTP — Vector of length |**NCAT**| **-** 1 containing the cutpoints defining the cells. (Input, if **NCAT** is positive, output, otherwise)
|**NCAT**| **-** 1 cutpoints define the cells to be used. If **NCAT** is positive, then the cutpoints are input by the user. The intervals defined by the cutpoints are such that the lower endpoint is not included while the upper endpoint is included in the interval.

P — *p*-value for the chi-squared statistic in CHISQ(|NCAT| + 1). (Output)
This chi-squared statistic has **DF** degrees of freedom.

Optional Arguments

IDO — Processing option. (Input)
Default: **IDO** = 0.

IDO	Action
0	This is the only call to CHIGF, and all of the data are input on this call.
1	This is the first call to CHIGF, and additional calls to CHIGF will be made. Initialization and updating for the data in X are performed.
2	This is an intermediate call to CHIGF. Updating for the data in X is performed.
3	This is the final call to CHIGF. Updating for the data in X and wrap-up computations are performed.

Calls to **CHIGF** with **IDO** = 2 or 3 may be intermixed. It is permissible for a call with **IDO** = 2 to follow a call with **IDO** = 3.

FRQ — Vector containing the frequencies. (Input)
If the first element of **FRQ** is -1.0, then all frequencies are taken to be 1 and **FRQ** is of length 1. Otherwise, **FRQ** is of length |**NELM**|, and the elements in **FRQ** contain the frequency of the corresponding observation in **X**. If the frequency is missing (NaN, not a number) (and **FRQ**(1) is not -1.0), the observation is ignored.
Default: **FRQ**(1) = -1.0.

COUNTS — Vector of length |**NCAT**| containing the counts in each of the cells. (Output, if **IDO** = 0 or 1; input/output, if **IDO** > 1)

EXPECT — Vector of length |**NCAT**| containing the expected count in each cell. (Output, if **IDO** = 0 or 3; not referenced otherwise)

CHISQ — Vector of length |**NCAT**| + 1 containing the contributions to chi-squared. (Output, if **IDO** = 0 or 3, not referenced otherwise)
Elements 1 through |**NCAT**| contain the contributions to chi-squared for the corresponding cell. Element |**NCAT**| + 1 contains the total chi-squared statistic.

DF — Degrees of freedom in chi-squared. (Output)

FORTRAN 90 Interface

Generic: **CALL CHIGF** (**CDF**, **NELM**, **X**, **NCAT**, **RNGE**, **NDFEST**, **CUTP**, **P** [, ...])

Specific: The specific interface names are **S_CHIGF** and **D_CHIGF**.

FORTRAN 77 Interface

Single: **CALL CHIGF** (**IDO**, **CDF**, **NELM**, **X**, **FRQ**, **NCAT**, **RNGE**, **NDFEST**, **CUTP**, **COUNTS**, **EXPECT**, **CHISQ**, **P**, **DF**)

Double: The double precision name is **DCHIGF**.

Description

Routine **CHIGF** performs a chi-squared goodness-of-fit test that a random sample of observations is distributed according to a specified theoretical cumulative distribution. The theoretical distribution, which may be continuous, discrete, or a mixture of discrete and continuous distributions, is specified via a user-defined **FUNCTION**. Because the user is allowed to specify a range for the observations, a test that is conditional upon the specified range is performed.

|**NCAT**| gives the number of intervals into which the observations are to be divided. These intervals can be specified via the vector **CUTP**, which contains the cutpoints (or endpoints) for the intervals. Or if **NCAT** is negative, equiprobable intervals computed by **CHIGF** can be used. Regardless of the method used to obtain them, the intervals are such that the lower endpoint is not included in the interval while the upper endpoint is always included. The user should determine the cutpoints when the cumulative distribution function has discrete elements since **CHIGF** cannot determine them in this case. Regardless of how the cutpoints are determined, the lower endpoint of the first interval is specified by **RNGE**(1) when **RNGE**(1) ≠ **RNGE**(2) and is given as minus machine infinity otherwise. The upper endpoint of the last interval is defined similarly.

Routine **CHIGF** tallies the observations in X as follows. If the cutpoints are determined by **CHIGF**, then the cumulative probability at x_i, $F(x_i)$, is computed via function **CDF**. The tally for x_i is made in interval number $\lfloor mF(x) + 1 \rfloor$, where m = |**NCAT**| and $\lfloor\ \rfloor$ is the function that takes the greatest integer that is no larger than the argument of the function. If the cutpoints are specified by the user, the tally is made in the interval to which x_i belongs using the endpoints specified by the user. Thus, if the computer time required to calculate the cumulative distribution function is large, user-specified cutpoints may be preferred in order to reduce the total computing time.

If the expected count in any cell is less than 1, then a rule of thumb is that the chi-squared approximation may be suspect. A warning message to this effect is issued in this case, as well as when an expected value is less than 5.

Programming Notes

The user must supply a function **CDF** with calling sequence **CDF**(**Y**), which returns the value of the cumulative distribution function at any point **Y** in the range of the distribution. The supplied function must be declared in an **EXTERNAL** statement in the calling program. Many of the IMSL cumulative distribution functions in Chapter 17,

"Probability Distribution Functions and Inverses" can be used for **CDF**, either directly, if the calling sequence is correct, or indirectly, if, for example, the sample means and standard deviations are to be used in computing the theoretical **CDF**.

Comments

Informational errors

Type	Code	Description
4	4	There are more observations deleted from a cell than added.
4	5	All observations are missing.
3	6	An expected value is less than 1.
3	7	An expected value is less than 5.
4	8	The function CDF is not a cumulative distribution function.
4	9	The probability of the range of the distribution is not positive.
4	10	An error has occurred when inverting the cumulative distribution function. This function must be continuous and defined over the whole real line. If all else fails, you must specify the cutpoints (i.e., NCAT must be positive).

Examples

Example 1

In this example, a discrete binomial random sample of size 1000 with binomial parameter p = 0.3 and binomial sample size 5 is generated via routine **RNBIN** (see Chapter 18, "Random Number Generation"). Routine **RNSET** is first used to set the seed. One call to **CHIGF** is made. Routine **BINDF** (see Chapter 17, "Probability Distribution Functions and Inverses") is used to compute the **CDF**.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    ISEED, NCAT, NDFEST, NELM
      PARAMETER  (ISEED=123457, NCAT=6, NDFEST=0, NELM=1000)
!
      INTEGER    I, IX(NELM), NOUT
      REAL       CDF, CHISQ(NCAT+1), COUNTS(NCAT), CUTP(NCAT-1), DF, &
                 EXPECT(NCAT), P, RNGE(2), X(NELM)
      EXTERNAL   CDF
!
      DATA RNGE/0.0, 0.0/
      DATA CUTP/.5, 1.5, 2.5, 3.5, 4.5/
!
      CALL RNSET (ISEED)
```

```
!                                  Generate the data
      CALL RNBIN (5, 0.3, IX)
      DO 10  I=1, NELM
         X(I) = IX(I)
   10 CONTINUE
!
      CALL CHIGF (CDF, NELM, X, NCAT, RNGE, NDFEST, CUTP, P, &
                  COUNTS=COUNTS, EXPECT=EXPECT, CHISQ=CHISQ, DF=DF)
!                                  Print results
      CALL WRRRN ('Counts', COUNTS, 1, NCAT, 1)
      CALL WRRRN ('Expect', EXPECT, 1, NCAT, 1)
      CALL WRRRN ('Contributions to Chi-squared', CHISQ, 1, NCAT, 1)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) CHISQ(NCAT+1), P, DF
99999 FORMAT (///'0Chi-squared       ', F8.4, /, ' P-value           ' &
            , F8.4, /, ' Degrees of freedom', F8.4)
      END
!
      REAL FUNCTION CDF (Y)
      REAL       Y
!
      INTEGER    I
      REAL       BINDF
      EXTERNAL   BINDF
!
      I   = Y
      CDF = BINDF(I,5,0.3)
      RETURN
      END
```

Output

```
*** WARNING  ERROR 7 from CHIGF.  An expected value is less than 5.

                    Counts
    1       2       3       4       5       6
170.0   331.0   320.0   148.0    28.0     3.0

                    Expect
    1       2       3       4       5       6
168.1   360.2   308.7   132.3    28.3     2.4

          Contributions to Chi-squared
    1       2       3       4       5       6
0.022   2.359   0.414   1.863   0.004   0.134

Chi-squared         4.7963
P-value             0.4412
Degrees of freedom  5.0000
```

Example 2

This example illustrates the use of **CHIGF** on a randomly generated sample from the normal distribution. One thousand randomly generated observations are tallied into 10 equiprobable intervals. Twelve calls to **CHIGF** are made. The first call is solely for initialization since **IDO** = 1 and **NROW** = 0. The next 10 calls tally the data, 100

observations at a time, with **IDO** = 2 and **NROW** = 100. The last call is for wrap up only since **IDO** = 3 and **NROW** = 0. All twelve calls could have been replaced with one call to **CHIGF** with **IDO** = 0 and **NROW** = 1000. **X** would need to be of length 1000 if one call were used. In this example, the null hypothesis is not rejected.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    ISEED, NCAT, NDFEST
      PARAMETER  (ISEED=123457, NCAT=-10, NDFEST=0)
!
      INTEGER    I, IDO, NOUT, NELM
      REAL       CHISQ(-NCAT+1), COUNTS(-NCAT), CUTP(-NCAT-1), &
                 DF, EXPECT(-NCAT), P, RNGE(2), X(100)
!
      DATA RNGE/0.0, 0.0/
!
      CALL RNSET (ISEED)
!                                Initialization
      IDO  = 1
      NELM = 0
      CALL CHIGF (S_ANORDF, NELM, X, NCAT, RNGE, NDFEST, CUTP, P,&
                  IDO=IDO, COUNTS=COUNTS, EXPECT=EXPECT,&
                  CHISQ=CHISQ, DF=DF)
!                                Add the data
      IDO  = 2
      NELM = 100
      DO 10  I=1, 10
         CALL RNNOR (X)
         CALL CHIGF (S_ANORDF, NELM, X, NCAT, RNGE, NDFEST, CUTP, P, &
                     IDO=IDO, COUNTS=COUNTS, EXPECT=EXPECT, &
                     CHISQ=CHISQ, DF=DF)
   10 CONTINUE
!                                Wrap up
      IDO  = 3
      NELM = 0
      CALL CHIGF (S_ANORDF, NELM, X, NCAT, RNGE, NDFEST, CUTP, &
                  P, IDO=IDO, COUNTS=COUNTS, EXPECT=EXPECT, &
                  CHISQ=CHISQ, DF=DF)
!                                Print results
      CALL WRRRN ('Cutpoints', CUTP, 1, -NCAT, 1)
      CALL WRRRN ('Counts', COUNTS, 1, -NCAT, 1)
      CALL WRRRN ('Expect', EXPECT, 1, -NCAT, 1)
      CALL WRRRN ('Contributions to Chi-squared', CHISQ, 1, -NCAT, 1)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) CHISQ(-NCAT+1), P, DF
99999 FORMAT (///'0Chi-squared       ', F8.4, /, ' P-value           ' &
            , F8.4, /, ' Degrees of freedom', F8.4)
      END
```

Output

```
                              Cutpoints
     1       2       3       4       5       6       7       8       9
-1.282  -0.842  -0.524  -0.253   0.000   0.253   0.524   0.842   1.282

                                Counts
    1       2       3       4       5       6       7       8       9      10
```

```
    106.0   109.0    89.0    92.0    83.0    87.0   110.0   104.0   121.0    99.0

                                 Expect
    1       2       3       4       5       6       7       8       9      10
    100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0   100.0

                        Contributions to Chi-squared
    1       2       3       4       5       6       7       8       9      10
    0.360   0.810   1.210   0.640   2.890   1.690   1.000   0.160   4.410   0.010

    Chi-squared        13.1806
    P-value             0.1546
    Degrees of freedom  9.0000
```

SPWLK

Performs a Shapiro-Wilk *W*-test for normality.

Required Arguments

X — Vector of length **NOBS** containing the observations. (Input)

W — Shapiro Wilk *W* statistic. (Output)

P — *P* -value for a test of normality. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be in the range from 3 to 2000 inclusive.
Default: **NOBS** = size (**X**,1).

NMISS — Number of missing observations. (Output)

FORTRAN 90 Interface

Generic: `CALL SPWLK (X, W, P [, ...])`
Specific: The specific interface names are **S_SPWLK** and **D_SPWLK**.

FORTRAN 77 Interface

Single: `CALL SPWLK (NOBS, X, W, P, NMISS)`
Double: The double precision name is **DSPWLK**.

Description

Routine **SPWLK** computes the Shapiro-Wilk *W*-statistic for testing for normality. This test is thought to be one of the best omnibus tests of normality (see D'Agostino and Stevens 1986, page 406). Routine **SPWLK** is based upon the approximations and code given by Royston (1982a, b, c). It may be used in samples as large as 2000, or as small as 3. In the Shapiro and Wilk test, *W* is given by**.**

$$W = \left\{ \sum_{i=1}^{n} a_i x_{(i)} \right\}^2 / \sum_{i=1}^{n} \left(x_i - \overline{x} \right)^2$$

where $x_{(i)}$ is the i-th largest order statistic,

$$\overline{x}$$

is the sample mean, and n is the number of observations. Royston (1982) gives approximations and tabled values which may be used to compute the coefficients a_i, i = 1, K, n, and obtain the significance level of the W statistic.

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2WLK/DS2WLK**. The reference is:

   ```
   CALL S2WLK (NOBS, X, W, P, NMISS, WK)
   ```

   The additional argument is:

   WK — Work vector of length **NOBS**. If **X** is not needed, then **WK** and **X** can share the same storage locations. On output, **WK** will contain the sorted nonmissing elements of **X**. If **X** is sorted, **WK** is not used.

2. Informational errors

Type	Code	Description
4	2	There are too many missing (NaN, "not a number") values in X for the test to be performed.
3	3	All observations in X are tied.

Example

The following example is taken from Conover (1980, pages 364 and 195). The data consists of 50 two digit numbers taken from a telephone book. The W test fails to reject the null hypothesis of normality at the .05 level of significance.

```
      USE SPWLK_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NMISS, NOBS, NOUT
      PARAMETER  (NOBS=50)
      REAL       P, W, X(NOBS)
!
      DATA X/23, 36, 54, 61, 73, 23, 37, 54, 61, 73, 24, 40, 56, 62, &
          74, 27, 42, 57, 63, 75, 29, 43, 57, 64, 77, 31, 43, 58, 65, &
          81, 32, 44, 58, 66, 87, 33, 45, 58, 68, 89, 33, 48, 58, 68, &
          93, 35, 48, 59, 70, 97/
```

```
      !
            CALL SPWLK (X, W, P, NMISS=NMISS)
      !                                 Write out results
            CALL UMACH(2, NOUT)
            WRITE(NOUT,5) W, P, NMISS
          5 FORMAT(/ ' W     = ', F6.4 / ' P     = ', F6.4 / &
                   ' NMISS = ',I3)
            END
```

Output

```
      W     = 0.9642
      P     = 0.2309
      NMISS =   0
```

LILLF

Performs Lilliefors test for an exponential or normal distribution.

Required Arguments

X — Vector of length NOBS containing the observations. (Input)

XMEAN — Sample mean. (Output)

STD — Sample standard deviation. (Output)

DIF — Maximum absolute difference between the empirical and the theoretical distributions. (Output)

PROB — Approximate probability of a greater DIF. (Output)
Probabilities less than 0.01 are reported as 0.01. Probabilities greater than 0.15 for the exponential distribution or greater than 0.10 for the normal distribution are reported as 0.5. Otherwise an approximate probability is computed.

NMISS — Number of missing (NaN, not a number) values. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than 4.
Default: NOBS = size (X,1).

IPDF — Distribution option. (Input)
IPDF = 0 means a test for normality is to be performed. IPDF = 1 means a test for the exponential distribution is to be performed.
Default: IPDF = 0.

FORTRAN 90 Interface

Generic: CALL LILLF (X, XMEAN, STD, DIF, PROB, NMISS [, ...])
Specific: The specific interface names are S_LILLF and D_LILLF.

FORTRAN 77 Interface

Single: CALL LILLF (NOBS, X, IPDF, XMEAN, STD, DIF, PROB, NMISS)

Double: The double precision name is **DLILLF**.

Description

Routine **LILLF** computes Lilliefors test and its *p*-values for either a normal distribution in which both the mean and variance are estimated, or an exponential distribution in which the mean is estimated. Routine **LILLF** uses a modified version of IMSL routine **KSONE** to compute the one-sample two-sided Kolmogorov-Smirnov statistic *D* (**DIF**). *p*-values are then computed for the exponential distribution via linear interpolation on the tabled values given by Stephens (1974). For the normal distribution, *p*-values are computed using an analytic approximation given by Dallal and Wilkinson (1986). Because Stephens' (1974) tables are in the inclusive range (0.01, 0.15) and Dallal and Wilkinson (1986) give approximations in the range (0.01, 0.10), if the computed probability of a greater *D* is less than 0.01, a level 1 message is issued (such messages are not generally printed, see the Reference Material) and the probability is set to 0.01. Similarly, if the probability is greater than 0.15 (0.10 for the normal), a level 1 message is issued and the *p*-value is set to 0.50. Note that because parameters are estimated, *p*-values in Lilliefors test are not the same as in the Kolmogorov-Smirnov test.

Observations from exponential or normal distributions should not be tied. If tied observations are found, an informational message is printed. Printing of this message can be turned off via a call to routine **ERSET** as is discussed in the Reference Material.

A general reference for Lilliefors test is Conover (1980). The original reference for the test for normality is Lilliefors (1967), while Lilliefors (1969) introduces the test for the exponential distribution.

Comments

1. Workspace may be explicitly provided, if desired, by use of **L2LLF/DL2LLF**. The reference is:

   ```
   CALL L2LLF (NOBS, X, IPDF, XMEAN, STD, DIF, PROB, NMISS, XWK)
   ```

   The additional argument is:

   XWK — Work vector of length **NOBS**.

2. Informational errors

Type	Code	Description
1	1	The computed probability of DIF is greater than 0.15 for an exponential distribution. PROB is set to 0.50.
1	2	The computed probability of DIF is less than the tabled probability of 0.01. PROB is set to 0.01.
1	3	The computed probability of DIF is greater than 0.10 for a normal distribution. PROB is set to 0.50.
1	4	The computed probability of DIF is less than 0.01. PROB is set to 0.01.
4	5	A negative value is encountered in X when IPDF = 1. Negative values are impossible for exponential distributions.
4	6	All elements in *X* are tied.

Example

The following example is taken from Conover (1980, page 358). It consists of 50 observations drawn at random from a telephone book. In this example, the null hypothesis is accepted. Note that the computed probability is outside the range (0.01, 0.10), and has thus been set to .50. Because many observations in X are tied, a warning message is issued. The printing of this message can be turned off through the use of routine ERSET (Reference Material).

```
      USE LILLF_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=50)
!
      INTEGER    NMISS, NOUT
      REAL       DIF, PROB, STD, X(NOBS), XMEAN
!
      DATA X/23, 23, 24, 27, 29, 31, 32, 33, 33, 35, 36, 37, 40, 42, &
          43, 43, 44, 45, 48, 48, 54, 54, 56, 57, 58, 57, 58, 58, 58, &
          59, 61, 61, 62, 63, 64, 65, 66, 68, 68, 70, 73, 73, 74, 75, &
          77, 81, 87, 89, 93, 97/
!
      CALL LILLF (X, XMEAN, STD, DIF, PROB, NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' XMEAN = '', F9.2, /, '' STD = '', F12.3, /,  '// &
           '     '' DIF = '', F13.4, /, '' PROB = '', F12.4, /,     '// &
           '  '' NMISS = '', I6)') XMEAN, STD, DIF, PROB, NMISS
      END
```

Output

```
 *** WARNING  ERROR 3 from L4LLF.  Two or more elements in X are tied.
     Here is a traceback of subprogram calls in reverse order:
```

```
        Routine name                    Error type  Error code
        ------------                    ----------  ----------
        L4LLF                               6           3    (Called internally)
        L3LLF                               0           0    (Called internally)
        L2LLF                               0           0    (Called internally)
        LILLF                               0           0
        USER                                0           0
    XMEAN =     55.04
    STD =       19.005
    DIF =        0.0811
    PROB =       0.5000
    NMISS =      0
```

MVMMT

Computes Mardia's multivariate measures of skewness and kurtosis and test for multivariate normality.

Required Arguments

X — NOBS by NVAR+ *m* matrix containing the data. (Input)
m is 0, 1, or 2 depending upon whether any columns in X contain frequencies or weights.

IND — Vector of length NVAR containing the column numbers in X for which statistics are desired. (Input)

STAT — Vector of length 13 containing the output statistics. (Output)
If a statistic is not computed, the corresponding element of STAT is set to not a number (NaN).

STAT(1) = estimated skewness.

STAT(2) = expected skewness assuming a multivariate normal distribution.

STAT(3) = asymptotic chi-squared statistic assuming a multivariate normal distribution.

STAT(4) = probability of a greater chi-squared.

STAT(5) = Mardia and Foster's standard normal score for skewness.

STAT(6) = estimated kurtosis.

STAT(7) = expected kurtosis assuming a multivariate normal distribution.

STAT(8) = asymptotic standard error of the estimated kurtosis.

STAT(9) = standard normal score obtained from STAT(6) through STAT(8).

STAT(10) = *p*-value corresponding to STAT(9).

STAT(11) = Mardia and Foster's standard normal score for kurtosis.

STAT(12) = Mardia's S_W statistic based upon STAT(5) and STAT(11).

STAT(13) = *p*-value for STAT(12).

STAT(12) and STAT(13) are only computed when ICMPUT = 0.

Optional Arguments

NOBS — Number of rows of data in X. (Input)
Default: NOBS = size (X,1).

NVAR — Dimensionality of the multivariate space for which the skewness and kurtosis are to be computed. (Input)
Default: NVAR = size (IND,1).

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. Positive IFRQ indicates that column number IFRQ of X contains the frequencies. All frequencies should be integer values. The NINT (nearest integer) function is used to obtain integer frequencies if this is not the case.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means that all weights are 1.0. Positive IWT means that column IWT of X contains the weights. Negative weights are not allowed.
Default: IWT = 0.

ICMPUT — Option parameter giving the statistics to compute. (Input)
Default: ICMPUT = 0.

ICMPUT	Output Statistics
0	Both skewness and kurtosis.
1	Kurtosis only.
2	Skewness only.

NI — The sum of the frequencies of all observations used in the computations. (Output)

SWT — The sum of the weights times the frequencies for all observations used in the computations. (Output)

XMEAN — Vector of length NVAR containing the sample means. (Output)

R — NVAR by NVAR upper triangular matrix containing the Cholesky $R^T R$ factorization of the covariance matrix. (Output)

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)

NRMISS — Number of rows of data in X containing any missing values (NaN, not a number). (Output)
Rows with missing values in the columns IND, IFRQ, and IWT are excluded from the analysis.

FORTRAN 90 Interface

Generic: CALL MVMMT (X, IND, STAT [, ...])

Specific: The specific interface names are S_MVMMT and D_MVMMT.

FORTRAN 77 Interface

Single: CALL MVMMT (NOBS, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, ICMPUT, NI, SWT, XMEAN, R, LDR, STAT, NRMISS)

Double: The double precision name is DMVMMT.

Description

Routine MVMMT computes Mardia's (1970) measures $b_{1,p}$ and $b_{2,p}$ of multivariate skewness and kurtosis, respectfully, for p = NVAR. These measures are then used in computing tests for multivariate normality. Three test statistics, one based upon $b_{1,p}$ alone, one based upon $b_{2,p}$ alone, and an omnibus test statistic formed by combining normal scores obtained from $b_{1,p}$ and $b_{2,p}$ are computed. On the order of np^3, operations are required in computing $b_{1,p}$ when the method of Isogai (1983) is used, where n = NOBS. On the order of np^2, operations are required in computing $b_{2,p}$.

Let

$$d_{ij} = \sqrt{w_i w_j}\left(x_i - \overline{x}\right)^T S^{-1}\left(x_j - \overline{x}\right)$$

where

$$S = \frac{\sum_{i=1}^{n} w_i f_i (x_i - \overline{x})(x_i - \overline{x})^T}{\sum_{i=1}^{n} f_i}$$

$$\overline{x} = \frac{1}{\sum_{i=1}^{n} w_i f_i} \sum_{i=1}^{n} w_i f_i x_i$$

f_i is the frequency of the i-th observation, and w_i is the weight for this observation. (Weights w_i are defined such that x_i is distributed according to a multivariate normal, $N(\boldsymbol{\mu}, \boldsymbol{\Sigma}/w_i)$ distribution, where $\boldsymbol{\Sigma}$ is the covariance matrix.) Mardia's multivariate skewness statistic is defined as:

$$b_{1,p} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} f_i f_j d_{ij}^3$$

while Mardia's kurtosis is given as:

$$b_{2,p} = \frac{1}{n}\sum_{i=1}^{n} f_i d_{ii}^2$$

Both measures are invariant under the affine (matrix) transformation $AX + D$, and reduce to the univariate measures when p = **NVAR** = 1. Using formulas given in Mardia and Foster (1983), the approximate expected value, asymptotic standard error, and asymptotic p-value for $b_{2,p}$, and the approximate expected value, an asymptotic chi-squared statistic, and p-value for the $b_{1,p}$ statistic are computed. These statistics are all computed under the null hypothesis of a multivariate normal distribution. In addition, standard normal scores $W_1(b_{1,p})$ and $W_2(b_{2,p})$ (different from but similar to the asymptotic normal and chi-squared statistics above) are computed. These scores are combined into an asymptotic chi-squared statistic with two degrees of freedom:

$$S_W = W_1^2\left(b_{1,p}\right) + W_2^2\left(b_{2,p}\right)$$

This chi-squared statistic may be used to test for multivariate normality. A p-value for the chi-squared statistic is also computed.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2MMT/DM2MMT**. The reference is:

   ```
   CALL M2MMT (NOBS, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, ICMPUT, NI, SWT, XMEAN, R,
       LDR, STAT, NRMISS, D, OB, CC)
   ```

   The additional arguments are as follows:

   D — Work vector of length **NVAR**.

   OB — Work vector of length **NVAR**.

   CC — Work vector of length m, where m = **NVAR** * **NVAR** if **ICMPUT** = 1 or m = **NVAR** * **NVAR** * **NVAR** otherwise.

2. Informational errors

Type	Code	Description
4	1	At least one of the variables in X is linearly related to the other variables in X.
4	2	The sum of the frequencies must be greater than the maximum of 3 and the number of variables plus one.

Example

In the following example, 150 observations from a 5 dimensional standard normal distribution are generated via routine **RNNOR** (see Chapter 18, "Random Number Generation"). The skewness and kurtosis statistics are then computed for these observations.

```
      USE IMSL_LIBRARIES
      INTEGER    LDR, LDX, NCOL, NVAR, I
      PARAMETER  (NCOL=5, LDX=150, NVAR=NCOL, LDR=NVAR)
!
      INTEGER    IND(5), NI, NOUT, NRMISS
      REAL       R(LDR,NVAR), STAT(13), SWT, X(LDX,NCOL), XMEAN(NVAR)
!
      DATA IND/1, 2, 3, 4, 5/
!
      CALL RNSET (123457)
      DO 10  I=1, NCOL
      CALL RNNOR(X(:, I))
      10 CONTINUE
!
      CALL MVMMT (X, IND, STAT, NI=NI, SWT=SWT, XMEAN=XMEAN, R=R, &
                  NRMISS=NRMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' NI = ', NI, ' SWT = ', SWT, ' NRMISS = ', NRMISS
      CALL WRRRN ('XMEAN', XMEAN, 1, NVAR, 1)
      CALL WRRRN ('R', R)
      CALL WRRRN ('STAT', STAT, 1, 13, 1)
!
      END
```

Output

```
 NI =   150 SWT =     150.0   NRMISS =   0

                  XMEAN
     1        2        3        4        5
0.0355   0.0467   0.0599   0.0957   0.1007

                     R
        1       2       3       4       5
1   1.033  -0.022  -0.037   0.055  -0.003
2   0.000   0.993  -0.119  -0.076  -0.056
3   0.000   0.000   0.997  -0.089   0.017
```

```
     4   0.000   0.000   0.000   1.008  -0.040
     5   0.000   0.000   0.000   0.000   1.027

                                    STAT
        1       2       3       4       5       6       7       8       9      10
    1.52    1.36   38.71    0.31    0.42   34.21   34.54    1.27   -0.26    0.80

       11      12      13
     0.18    0.21    0.90
```

ADNRM

Performs an Anderson-Darling test for normality.

Required Arguments

X — Array of length **NOBS** containing the observations. (Input)

A — Anderson-Darling statistic. (Output)

P — *p*-value for a test of normality. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than or equal to 3.
Default: **NOBS** = size (**X**).

NMISS — Number of missing observations. (Output)

FORTRAN 90 Interface

Generic: **CALL ADNRM** (**X**, **A**, **P** [, ...])
Specific: The specific interface names are **S_ADNRM** and **D_ADNRM**.

Description

Given a data sample $\{X_i,\ i=1 .. n\}$, where n = **NOBS** and X_i = **X(I)**, routine **ADNRM** computes the Anderson-Darling (AD) normality statistic A and the corresponding p-value P = {probability that a normally distributed n element sample would have an AD statistic $> A$}. If P is sufficiently small (e.g. $P < .05$), then the AD test indicates that the null hypothesis that the data sample is normally-distributed should be rejected. A is calculated as:

$$A = -n - \frac{1}{n}\sum_{i=1}^{n}[(2i-1)\ln(\Phi(Y_i)) + (2n-2i+1)\ln(1-\Phi(Y_i))]$$

where $Y_i = (X_i - \overline{X})/s$ and $\overline{X}$ and s are the sample mean and standard deviation respectively. P is calculated by first transforming A to an "n-adjusted" statistic A*:

$$A^* = A\left(1.0 + \frac{0.75}{n} + \frac{2.25}{n^2}\right)$$

and then calculating P in terms of A^* using a parabolic approximation taken from Table 4.9 in Stephens (1986).

Comments

Informational errors

Type	Code	Description
3	1	The *p*-value has fallen below the minimum value for which its calculation has any accuracy; zero is returned.
4	1	After removing the missing observations only *n* observations remain. The test cannot proceed.

Example

The following example is taken from Conover (1980, pages 364 and 195). The data consists of 50 two digit numbers taken from a telephone book. The AD test fails to reject the null hypothesis of normality at the .05 level of significance.

```
      USE ADNRM_INT
      USE UMACH_INT
      IMPLICIT NONE

      INTEGER, PARAMETER :: NOBS=50
      INTEGER NMISS, NOUT
      REAL P, A, X(NOBS)

      DATA X/ 23, 36, 54, 61, 73, 23, 37, 54, 61, 73, 24, 40, 56, 62,&
          74, 27, 42, 57, 63, 75, 29, 43, 57, 64, 77, 31, 43, 58, 65, &
          81, 32, 44, 58, 66, 87, 33, 45, 58, 68, 89, 33, 48, 58, 68, &
          93, 35, 48, 59, 70, 97/

      CALL ADNRM (X, A, P, NMISS=NMISS)

!                               Write out results
      CALL UMACH(2, NOUT)
      WRITE(NOUT,5) A, P, NMISS
    5 FORMAT(/ ' A     = ', F6.4 / ' P     = ', F6.4 / &
           ' NMISS = ',I3)
      END
```

Output

```
 A     = 0.3339
 P     = 0.5024
 NMISS =   0
```

CVMNRM

Performs a Cramer-von Mises test for normality.

Required Arguments

X — Array of length **NOBS** containing the observations. (Input)

W — Cramer-von Mises statistic. (Output)

P — *p*-value for a test of normality. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
NOBS must be greater than or equal to 3.
Default: **NOBS** = size (**X**).

NMISS — Number of missing observations. (Output)

FORTRAN 90 Interface

Generic: `CALL CVMNRM (X, W, P [, ...])`
Specific: The specific interface names are `S_CVMNRM` and `D_CVMNRM`.

Description

Given a data sample $\{X_i,\ i=1 .. n\}$, where n = **NOBS** and X_i = **X(I)**, routine **CVMNRM** computes the Cramer-von Mises (CvM) normality statistic W and the corresponding p-value P = {probability that a normally distributed n element sample would have a CvM statistic $> W$}. If P is sufficiently small (e.g. $P < .05$), then the CvM test indicates that the null hypothesis that the data sample is normally-distributed should be rejected. W is calculated as:

$$W = \frac{1}{12n} + \sum_{i=1}^{n}\left[\Phi\left(Y_i\right) - \frac{2i-1}{2n}\right]^2$$

where $\Phi(Y_i)$ is the cumulative distribution function of standard normal N(0,1) distribution, $Y_i = (X_i - \overline{X})/s$, and $\overline{X}$ and s are the sample mean and standard deviation respectively. *P* is calculated by first transforming *W* to an "*n*-adjusted" statistic *W**:

$$W* = W\left(1.0 + \frac{0.5}{n}\right)$$

and then calculating *P* in terms of *W** using a parabolic approximation taken from Table 4.9 in Stephens (1986).

Comments

Informational errors

Type	Code	Description
3	1	The *p*-value has fallen below the minimum value for which its calculation has any accuracy; zero is returned.
4	1	After removing the missing observations only *n* observations remain. The test cannot proceed.

Example

The following example is taken from Conover (1980, pages 364 and 195). The data consists of 50 two digit numbers taken from a telephone book. The CvM test fails to reject the null hypothesis of normality at the .05 level of significance.

```
      USE CVMNRM_INT
      USE UMACH_INT
      IMPLICIT NONE

      INTEGER, PARAMETER :: NOBS=50
      INTEGER NMISS, NOUT
      REAL P, W, X(NOBS)

      DATA X/ 23, 36, 54, 61, 73, 23, 37, 54, 61, 73, 24, 40, 56, 62,&
          74, 27, 42, 57, 63, 75, 29, 43, 57, 64, 77, 31, 43, 58, 65,&
          81, 32, 44, 58, 66, 87, 33, 45, 58, 68, 89, 33, 48, 58, 68,&
          93, 35, 48, 59, 70, 97/

      CALL CVMNRM (X, W, P, NMISS=NMISS)

!                                Write out results
      CALL UMACH(2, NOUT)
      WRITE(NOUT,5) W, P, NMISS
```

```
      5 FORMAT(/ ' W     = ', F6.4 / ' P     = ', F6.4 / &
             ' NMISS = ',I3)
        END
```

Output

```
     W     = 0.0520
     P     = 0.4747
     NMISS =   0
```

KSTWO

Performs a Kolmogorov-Smirnov two-sample test.

Required Arguments

X — Vector of length NOBSX containing the observations in sample one. (Input)

Y — Vector of length NOBSY containing the observations in sample two. (Input)

PDIF — Vector of length 6 containing the output statistics. (Output)

I	PDIF(I)
1	D_{mn} = Maximum of the absolute values of D_{mn}^{+} and D_{mn}^{-}.
2	D_{mn}^{+} = Maximum difference between the empirical cumulative distribution function (CDF) of X minus the empirical CDF of Y.
3	D_{mn}^{-} = Maximum difference between the empirical CDF of X minus the empirical CDF of Y. (The maximum of the negative differences.)
4	Z = Standardized value of D_{mn}. A two-sample approximation with no correction for continuity is used.
5	One-sided probability of a larger D_{mn} under the null hypothesis of equal distributions.
6	Two-sided probability of exceeding D_{mn} under the null hypothesis of equal distributions.

Optional Arguments

NOBSX — Number of observations in sample one. (Input)
Default: NOBSX = size (X,1).

NOBSY — Number of observations in sample two. (Input)
Default: NOBSY = size (Y,1).

NMISSX — Number of missing observations in the X sample. (Output)

NMISSY — Number of missing observations in the Y sample. (Output)

FORTRAN 90 Interface

Generic: CALL KSTWO (X, Y, PDIF [, ...])

Specific: The specific interface names are S_KSTWO and D_KSTWO.

FORTRAN 77 Interface

Single: CALL KSTWO (NOBSX, X, NOBSY, Y, PDIF, NMISSX, NMISSY)

Double: The double precision name is DKSTWO.

Description

Routine KSTWO computes Kolmogorov-Smirnov two-sample test statistics for testing that two continuous cumulative distribution functions (CDF's) are identical based upon two random samples. One- or two-sided alternatives are allowed. Exact *p*-values are computed for the two-sided test when NOBSX * NOBSY is less than 104.

Let $F_n(x)$ denote the empirical CDF in the *X* sample, let $G_m(y)$ denote the empirical CDF in the *Y* sample, where n = NOBSX - NMISSX and m = NOBSY - NMISSY, and let the corresponding population distribution functions be denoted by $F(x)$ and $G(y)$, respectively. Then, the hypotheses tested by KSTWO are as follows:

- $H_0: F(x) = G(x) \quad H_1: F(x) \neq G(x)$
- $H_0: F(x) \leq G(x) \quad H_1: F(x) > G(x)$
- $H_0: F(x) \geq G(x) \quad H_1: F(x) < G(x)$

The test statistics are given as follows:

$$D_{mn} = \max\left(D_{mn}^{+}, D_{mn}^{-}\right) \qquad \text{(PDIF(1))}$$

$$D_{mn}^{+} = \max_x(F_n(x) - G_m(x)) \qquad \text{(PDIF(2))}$$

$$D_{mn}^{-} = \max_x(G_m(x) - F_n(x)) \qquad \text{(PDIF(3))}$$

Asymptotically, the distribution of the statistic

$$Z = D_{mn}\sqrt{(mn) / (m+n)}$$

(returned in PDIF(4)) converges to a distribution given by Smirnov (1939).

Exact probabilities for the two-sided test are computed when *nm* is less than or equal to 10^4, according to an algorithm given by Kim and Jennrich (1973), and computed here via function AKS2DF (see Chapter 17, "Probability Distribution Functions and Inverses"). When *nm* is greater than 10^4, the very good approximations given by

Kim and Jennrich are used to obtain the two-sided *p*-values. The one-sided probability is taken as one half the two-sided probability. This is a very good approximation when the *p*-value is small (say, less than 0.10) and not very good for large *p*-values

Comments

Workspace may be explicitly provided, if desired, by use of K2TWO/DK2TWO. The reference is:

```
CALL K2TWO (NOBSX, X, NOBSY, Y, PDIF, NMISSX, NMISSY, XWK, YWK)
```

The additional arguments are as follows:

XWK — Work vector of length NOBSX + 1.

YWK — Work vector of length NOBSY + 1.

Example

The following example illustrates the KSTWO routine with two randomly generated samples from a uniform(0,1) distribution. Since the two theoretical distributions are identical, we would not expect to reject the null hypothesis.

```
      USE RNSET_INT
      USE RNUN_INT
      USE KSTWO_INT
      USE UMACH_INT

      IMPLICIT       NONE
      INTEGER        ISEED, NOBSX, NOBSY, NMISSX, NMISSY, NOUT
      PARAMETER      (ISEED=123457, NOBSX=100, NOBSY=60)
      REAL           X(NOBSX), Y(NOBSY), PDIF(6)
!                                  Generate the sample
      CALL RNSET(ISEED)
      CALL RNUN (X)
      CALL RNUN (Y)
!
      CALL KSTWO (X, Y, PDIF, NMISSX=NMISSX, NMISSY=NMISSY)
!
      CALL UMACH(2, NOUT)
      WRITE(NOUT, 5) PDIF, NMISSX, NMISSY
    5 FORMAT(' D     = ', F8.4 / ' D+    = ', F8.4 / ' D-    = ', F8.4,/ &
        ' Z     = ', F8.4 / ' Prob greater D one sided = ', F8.4 / &
        ' Prob greater D two sided = ', F8.4 / &
        ' Missing X = ', I3 / ' Missing Y = ', I3)
      END
```

Output

```
 D     =   0.1800
 D+    =   0.1800
 D-    =   0.0100
```

```
    Z     =   1.1023
    Prob greater D one sided =   0.0720
    Prob greater D two sided =   0.1440
    Missing X =   0
    Missing Y =   0
```

RUNS

Performs a runs up test.

Required Arguments

X — Vector of length **NRAN** containing the data elements to be added to the test on this invocation. (Input)

COUNT — Vector of length **NRUN** containing the counts of the number of runs up of each length. (Output, if **IDO** = 0 or 1; Input/Output, if **IDO** = 2 or 3)

EXPECT — Vector of length **NRUN** containing the expected number of runs of each length. (Output, if **IDO** = 0 or 3; not referenced otherwise)

COVAR — **NRUN** by **NRUN** matrix containing the variances and covariances of the counts (Output, if **IDO** = 0 or 3; not referenced otherwise)

CHISQ — Chi-squared statistic for testing the null hypothesis of a uniform distribution. (Output, if **IDO** = 0 or 3; not referenced otherwise)

DF — Degrees of freedom for chi-squared. (Output, if **IDO** = 0 or 3; not referenced otherwise)

PROB — Probability of a larger chi-squared. (Output, if **IDO** = 0 or 3; not referenced otherwise)

Optional Arguments

IDO — Processing option. (Input)
Default: **IDO** = 0.

IDO	Action
0	This is the only invocation of RUNS, and all the data are input at once.
1	This is the first invocation of RUNS, and additional calls will be made. Initialization and updating for the NRAN data elements are performed.
2	This is an intermediate invocation of RUNS, and updating for the NRAN data elements is performed.
3	This is the final invocation of RUNS for this data. Updating for the NRAN data elements is performed, followed by the wrap-up computations.

NRAN — Number of data points currently input in **X**. (Input)
NRAN may be positive or zero on any invocation of **RUNS**.
Default: **NRAN** = size (**X**,1).

NRUN — Length of the longest run for which tabulation is desired. (Input)
Runs of length 1, 2, K, **NRUN** - 1 are counted in **COUNT**(1) - **COUNT**(**NRUN** - 1). **COUNT**(**NRUN**) contains the number of runs of length **NRUN** or greater. **NRUN** must be greater than or equal to one.
Default: **NRUN** = size (**COUNT**,1).

LDCOVA — Leading dimension of **COVAR** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOVA** = size (**COVAR**,1).

FORTRAN 90 Interface

Generic: CALL RUNS (X, COUNT, EXPECT, COVAR, CHISQ, DF, PROB [, ...])
Specific: The specific interface names are S_RUNS and D_RUNS.

FORTRAN 77 Interface

Single: CALL RUNS (IDO, NRAN, X, NRUN, COUNT, EXPECT, COVAR, LDCOVA, CHISQ, DF, PROB)
Double: The double precision name is DRUNS.

Description

Routine **RUNS** computes statistics for the runs up test. Runs tests are used to test for cyclical trend in sequences of random numbers. Routine **RUNS** may be called once (**IDO** = 0) or several times (**IDO** = 1, 2, and 3). If all of the data will not fit into memory, the second mode of operation must be used. If the data fit into memory, then the first mode of operation is slightly more efficient. If the runs down test is desired, each observation should first be multiplied by -1 to change its sign, and **RUNS** called with the modified vector of observations.

Routine **RUNS** first tallies the number of runs up (increasing sequences) of each desired length. For *i* = 1, K, *r* - 1, where *r* = **NRUN**, **COUNT**(*i*) contains the number of runs of length *i*. **COUNT**(**NRUN**) contains the number of runs of length **NRUN** or greater. As an example of how runs are counted, the sequence (1, 2, 3, 1) contains 1 run up of length 3, and one run up of length 1.

After tallying the number of runs up of each length, **RUNS** computes the expected values and the covariances of the counts according to methods given by Knuth (1981, pages 65-67). Let *R* denote a vector of length **NRUN** containing the number of runs of each length so that the *i*-th element of *R*, r_i, contains the count of the runs of length *i*. Let Σ_R denote the covariance matrix of *R* under the null hypothesis of randomness, and let μ_R denote the vector of expected values for *R* under this null hypothesis. Then, an approximate chi-squared statistic with **NRUN** degrees of freedom is given as

$$\chi^2 = (R - \mu_R)^T \sum_R^{-1} (R - \mu_R)$$

In general, the larger the value of each element of μ_R, the better the chi-squared approximation.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2NS/DR2NS**. The reference is:

   ```
   CALL R2NS (IDO, NRAN, X, NRUN, COUNT, EXPECT, COVAR, LDCOVA, CHISQ, DF, PROB,
       RWK, CWK, LRUN, NOBS, XLAST)
   ```

   The additional arguments are as follows:

   RWK — Work vector of length **NRUN**.

   CWK — Work vector of length **NRUN * NRUN**.

   LRUN — Scalar used to keep track of number of last runs. (Output, if **IDO** = 0 or 1; input/output, otherwise)
   LRUN should not be changed between calls with the same data set.

   NOBS — Scalar used to keep track of total number of observations. (Output, if **IDO** = 0 or 1; input/output, otherwise)
   NOBS should not be changed between calls with the same data set.

   XLAST — Scalar used to keep track of last run. (Output, if **IDO** = 0 or 1; input/output, otherwise)
   XLAST should not be changed between calls with the same data set.

2. Informational errors

Type	Code	Description
3	1	At least one tie is detected in X.
4	2	The covariance matrix of the runs score is not positive definite. Use a smaller value of NRUN.

Example

The following example illustrates the use of the runs test on 10^4 pseudo-random uniform deviates. In the example, 2000 deviates are generated for each call to **RUNS**. The **IDO** parameter is set to 1 on the first call to **RUNS**, 2 on the second, third, and fourth calls, and 3 on the last call. Since the probability of a larger chi-squared statistic is 0.1872, there is no strong evidence to support rejection of this null hypothesis of randomness.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCOVA, NRAN, NRUN
      PARAMETER  (LDCOVA=6, NRAN=2000, NRUN=6)
!
      INTEGER    I, IDO, NOUT
      REAL       CHISQ, COUNT(NRUN), COVAR(LDCOVA,NRUN), DF, &
                 EXPECT(NRUN), PROB, X(NRAN)
!
      CALL RNSET (123457)
!
      DO 10  I=1, 5
!                                Set IDO
         IF (I .EQ. 1) THEN
            IDO = 1
         ELSE IF (I .EQ. 5) THEN
            IDO = 3
         ELSE
            IDO = 2
         END IF
!                                Generate the random numbers
         CALL RNUN (X)
!
         CALL RUNS (X, COUNT, EXPECT, COVAR, CHISQ, DF, PROB, IDO=IDO)
   10 CONTINUE
!
      CALL WRRRN ('COUNT', COUNT, 1, NRUN, 1)
      CALL WRRRN ('EXPECT', EXPECT, 1, NRUN, 1)
      CALL WRRRN ('COVAR', COVAR)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' CHISQ = ', CHISQ
      WRITE (NOUT,*) ' DF    = ', DF
      WRITE (NOUT,*) ' PROB  = ', PROB
      END
```

Output

```
                    COUNT
       1         2         3         4         5         6
  1709.0    2046.0     953.0     260.0      55.0       4.0

                   EXPECT
       1         2         3         4         5         6
  1667.3    2083.4     916.5     263.8      57.5      11.9

                   COVAR
  1         2         3         4         5         6
  1    1278.2    -194.6    -148.9     -71.6     -22.9      -6.7
  2    -194.6    1410.1    -490.6    -197.2     -55.2     -14.4
  3    -148.9    -490.6     601.4    -117.4     -31.2      -7.8
  4     -71.6    -197.2    -117.4     222.1     -10.8      -2.6
  5     -22.9     -55.2     -31.2     -10.8      54.8      -0.6
  6      -6.7     -14.4      -7.8      -2.6      -0.6      11.7
  CHISQ =      8.76514
  DF    =      6.00000
  PROB  =     0.187225
```

PAIRS

Performs a pairs test.

Required Arguments

X — Vector of length NRAN containing the data elements to be added to the test on this invocation. (Input)

LAG — The lag to be used in computing the pairs statistic. (Input)
Pairs (X(*i*), X(*i* + LAG)) for *i* = 1, K, *N* - LAG are tabulated, where *N* is the total sample size.

COUNT — NCELL by NCELL matrix containing the count of the number of pairs in each cell. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

EXPECT — Expected number of counts in each cell. (Output, if IDO = 0 or 3; not referenced otherwise)

CHISQ — Chi-squared statistic for testing the null hypothesis of a uniform distribution. (Output, if IDO = 0 or 3; not referenced otherwise)

DF — Degrees of freedom for chi-squared. (Output, if IDO = 0 or 3; not referenced otherwise)

PROB — Probability of a larger chi-squared. (Output, if IDO = 0 or 3; not referenced otherwise)

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of PAIRS, and all the data are input at once.
1	This is the first invocation of PAIRS, and additional calls will be made. Initialization and updating for the NRAN data elements are performed.
2	This is an intermediate invocation of PAIRS, and updating for the NRAN data elements is performed.
3	This is the final invocation of PAIRS. Updating for the NRAN data elements is performed, followed by the wrap-up computations.

NRAN — Number of random deviates currently input in X. (Input)
NRAN may be positive or zero on any invocation of PAIRS.
Default: NRAN = size (X,1).

NCELL — Number of equiprobable cells on each axis into which the pairs statistics are to be tabulated. (Input)
Default: NCELL = size (COUNT,1).

LDCOUN — Leading dimension of COUNT exactly as specified in the dimension statement of the calling program. (Input)
Default: LDCOUN = size (COUNT,1).

FORTRAN 90 Interface

Generic: CALL PAIRS (X, LAG, COUNT, EXPECT, CHISQ, DF, PROB [, ...])
Specific: The specific interface names are S_PAIRS and D_PAIRS.

FORTRAN 77 Interface

Single: CALL PAIRS (IDO, NRAN, X, NCELL, LAG, COUNT, LDCOUN, EXPECT, CHISQ, DF, PROB)
Double: The double precision name is DPAIRS.

Description

Routine PAIRS computes the pairs test (or the Good's serial test) on a hypothesized sequence of uniform (0,1) pseudorandom numbers. The test proceeds as follows. Subsequent pairs
(X(*i*), X(*i* + LAG)) are tallied into a $k \times k$ matrix, where k = NCELL. In this tally, element (j, m) of the matrix is incremented, where

$$j = \lfloor kX(i) \rfloor + 1$$
$$m = \lfloor kX(i+l) \rfloor + 1$$

where l = LAG, and the notation $\lfloor\ \rfloor$ represents the greatest integer function, $\lfloor Y \rfloor$ is the greatest integer less than or equal to Y, where Y is a real number. If l = 1, then i = 1, 3, 5, K , n - 1. If l > 1, then i = 1, 2, 3, ..., n - l, where n is the total number of pseudorandom numbers input on the current invocation of PAIRS (*i*.e., n = NRAN).

Given the tally matrix in COUNT, chi-squared is computed as

$$\chi^2 = \sum_{i,j=1}^{k} \frac{(o_{ij} - e)^2}{e}$$

where $e = \Sigma o_{ij}/k^2$, and o_{ij} is the observed count in cell (i, j) (o_{ij} = COUNT(i, j)).

Because pair statistics for the trailing observations are not tallied on any call, the user should call **PAIRS** with **NRAN** as large as possible. For **LAG** < 20 and **NRAN** = 2000, little power is lost.

Comments

Informational errors

Type	Code	Description
3	1	For better efficiency, it is recommended that NRAN be at least twice as large as LAG.
4	2	The sum of the counts is zero. All output statistics are set to NaN (not a number).

Example

The following example illustrates the calculations of the **PAIRS** statistics when a random sample of size 10^4 is used and the **LAG** is 1. The results are not significant. On each call to **PAIRS**, 2000 random deviates are processed. On the first call, initialization is also performed, while on the fifth call the wrap-up computations are performed. Routine **RNUN** (see Chapter 18, "Random Number Generation") is used in obtaining the pseudorandom deviates.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LAG, LDCOUN, NCELL, NOBS
      PARAMETER  (LAG=5, LDCOUN=10, NCELL=10, NOBS=2000)
!
      INTEGER    I, IDO, NOUT
      REAL       CHISQ, COUNT(LDCOUN,NCELL), DF, EXPECT, PROB, X(NOBS)
!
      CALL RNSET (123467)
!
      DO 10  I=1, 5
         CALL RNUN (X)
         IF (I .EQ. 1) THEN
            IDO = 1
         ELSE IF (I .EQ. 5) THEN
            IDO = 3
         ELSE
            IDO = 2
         END IF
         CALL PAIRS (X, LAG, COUNT, EXPECT, CHISQ, DF, PROB, IDO=IDO)
   10 CONTINUE
      CALL UMACH (2, NOUT)
      CALL WRRRN ('COUNT', COUNT)
      WRITE(NOUT,'('' Expect = '', F12.2, /, '' Chi-squared = '', F12.2, &
           '' DF = '', F12.0, /, '' PROBABILITY = '', F12.4)') &
          EXPECT, CHISQ, DF, PROB
      END
```

Output

COUNT

	1	2	3	4	5	6	7	8	9
1	111.0	82.0	95.0	117.0	102.0	102.0	112.0	84.0	90.0
2	104.0	106.0	109.0	108.0	101.0	97.0	102.0	92.0	109.0
3	88.0	111.0	86.0	105.0	112.0	79.0	103.0	105.0	106.0
4	91.0	110.0	108.0	92.0	88.0	108.0	113.0	93.0	105.0
5	104.0	105.0	103.0	104.0	101.0	94.0	96.0	86.0	93.0
6	98.0	104.0	103.0	104.0	79.0	89.0	92.0	104.0	92.0
7	103.0	91.0	97.0	101.0	116.0	83.0	117.0	118.0	106.0
8	105.0	105.0	110.0	91.0	92.0	82.0	100.0	104.0	110.0
9	92.0	102.0	82.0	101.0	93.0	128.0	101.0	109.0	125.0
10	79.0	99.0	103.0	97.0	104.0	101.0	93.0	93.0	98.0

	10
1	73.0
2	88.0
3	99.0
4	114.0
5	103.0
6	99.0
7	99.0
8	89.0
9	98.0
10	105.0

Expect = 99.75

Chi-squared = 104.31 DF = 99.

Probability = 0.3379

DSQAR

Performs a d^2 test.

Required Arguments

X — Vector of length **NRAN** containing the data elements to be added to the test on this invocation. (Input)

COUNT — Vector of length **NCELL** containing the count of the number of d^2 values in each cell. (Output, if **IDO** = 0 or 1. Input/Output, if **IDO** = 2 or 3.)

EXPECT — The expected number of counts in each cell. (Output, if **IDO** = 0 or 3; not referenced otherwise)

CHISQ — Chi-squared statistic for testing the null hypothesis of a uniform distribution. (Output, if **IDO** = 0 or 3; not referenced otherwise)

DF — Degrees of freedom for chi-squared. (Output, if **IDO** = 0 or 3; not referenced otherwise)

PROB — Probability of a larger chi-squared. (Output, if **IDO**= 0 or 3; not referenced otherwise)

Optional Arguments

IDO — Processing Option. (Input)
Default: **IDO** = 0.

IDO	Action
0	This is the only invocation of DSQAR, and all the data are input at once.
1	This is the first invocation of DSQAR, and additional calls will be made. Initialization and updating for the NRAN data elements are performed.
2	This is an intermediate invocation of DSQAR, and updating for the NRAN data elements is performed.
3	This is the final invocation of DSQAR for this data set. Updating for the NRAN data elements is performed, followed by the wrap-up computations.

NRAN — Number of data elements currently input in **X**. (Input)
NRAN may be positive or zero on any invocation of **DSQAR**.
Default: **NRAN** = size (**X**,1).

NCELL — The number of equiprobable cells into which the d^2 statistics are to be tabulated. (Input)
Default: **NCELL** = size (**COUNT**,1).

FORTRAN 90 Interface

Generic: **CALL DSQAR** (**X**, **COUNT**, **EXPECT**, **CHISQ**, **DF**, **PROB** [, ...])
Specific: The specific interface names are **S_DSQAR** and **D_DSQAR**.

FORTRAN 77 Interface

Single: **CALL DSQAR** (**IDO**, **NRAN**, **X**, **NCELL**, **COUNT**, **EXPECT**, **CHISQ**, **DF**, **PROB**)
Double: The double precision name is **DDSQAR**.

Description

Routine **DSQAR** computes the d^2 test for succeeding quadruples of hypothesized pseudorandom uniform (0, 1) deviates. The d^2 test is performed as follows. Let X_1, X_2, X_3, and X_4 denote four pseudorandom uniform deviates, and consider

$$D^2 = (X_3 - X_1)^2 + (X_4 - X_2)^2$$

The probability distribution of D^2 is given as

$$\Pr(D^2 \le d^2) = d^2\pi - \frac{8d^3}{3} + \frac{d^4}{2}$$

when $D^2 \le 1$, where π denotes the value of pi. If $D^2 > 1$, this probability is given as

$$\Pr(D^2 \le d^2) = \frac{1}{3} + (\pi - 2)d^2 + 4\sqrt{d^2 - 1} + 8\frac{\left(d^2 - 1\right)^{\frac{3}{2}}}{3} - \frac{d^4}{2} - 4d^2 \arctan\left(\frac{\sqrt{1 - \frac{1}{d^2}}}{\frac{1}{d}}\right)$$

See Gruenberger and Mark (1951) for a derivation of this distribution.

For each succeeding set of 4 pseudorandom uniform numbers input in **X**, d^2 and the cumulative probability of d^2 ($\Pr(D^2 \le d^2)$) are computed. The resulting probability is tallied into one of k = **NCELL** equally spaced intervals.

Let n denote the number of sets of four random numbers input (n = the total number of observations/4). Then, under the null hypothesis that the numbers input are random uniform (0, 1) numbers, the expected value for each element in **COUNT** is $e = n/k$. An approximate chi-squared statistic is computed as

$$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e)^2}{e}$$

where o_i = **COUNT**(i) is the observed count. Thus, $\mathbf{X}^2$ has k - 1 degrees of freedom, and the null hypothesis of pseudorandom uniform (0, 1) deviates is rejected if $\mathbf{X}^2$ is too large. As n increases, the chi-squared approximation becomes better. A useful generalization is that $e > 5$ yields a good chi-squared approximation.

Comments

Informational errors

Type	Code	Description
3	1	The expected value of a each cell is less than 5. The chi-squared approximation may not be good.
4	2	The sum of the counts is equal to zero. There are no data elements so the chi-squared statistic cannot be computed.

Example

In the following example, 2000 observations generated via routine **RNUN** (see Chapter 18, "Random Number Generation") are input to **DSQAR** in one call. In the example, the null hypothesis of a uniform distribution is not rejected.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    IDO, NCELL, NROW
      PARAMETER  (NCELL=6, NROW=2000)
!
      INTEGER    NOUT
      REAL       CHISQ, COUNT(NCELL), DF, EXPECT, PROB, X(NROW)
!
      CALL RNSET (123457)
!                                Generate the random numbers
      CALL RNUN (X)
!
      CALL DSQAR (X, COUNT, EXPECT, CHISQ, DF, PROB)
!
      CALL WRRRN ('COUNT', COUNT, 1, NCELL, 1)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' EXPECT = ', EXPECT
```

```
          WRITE (NOUT,*) ' CHISQ  = ', CHISQ
          WRITE (NOUT,*) ' DF     = ', DF
          WRITE (NOUT,*) ' PROB   = ', PROB
          END
```

Output

```
                        COUNT
       1       2       3       4       5       6
    87.00   84.00   78.00   76.00   92.00   83.00
    EXPECT =     83.3333
    CHISQ  =     2.056
    DF     =     5.0
    PROB   =    0.841343
```

DCUBE

Performs a triplets test.

Required Arguments

X — Vector of length **NRAN** containing the data elements to be added to the test on this invocation. (Input)

COUNT — **NCELL** by **NCELL** by **NCELL** array containing the tabulations for the triplets test. (Output, if **IDO** = 0 or 1**.** Input/Output, if **IDO** = 2 or 3**.**)

EXPECT — Expected number of counts in each cell. (Output, if **IDO** = 0 or 3; not referenced otherwise)

CHISQ — Chi-squared statistic for testing the null hypothesis of a uniform distribution. (Output, if **IDO** = 0 or 3; not referenced otherwise)

DF — Degrees of freedom for chi-squared. (Output, if **IDO** = 0 or 3; not referenced otherwise)

PROB — Probability of a larger chi-squared. (Output, if **IDO** = 0 or 3; not referenced otherwise)

Optional Arguments

IDO — Processing Option. (Input)
Default: **IDO** = 0.

IDO	Action
0	This is the only invocation of DCUBE, and all the data are input at once.
1	This is the first invocation of DCUBE, and additional calls will be made. Initialization and updating for the NRAN data elements are performed.
2	This is an intermediate invocation of DCUBE, and updating for the NRAN data elements is performed.
3	This is the final invocation of DCUBE for this data set. Updating for the NRAN data elements is performed, followed by the wrap-up computations.

NRAN — Number of random deviates currently input in **X**. (Input)
NRAN may be positive or zero on any invocation of **DCUBE**. **NRAN** must be evenly divisible by 3.
Default: **NRAN** = size (**X**,1).

NCELL — The number of equiprobable cells on each of the three axes into which the triplets are to be tabulated. (Input)
Each set of three data elements is tabulated into a three dimensional cube, each axis of which has **NCELL** cells.
Default: **NCELL** = size (**COUNT**,1).

LDCOUN — Leading and second dimension of matrix **COUNT** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOUN** = size (**COUNT**,1).

FORTRAN 90 Interface

Generic: **CALL DCUBE** (**X**, **COUNT**, **EXPECT**, **CHISQ**, **DF**, **PROB** [, ...])
Specific: The specific interface names are **S_DCUBE** and **D_DCUBE**.

FORTRAN 77 Interface

Single: **CALL DCUBE** (**IDO**, **NRAN**, **X**, **NCELL**, **COUNT**, **LDCOUN**, **EXPECT**, **CHISQ**, **DF**, **PROB**)
Double: The double precision name is **DDCUBE**.

Description

Routine **DCUBE** computes the triplets test on a sequence of hypothesized pseudorandom uniform (0, 1) deviates. The triplets test is computed as follows: Each set of three successive deviates, X_1, X_2, and X_3, is tallied into one of m^3 equal sized cubes, where m = **NCELL**. Let $i = [mX_1] + 1$, $j = [mX_2] + 1$, and $k = [mX_3] + 1$. For the triplet (X_1, X_2, X_3), **COUNT**(i, j, k) is incremented.

Under the null hypothesis of pseudorandom uniform(0, 1) deviates, the m^3 cells are equally probable and each has expected value $e = n/m^3$, where n is the number of triplets tallied. An approximate chi-squared statistic is computed as

$$\chi^2 = \sum_{i,j,k=1}^{k} \frac{\left(o_{ijk} - e\right)^2}{e}$$

where o_{ijk} = **COUNT**(i, j, k).

The computed chi-squared has m^3 **-** 1 degrees of freedom, and the null hypothesis of pseudorandom uniform (0, 1) deviates is rejected if $\mathbf{X}^2$ is too large.

Comments

Informational error

Type	Code	Description
4	1	The sum of the counts is equal to zero. There are no data elements so the chi-squared statistic cannot be computed. CHISQ and PROB are set to NaN (not a number).

Example

In the following example, 2001 deviates generated by IMSL routine **RNUN** (see Chapter 18, "Random Number Generation") are input to **DCUBE**, and tabulated in 27 equally sized cubes. In the example, the null hypothesis is not rejected.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCOUN, NCELL, NRAN
      PARAMETER  (LDCOUN=3, NCELL=3, NRAN=2001)
!
      INTEGER    I, NOUT
      REAL       CHISQ, COUNT(LDCOUN,LDCOUN,NCELL), DF, EXPECT, PROB, &
                 X(NRAN)
!
      CALL RNSET (123457)
!                                Generate the random numbers
      CALL RNUN (X)
!
      CALL DCUBE (X, COUNT, EXPECT, CHISQ, DF, PROB)
!
      DO 10  I=1, NCELL
         CALL WRRRN ('COUNT', COUNT(1:,1:,I))
   10 CONTINUE
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' EXPECT = ', EXPECT
      WRITE (NOUT,*) ' CHISQ  = ', CHISQ
      WRITE (NOUT,*) ' DF     = ', DF
      WRITE (NOUT,*) ' PROB   = ', PROB
      END
```

Output

```
            COUNT
              1       2       3
1     26.00   27.00   24.00
2     20.00   17.00   32.00
3     30.00   18.00   21.00

            COUNT
              1       2       3
1     20.00   16.00   26.00
```

```
    2   22.00   22.00   27.00
    3   30.00   24.00   26.00

                COUNT
             1       2       3
    1   28.00   30.00   22.00
    2   23.00   24.00   22.00
    3   33.00   30.00   27.00
    EXPECT =     24.7037
    CHISQ  =     21.7631
    DF     =     26.0000
    PROB   =    0.701586
```

Time Series Analysis and Forecasting

Routines

Usage Notes

The name of a time series routine is a combination of three to four sets of one to four letters. The first set specifies the type of model or method. The second set identifies the particular approach. If the name uses four sets of letters, then both the second and third sets are used to identify the particular approach. The final set always specifies the general procedure. The table below summarizes the naming convention of the time series analysis and forecasting routines.

The names and meanings of arguments are consistent within a set of routines pertaining to a particular topic. For example, **XCNTR** corresponds to the constant used to center the time series **X** in all of the spectral analysis routines. Note that **IPRINT** always represents the printing option, the values and possible choices of output necessarily depend on the given routine. An option argument always begins with the letter "**I**," and a leading dimension argument always begins with "**LD**."

The routines in this chapter assume the time series does not contain any missing observations. If missing values are present, they should be set to NaN (see the Reference Material section for the routine **AMACH**), and the routine will return an appropriate error message. To enable fitting of the model, the missing values must be replaced by appropriate estimates.

Meaning	Abbreviation
Bayesian	BAY*
Nonseasonal ARMA	NS*
Transfer Function	TF*
Maximum Likelihood	MAX*
Multichannel	M*
Periodogram	P*
Cross Periodogram	CP*
Spectral Density	S*
Automatic Model Selection	AUTO*
Cross-Spectral Density	CS*
Preliminary	*P*
Univariate	*UNI*
Multivariate	*MUL*
Final Prediction Error	*FPE*
Method of Moments	*MM*

Meaning	Abbreviation
Least-Squares	*LS*
Box-Jenkins	*BJ*
Spectral Window	*SW*
Weights	*WE*
Autoregressive	*AR
Autoregressive, Moving Average	*ARMA
Seasonal Modeling	*SEA
Estimation	*E
Forecast	*F
Fast Fourier Transform	*FFT
Periodogram	*P
Data	*D

The "*" represents one or more letters.

General Methodology

A major component of the model identification step concerns determining if a given time series is stationary. The sample correlation functions computed by routines ACF, PACF, CCF, and MCCF may be used to diagnose the presence of nonstationarity in the data, as well as to indicate the type of transformation require to induce stationarity. The family of power transformations provided by routine BCTR coupled with the ability to difference the transformed data using routine DIFF affords a convenient method of transforming a wide class of nonstationary time series to stationarity.

The "raw" data, transformed data, and sample correlation functions also provide insight into the nature of the underlying model. Typically, this information is displayed in graphical form via time series plots, plots of the lagged data, and various correlation function plots. The routines in Chapter 16, "Line Printer Graphics" provide the necessary tools to produce the visual displays of this quantitative information.

The observed time series may also be compared with time series generated from various theoretical models to help identify possible candidates for model fitting. The routine RNARM in Chapter 18, "Random Number Generation" may be used to generate a time series according to a specified autoregressive moving average model.

Time Domain Methodology

Once the data are transformed to stationarity, a tentative model in the time domain is often proposed and parameter estimation, diagnostic checking and forecasting are performed.

Autoregressive Moving Average Model

A parsimonious, yet comprehensive, class of stationary time series models consists of the nonseasonal autoregressive moving average (**ARMA**) processes defined by

$$\phi(B)(W_t - \mu) = \theta(B)A_t \qquad t \in ZZ$$

where

$$ZZ = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

denotes the set of integers, B is the backward shift operator defined by $B^k W_t = W_{t-k}$, μ is the mean of W_t,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p \qquad p \geq 0$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q \qquad q \geq 0$$

The model is of order (p, q) and is referred to as an **ARMA**(p, q) model.

An equivalent version of the **ARMA**(p, q) model is given by

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in ZZ$$

where θ_0 is an overall constant defined by

$$\theta_0 = \mu\left(1 - \sum_{i=1}^{p}\phi_i\right)$$

See Box and Jenkins (1976, pages 92–93) for a discussion of the meaning and usefulness of the overall constant. The coefficients in the ARMA model can be estimated using MAX_ARMA.

Parameter estimates for ARMA processes can also be obtained using the **MAX_ARMA** routine. This routine uses the maximum likelihood method to obtain estimates for the moving average and autoregressive parameters in an ARMA model. This routine also requires initial parameter estimates and further requires that these initial values represent a stationary time series. If they are not stationary, **MAX_ARMA** replaces these estimates with initial estimates that are stationary. However these may be far away from the values selected to initially describe this series.

Moreover, the method of maximum likelihood for estimating ARMA parameters may not converge to stationary estimates. In this case, **MAX_ARMA** will display a warning message and sets its convergence parameter **ICONV** to zero.

If the "raw" data $\{Z_t\}$ are homogeneous nonstationary, then differencing induces stationarity and the model is called autoregressive *integrated* moving average (ARIMA). Parameter estimation is performed on the stationary time series

$$W_t = \nabla^d Z_t$$

where

$$\nabla^d = (1 - B)^d$$

is the backward difference operator with period 1 and order d, $d > 0$.

Typically, routine NSPE is first applied to the transformed data to provide preliminary parameter estimates. These estimates are used as initial values in an estimation procedure. In particular, routine NSLSE may be used to compute conditional or unconditional least-squares estimates of the parameters, depending on the choice of the backcasting length. Parameter estimates from either NSPE or NSLSE may be input to routine NSBJF to produce forecasts with associated probability limits. The routines for preliminary parameter estimation, least squares parameter estimation, and forecasting follow the approach of Box and Jenkins (1976, programs 2–4, pages 498–509).

Regression in Autoregressive Integrated Moving Average

There may be one or more external time series that relate to the time series of interest, which may be useful in improving forecasts. Routine REG_ARIMA allows for the inclusion of one or more regression time series in the above ARIMA model. That is, if there are r time series $\{X_{i,t}, i = 1, \ldots, r\}$ associated with a times series Y_t, the regression ARIMA model (integrated of order d) is

$$W_t = \nabla^d Z_t$$

where

$$Z_t = \left(Y_t - \sum_{t=1}^{r} \beta_i X_{i,t} \right)$$

That is, Z_t is the residual (indexed by t) of the regression of Y_t on $\{X_{i,t}, i = 1, \ldots, r\}$.

Transfer Function Model

Define $\{x_t\}$ and $\{y_t\}$ by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and undifferenced output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and differenced output series may be obtained using the routine DIFF following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \delta^{-1}(B)\omega(B)X_{t-b} + n_t$$

or equivalently,

$$y_t = \delta^{-1}(B)\omega(B)x_{t-b} + n_t$$

where $n_t = \nabla^d N_t$ for $d \geq 0$, and the left-hand side and right-hand side transfer function polynomial operators are, respectively,

$$\delta(B) = 1 - \delta_1 B - \delta_2 B^2 - \ldots - \delta_r B^r$$

$$\omega(B) = \omega_0 - \omega_1 B - \omega_2 B^2 - \ldots - \omega_s B^s$$

with $r \geq 0$, $s \geq 0$, and $b \geq 0$. The noise process $\{N_t\}$ and the input process $\{X_t\}$ are assumed to be independent, with the noise process given by the ARIMA model

$$\phi(B)n_t = \theta(B)A_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

with $p \geq 0$ and $q \geq 0$.

The impulse response weights $\{v_k\}$ of the transfer function

$$\nu(B) = \delta^{-1}(B)\omega(B) = \nu_0 + \nu_1 B + \nu_2 B^2 + \ldots$$

and the differenced noise series $\{n_t\}$ are estimated using routine IRNSE. Preliminary estimates of the transfer function parameters and noise model parameters are computed by routine TFPE.

Multivariate Autoregressive Time Series

A multivariate autoregressive time series can be expressed as:

$$X_t = \sum_{i=1}^{p} A_i \cdot X_{t-i} + U_t$$

where

$$X_t = (x_{1,t},\ x_{2,t},\ \ \ldots\ ,\ x_{m,t})'$$

is a column vector containing the values for the *m* univariate time series at time = *t*.

$$A_1,\ A_2,\ \ \ldots\ ,\ A_p$$

are the *m* x *m* matrices containing the autoregressive parameters for lags 1, 2, ..., *p*.

$$U_t = (\varepsilon_{1,t},\ \varepsilon_{2,t},\ \ \ldots\ ,\ \varepsilon_{m,t})'$$

is a column vector containing the values for the *m* white noise values for each time series at time = *t*.

Akaike's Information Criterion (AIC) can be used to identify the optimum number of lags. For a multivariate autoregressive time series,

$$AIC = (N - p) \cdot \ln(\left\|\hat{\Sigma}_p\right\|) + 2 \cdot K + \left(N - p\right)\left(\ln\left(2\pi\right) + 1\right)$$

where

N = number of observations in each of the *m* univariate time series,

K = number of non-zero autoregressive coefficients in the parameter matrices,

p = maximum number of lags used in calculating AIC, and

$\left\|\hat{\Sigma}_p\right\|$ = determinate of the estimated *m* by *m* covariance matrix for the white noise, U_t.

Routine **AUTO_MUL_AR** calculates AIC for selected lags and returns parameter estimates for the optimum lag.

Multichannel Time Series

A multichannel time series *X* is simply a multivariate time series whose channels correspond to interrelated univariate time series. In this setting, the model-building process is a logical extension of the procedures used to identify, estimate, and forecast univariate time series. In particular, the multichannel cross-correlation function computed by routine **MCCF** may help identify a tentative model. A particular regression model may be fit using routine **MLSE**, with the Wiener filter estimated using routine **MWFE**. The Wiener forecast function for a single channel may be obtained by routine **SPWF**. The state space approach to fitting many time domain models is available through routine **KALMN**.

Automatic Model Selection

There are two popular criteria for comparing autoregressive (AR) models with different lags:

- FPE – Final Prediction Error
- AIC – Akaike's Information Criterion.

These are defined for both univariate and multivariate time series. FPE for an autoregressive univariate model with p lags is calculated using the formula:

$$FPE_p = \frac{(N+p+1)}{(N-p-1)}\,\hat{\sigma}_p^2$$

where

N = number of observations in the time series, and $\hat{\sigma}_p^2$ = the estimate for the variance of the white noise term in an AR model of order *p*. The fit with the smallest FPE is considered best.

Similarly, AIC for an AR univariate series is calculated by:

$$AIC = -2 \cdot \ln(L) + 2p$$

where

L = the value for the maximum likelihood function for the fitted model, and

p = number of lags for the AR model. In some routines this is approximated by

$$AIC = (N-p) \cdot \ln(\hat{\sigma}_p^2) + 2 \cdot K + \left(N-p\right)\left(\ln\left(2\pi\right)+1\right)$$

where *K*= number of nonzero parameters in the model.

Similar to FPE, the fit with the smallest AIC is considered best.

A formula also exist for the final prediction error of a multivariate time series:

$$MFPE_p = \frac{\left(1+\frac{pm+1}{N}\right)^m}{\left(1-\frac{pm+1}{N}\right)^m}\,\left\|\hat{\Sigma}_p\right\|$$

where *m* = number of univariate time series, and *p* = maximum number of lags in the AR model. The equivalent multivariate AIC calculation is

$$AIC = (N - p) \cdot \ln(\left\|\hat{\Sigma}_p\right\|) + 2 \cdot K + \left(N - p\right)\left(\ln\left(2\pi\right) + 1\right)$$

The routine **AUTO_PARM** uses a third criterion, called "Minimum Description Length" or MDL, to automatically fit piecewise AR models to a time series with structural breaks (i.e., a potentially non-stationary time series having stationary segments).

The MDL is defined as

$$MDL(m,\tau_1, \ldots ,\tau_m,p_1, \ldots ,p_{m+1}) = \ln\, m + (m-1)\ln\, n + \sum_{j=1}^{m+1} \frac{2+p_j}{2} \ln\, n_j - \ln\, L,$$

where m is the number of structural breaks in the series, $\{\tau_1, \tau_2, \ldots, \tau_{m+1}$ are the locations of the breaks, n_j is the number of observations in the j-th segment, p_j is the order of the AR model fit to the j-th segment, and L is the combined likelihood over all segments. **AUTO_PARM** also allows the choice to use the AIC criterion,

$$AIC(m,\tau_1, \ldots , \tau_m,p_1, \ldots , p_{m+1}) = 2(\text{number of parameters}) - 2\ln\, L = 2(1 + m + \sum_{j=1}^{m+1}(2 + p_j)) - 2\ln\, L.$$

The table below summarizes the five routines for automatic AR fitting.

Routine	Variables	Criterion
AUTO_UNI_AR	Univariate	AIC
AUTO_MUL_AR	Multivariate	AIC
AUTO_FPE_UNI_AR	Univariate	FPE
AUTO_FPE_MUL_AR	Multivariate	FPE
AUTO_PARM	Univariate	MDL/AIC

Frequency Domain Methodology

An alternative method of time series analysis with much less emphasis on the form of the model may be performed in the frequency domain.

Spectral Analysis

Let $\{X(t)\}$ denote a continuous-parameter stationary process with mean

$$\mu = E[X(t)]$$

and autocovariance function

$$\sigma(k) = \text{cov}\{X(t), X(t+k)\} = \text{E}\{[X(t) - \mu][X(t+k) - \mu]\} \qquad k \in \mathbf{R}$$

Similarly, let $\{X_t\}$ denote a discrete-parameter stationary process with mean

$$\mu = E[X_t]$$

and autocovariance function

$$\sigma(k) = \text{cov}\{X_t, X_{t+k}\} = \text{E}\{[X_t - \mu][X_{t+k} - \mu]\} \qquad k \in ZZ$$

Note that $\sigma(0) = \sigma^2$ is the variance of the process.

The routines for the spectral analysis of time series are concerned with the estimation of the spectral density of a stationary process given a finite realization $\{X_t\}$ for $t = 1, \ldots, n$ where n = **NOBS**. This realization consists of values sampled at equally spaced time intervals in the continuous-parameter case or of values observed consecutively in the discrete-parameter case. Hence, we need only develop methodology concerned with the spectral analysis of discrete-parameter stationary processes and later account for the time sampling in the continuous-parameter model.

The nonnormalized spectral density $h(\omega)$ and the autocovariance function $\sigma(k)$ of the stationary process form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$h(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \sigma(k)e^{-i\omega k}dk$$
$$\sigma(k) = \int_{-\pi}^{\pi} h(\omega)e^{i\omega k}d\omega$$

Similarly, the normalized spectral density $f(\omega)$ and the autocorrelation function $\rho(k) = \sigma(k)/\sigma(0)$ of the stationary process form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$\begin{aligned} f(\omega) &= \frac{h(\omega)}{\sigma(0)} \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \rho(k)e^{-i\omega k}dk \\ \rho(k) &= \int_{-\pi}^{\pi} f(\omega)e^{i\omega k}d\omega \end{aligned}$$

The discrete-parameter analogs of the above equations involve summation over k instead of integration over dk. Also, the normalized spectral density $f(\omega)$ satisfies

$$\int_{-\pi}^{\pi} f(\omega)d\omega = 1$$

Discrete Fourier Transform. The discrete Fourier transform of the sequence $\{Z_t\}$ for $t = 1, \ldots, N$ is defined by

$$\zeta\left(\omega_p\right) = \sum_{t=1}^{N} Z_t e^{-i\omega_p t}$$

over the discrete set of frequencies

$$\omega_p = \frac{2\pi p}{N} \quad p = 0, \pm 1, \ \ldots, \ \pm \lfloor N/2 \rfloor$$

where the function $\lfloor r \rfloor$ determines the greatest integer less than or equal to *r*. An alternative representation of $\zeta(\omega_p)$ in terms of cosine and sine transforms is

$$\zeta(\omega_p) = \alpha(\omega_p) - i\beta(\omega_p)$$

where

$$\alpha(\omega_p) = \sum_{t=1}^{N} Z_t \cos(\omega_p t)$$
$$\beta\left(\omega_p\right) = \sum_{t=1}^{N} Z_t \sin(\omega_p t)$$

The fast Fourier transform algorithm implemented in the IMSL MATH/LIBRARY routine **FFTCF** is used to compute the discrete Fourier transform. All of the frequency domain routines that output a periodogram utilize the fast Fourier transform algorithm.

Centering and Padding. Consider the centered and padded realization

for $t = 1, \ldots, N$ defined by

$$\widetilde{X}_t = \begin{cases} X_t - \hat{\mu} & t = 1, \ \ldots, \ n \\ 0 & t = (n+1), \ \ldots, \ N \end{cases} \qquad (1)$$

where $N = (n + n_0)$ and

$$\hat{\mu} = XCNTR$$

is

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu \text{ unknown} \end{cases} \qquad (2)$$

Centering the data simplifies the formulas for estimation of the periodogram and spectral density. The addition of n_0 = **NPAD** zeros to the end of the data is called *padding*. This procedure increases the effective length of the data from *n* to *N* in an effort to:

- increase the computational efficiency of the Fourier transformation of the series by providing a more suitable series length *N* (Priestley 1981, page 577).
- obtain the periodogram ordinates required to give the exact expression of the sample autocovariances in terms of the inverse Fourier transformation of the periodogram (Priestley 1981, page 579).
- produce periodogram ordinates over a more refined range of frequencies ω_p.

Any desired filtering, prewhitening, or data tapering should be performed prior to estimating the spectral density. The resulting estimate may be adjusted accordingly.

Periodogram. The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the centered and padded sequence

is defined by

$$I_{n,N,\tilde{X}}(\omega_p) = K\left|\sum_{t=1}^{N}\tilde{X}_t e^{-i\omega_p t}\right|^2 = K\left|\zeta_{\tilde{X}}(\omega_p)\right|^2$$

where *K* is the scale factor

$$K = \begin{cases} \frac{2}{n} & \text{for the usual periodogram} \\ \frac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The scale factor of the usual periodogram relates the ordinates to the sum of squares of

$$X_t - \hat{\mu}$$

(Fuller 1976, pages 276–277). If the first ordinate (corresponding to $p = 0$) is replaced by one-half of its value, then if *N* is odd, the sum of the $\lfloor N/2 \rfloor + 1$ ordinates corresponding to $p = 0, 1, \ldots, \lfloor N/2 \rfloor$ is

$$\frac{N}{n}\sum_{t=1}^{n}(X_t - \hat{\mu})^2$$

The modified periodogram is an asymptotically unbiased estimate of the nonnormalized spectral density function at each frequency ω_p (Priestley 1981, page 417). The argument `IPVER` is used to specify the version of the periodogram.

Spectral Density. The relationship between the sample autocovariance function and estimate of the nonnormalized spectral density function is similar to the theoretical situation previously discussed.

Define the sample autocovariance function of the X_t process by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-|k|}\left\{[X_t - \hat{\mu}][X_{t+|k|} - \hat{\mu}]\right\} \quad k = 0, \pm 1, \ \ldots, \ \pm(n-1)$$

where

$$\hat{\mu}$$

is given by Equation 2. Note that

$$\hat{\sigma}(0) = \hat{\sigma}^2$$

is the sample variance. The nonnormalized spectral density may be estimated directly from the sample autocovariances by

$$\hat{h}(\omega) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)\hat{\sigma}(k)e^{-i\omega k}$$

The sequence of weights $\{\boldsymbol{\lambda}_n(k)\}$ called the *lag window* decreases at a rate appropriate for consistent estimation of $h(\boldsymbol{\omega})$.

An algebraically equivalent method of estimating $h(\boldsymbol{\omega})$ consists of locally smoothing the modified periodogram in a neighborhood of $\boldsymbol{\omega}$. Let

$$I_{n,N,\tilde{X}}$$

denote the modified periodogram of the centered and padded realization

$$\left\{\tilde{X}_t\right\}$$

defined in Equation 1. Then, an estimate of the nonnormalized spectral density is given by

$$\hat{h}(\omega) = \frac{2\pi}{N}\sum_{p=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}}(\omega_p)W_n(\omega - \omega_p) \quad (3)$$

where

$$W_n(\theta) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)e^{-i\theta k}$$

The *spectral window* $W_n(\theta)$ is the discrete Fourier transform of the lag window $\lambda_n(k)$. We note that for $N = 2n - 1$, the modified periodogram and autocovariances,

$$I_{n,2n-1,\tilde{X}}(\omega_p) \text{ and } \hat{\sigma}(k)$$

form the discrete Fourier transform pair

$$I_{n,N,\tilde{X}}(\omega_p) = \frac{1}{2\pi} \sum_{k=-(n-1)}^{n-1} \hat{\sigma}(k) e^{-i\omega_p k}, \quad p = 0, \pm 1, \ \ldots, \ \pm(n-1)$$

$$\hat{\sigma}(k) = \frac{2\pi}{N} \sum_{p=-(n-1)}^{n-1} I_{n,N,\tilde{X}}(\omega_p) e^{i\omega_p k}, \quad k = 0, \pm 1, \ \ldots, \ \pm(n-1)$$

This relationship is exact and recovers the $(n - 1)$ sample autocovariances only when $n_0 = (n - 1)$ zeros are padded, since then $\lfloor N/2 \rfloor = (n - 1)$.

Another method of estimating $h(\omega)$ is given by

$$\hat{h}(\omega) = \sum_j w_j I_{n,N,\tilde{X}}(\omega_{p,j}) \qquad (4)$$

where

$$\omega_{p,j} = \frac{2\pi\{p(\omega) + j\}}{N}$$

and $p(\omega)$ is the integer such that $\omega_{p,0}$ is closest to ω. The sequence of m weights $\{w_j\}$ for $j = -[m/2], \ldots, (m - [m/2] - 1)$ is fixed in the sense that they do not depend on the frequency, ω, and satisfy $\Sigma_j w_j = 1$. Priestley (1981, page 581) notes that if we write

$$w_j = \frac{2\pi}{N} W_n(\omega - \omega_{p,j})$$

then Equation 4 and Equation 3 are quite similar except that the weights $\{w_j\}$ depend on ω. In fact, if $p(\omega) = 0$ and $m = N$, these equations are equivalent.

Given estimates

$$\hat{h}(\omega) \text{ and } \hat{\sigma}(0)$$

the estimate of the normalized spectral density is given by

$$\hat{f}(\omega) = \frac{\hat{h}(\omega)}{\hat{\sigma}(0)}$$

This follows directly from the definition of $f(\omega)$.

Spectral Window. The following spectral windows $W_n(\theta)$ are available in routines containing the argument `ISWVER`.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right)$$

for $0 < a \le 0.25$, where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hanning window is obtained when $a = 0.23$, and the Tukey-Hamming window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2\left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \leq \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

The *window parameter M* is inversely proportional to the *bandwidth* of the spectral window. Priestley (1981, pages 520–522) discusses a number of definitions of bandwidth and concludes that the particular definition adopted is of little significance. The choice of spectral window bandwidth, and hence, the choice of *M*, is a more important problem. One practical choice for *M* is the last lag at which the estimated autocorrelation function

$$\hat{\rho}(k)$$

is significantly different from zero, i.e.,

$$\hat{\rho}(k) \approx 0 \text{ for } k > M$$

The estimated autocorrelations and their associated estimated standard errors can be computed using routine ACF. See Priestley (1981, pages 528–556) for alternative strategies of determining the window parameter *M*.

Since the spectral window is the Fourier transform of the lag window, we estimate the spectral density function by application of a particular spectral window to the periodogram. Note that *M* is directly related to the rate of decay of the lag window.

Time Interval. Consider the continuous-parameter stationary process {*X*(*t*)} and let {X_t} denote a realization of this process sampled at equal time intervals **Δ***t* = **TINT**. Although the spectral density of *X*(*t*) extends over the frequency range (**-π**, **π**), the spectral density of X_t is unique over the restricted frequency range (**-π**/**Δ***t*, **π**/**Δ***t*). This problem of aliasing or *spectrum folding* is inherent to spectral analysis, see Blackman and Tukey (1958) and Priestley (1981) for further discussion.

In practice, the {X_t} realization is treated as a discrete parameter process with spectral density

$$h_X^{\dagger}(\omega)$$

defined over the frequency range (**-π**, **π**). This corresponds to setting **Δ***t* = 1. The transformation of the spectral density to the restricted frequency range (**-π**/**Δ***t*, **π**/**Δ***t*) is given by

$$h_X(\omega) = \Delta t\, h_X^{\dagger}(\omega \Delta t) \quad |\omega| \leq \pi / \Delta t$$

Priestley (1981, pages 507–508) considers a method of choosing **Δ***t*. A similar transformation is performed for the estimated spectral density.

Frequency Scale. The argument **IFSCAL** is used to specify the scale of the frequencies at which to estimate the spectral density. The **NF** frequencies are contained in the argument **F**.

Approximate Confidence Intervals for Spectral Ordinates. An approximate (1 - α)100% confidence interval for the value of the nonnormalized spectral density function $h(\omega)$ at a particular frequency ω is given by the formula (Priestley 1981, page 468)

$$\left(\frac{\mathrm{DF} \times \hat{h}(\omega)}{\chi^2_{\mathrm{DF},1-\alpha/2}}, \frac{\mathrm{DF} \times \hat{h}(\omega)}{\chi^2_{\mathrm{DF},\alpha/2}} \right)$$

Routine CHIIN using argument P = 1 - α/2 and P = α/2 can be used to compute the percentage point

$$\chi^2_{\mathrm{DF},P}$$

Also, routine CHIIN should be used with degrees of freedom (DF), which depend upon the version of the spectral window (ISWVER), as given in the following table (Priestley 1981, page 467).

ISWVER	Window	DF
1	Modified Bartlett	3*n*/*M*
2	Daniell	2*n*/*M*
3	Tukey-Hamming	2.5164*n*/*M*
4	Tukey-Hanning	2 2/3*n*/*M*
5	Parzen	3.708614*n*/*M*
6	Bartlett-Priestley	1.4*n*/*M*

If one of the windows above is not specified and the user provides relative weights, such as with routine SWED, the weights are normalized to sum to one in the actual computations. Given all *m* (*m* odd) normalized weights w_j, then for $2\pi\lfloor m/2 \rfloor/n < \omega < \pi(1 - 2\lfloor m/2 \rfloor/n)$ the degrees of freedom for a confidence interval on $h(\omega)$ are given by Fuller (1976, page 296)

$$\mathrm{DF} = \frac{2}{\sum_{j=-\lfloor m/2 \rfloor}^{\lfloor m/2 \rfloor} w_j^2}$$

Frequently, confidence intervals on the ln $h(\omega)$ are suggested because this produces fixed width intervals. The interval is

$$\left(\ln\hat{h}(\omega) + \ln\left[\frac{\mathrm{DF}}{\chi^2_{\mathrm{DF},1-\alpha/2}} \right], \ln\hat{h}(\omega) + \ln\left[\frac{\mathrm{DF}}{\chi^2_{\mathrm{DF},a/2}} \right] \right)$$

Cross-Spectral Analysis

The routines for cross-spectral analysis are concerned with the estimation of the crossspectral density of two jointly stationary processes given finite realizations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, \ldots, n$. These realizations consist of values sampled at equally spaced time intervals in the continuous-parameter case or of values observed consecutively in the discrete-parameter case. Again, we develop methodology concerned with the cross-spectral analysis of discrete-parameter stationary processes and later account for the time sampling in the continuous-parameter model.

Let μ_X and $\sigma_{XX}(k)$ denote the mean and autocovariance function of the X_t process; similarly, define μ_Y and $\sigma_{YY}(k)$, with respect to the Y_t process. Define the cross-covariance function between X_t and Y_t by

$$\sigma_{XY}(k) = \mathrm{cov}\{[X_t - \mu_X][Y_{t+k} - \mu_Y]\} \qquad k \in ZZ$$

Then, the nonnormalized cross-spectral density $h_{XY}(\omega)$ and the cross-covariance function $\sigma_{XY}(k)$ form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$h_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\sigma_{XY}(k)e^{-i\omega k}dk$$
$$\sigma_{XY}(k) = \int_{-\pi}^{\pi}h_{XY}(\omega)e^{i\omega k}d\omega$$

Similarly, the normalized cross-spectral density $f_{XY}(\omega)$ and the cross-correlation function $\rho_{XY}(k) = \sigma_{XY}(k)/[\sigma_{XX}(0)\sigma_{YY}(0)]$ form a Fourier transform pair. The relationship in the continuous-parameter case is given by

$$f_{XY}(\omega) = \frac{h_{XY}(\omega)}{\sigma_{XX}(0)\sigma_{YY}(0)} = \frac{1}{2\pi}\int_{-\infty}^{\infty}\rho_{XY}(k)e^{-i\omega k}dk$$
$$\rho_{XY}(k) = \int_{-\pi}^{\pi}f_{XY}(\omega)e^{i\omega k}d\omega$$

The discrete-parameter analogs of the above equations involve summation over k instead of integration over dk.

The cross-spectral density function is often written in terms of real and imaginary components, since in general, the function is complex-valued. In particular,

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

where the *cospectrum* and *quadrature spectrum* of the X_t and Y_t process are respectively defined by

$$c_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{1}{2}[\sigma_{XY}(k) + \sigma_{XY}(-k)]\cos k\omega\, dk$$
$$q_{XY}(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{1}{2}[\sigma_{XY}(k) - \sigma_{XY}(-k)]\sin k\omega\, dk$$

The *polar* form of $h_{XY}(\omega)$ is defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\varphi_{XY}(\omega)}$$

where the *cross-amplitude spectrum* is

$$\alpha_{XY}(\omega) = \left| h_{XY}(\omega) \right| = \left[c_{XY}^2(\omega) + q_{XY}^2(\omega) \right]^{1/2}$$

and the *phase spectrum* is

$$\varphi_{XY}(\omega) = \tan^{-1}[-q_{XY}(\omega)/c_{XY}(\omega)]$$

The *coherency spectrum* is defined by

$$w_{XY}(\omega) = \frac{h_{XY}(\omega)}{[h_{XX}(\omega)h_{YY}(\omega)]^{1/2}}$$

For a given frequency **ω**, the *coherency* $|w_{XY}(\omega)|$ lies between zero and one, inclusive, and reflects the linear relationship between the random coefficients. See Priestley (1981, pages 654–661) for additional information concerning the interpretation of the components of the cross-spectral density.

Centering and Padding. The centered and padded realizations

$$\left\{ \tilde{X}_t \right\} \text{ and } \left\{ \tilde{Y}_t \right\}$$

are defined as in Equation 1 with centering constants

$$\hat{\mu}_X = \text{XCNTR and } \hat{\mu}_Y = \text{YCNTR}$$

Any desired filtering, prewhitening, or data tapering should be performed prior to estimating the crossspectral density. The resulting estimate may be adjusted accordingly.

Cross Periodogram. The cross periodogram of the sample sequences $\{X_t\}$ and $\{Y_t\}$, $t = 1, \ldots, n$ computed with the padded sequences

$$\left\{ \tilde{X}_t \right\} \text{ and } \left\{ \tilde{Y}_t \right\}$$

$t = 1, \ldots, N$ is defined by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_p) = \left(\sum_{t=1}^{N} \tilde{X}_t e^{-i\omega_p t} \right) \left(\sum_{t=1}^{N} \tilde{Y}_t e^{i\omega_p t} \right) = K\zeta_{\tilde{X}}(\omega_p)\zeta_{\tilde{Y}}^*(\omega_p)$$

where K is the scale factor

$$K = \begin{cases} \frac{2}{n} & \text{for the usual cross periodogram} \\ \frac{1}{2\pi n} & \text{for the modified cross periodogram} \end{cases}$$

The scale factor option is maintained for compatibility with the spectral routines. The argument **IPVER** is used to specify the version of the periodogram used to compute the cross periodogram.

Cross-Spectral Density Estimation. The relationship between the sample cross-covariance function and estimate of the nonnormalized cross-spectral density function is similar to the theoretical situation previously discussed.

Define the sample cross-covariance function between the X_t and Y_t process by

$$\tilde{\sigma}_{XY}(k) = \begin{cases} \frac{1}{n}\sum_{t=1}^{n-k}\left\{\left[X_t - \hat{\mu}_X\right]\left[Y_{t+k} - \hat{\mu}_Y\right]\right\} & k = 0,\ 1,\ \ldots,\ (n-1) \\ \frac{1}{n}\sum_{t=1-k}^{n}\left\{\left[X_t - \hat{\mu}_X\right]\left[Y_{t+k} - \hat{\mu}_Y\right]\right\} & k = -1,\ -2,\ \ldots,\ -(n-1) \end{cases}$$

The nonnormalized cross-spectral density may be estimated directly from the sample cross-covariances by

$$\tilde{h}_{XY}(\omega) = \frac{1}{2\pi}\sum_{k=-(n-1)}^{(n-1)} \lambda_n(k)\hat{\sigma}_{XY}(k)e^{-i\omega k}$$

The sequence of weights $\{\boldsymbol{\lambda}_n(k)\}$ called the *lag window* decreases at a rate appropriate for consistent estimation of $h_{XY}(\boldsymbol{\omega})$.

An algebraically equivalent method of estimating $h_{XY}(\boldsymbol{\omega})$. consists of locally smoothing the modified cross periodogram in a neighborhood of $\boldsymbol{\omega}$. Let

$$I_{n,N,\tilde{X}\tilde{Y}}$$

denote the modified cross periodogram of the centered and padded realizations

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

Then, an estimate of the nonnormalized cross-spectral density is given by

$$\hat{h}_{XY}(\omega) = \frac{2\pi}{N}\sum_{p=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}\tilde{Y}}(\omega_p)W_n(\omega - \omega_p) \qquad (5)$$

where $W_n(\boldsymbol{\theta})$ is the spectral window.

Another method of estimating $h_{XY}(\boldsymbol{\omega})$ is given by

$$\hat{h}_{XY}(\omega) = \sum_{j} w_j I_{n,N,\tilde{X}\tilde{Y}}(\omega_{p,j}) \qquad (6)$$

where $\boldsymbol{\omega}_{p,j}$, $p(\boldsymbol{\omega})$, and the weights $\{w_j\}$ are as defined in the univariate setting.

Given estimates

$$\hat{h}_{XY}(\omega)\hat{\sigma}_{XX}(0) \text{ and } \hat{\sigma}_{YY}(0)$$

the estimate of the normalized cross-spectral density is given by

$$\hat{f}_{XY}(\omega) = \frac{\hat{h}_{XY}(\omega)}{\hat{\sigma}_{XX}(0)\hat{\sigma}_{YY}(0)}$$

This follows directly from the definition of $f_{XY}(\boldsymbol{\omega})$.

BCTR

Performs a forward or an inverse Box-Cox (power) transformation.

Required Arguments

Z — Vector of length **NOBS** containing the data. (Input)

POWER — Exponent parameter in the power transformation. (Input)

SHIFT — Shift parameter in the power transformation. (Input)
SHIFT must satisfy the relation min(**Z**(*i*)) + **SHIFT** > 0.

X — Vector of length **NOBS** containing the transformed data. (Output)
If **Z** is not needed, then **X** and **Z** can occupy the same storage locations. In this case, **IPRINT** = 1 will print two identical vectors.

Optional Arguments

NOBS — Number of observations in **Z**. (Input)
NOBS must be greater than or equal to one.
Default: **NOBS** = size (**Z**,1).

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	Prints Z and the transformed data, X.

IDIR — Direction of transformation option. (Input)
Default: **IDIR** = 0.

IDIR	Action
0	Forward transformation.
1	Inverse transformation.

FORTRAN 90 Interface

Generic: CALL BCTR (Z, POWER, SHIFT, X [, ...])
Specific: The specific interface names are S_BCTR and D_BCTR.

FORTRAN 77 Interface

Single: CALL BCTR (NOBS, Z, IPRINT, IDIR, POWER, SHIFT, X)
Double: The double precision name is DBCTR.

Description

Routine BCTR performs a forward or inverse Box-Cox transformation of the n = NOBS observations $\{Z_t\}$ for $t = 1, 2, \ldots, n$.

The forward transformation is useful in the analysis of linear models or models with nonnormal errors or nonconstant variance (Draper and Smith 1981, page 222). In the time series setting, application of the appropriate transformation and subsequent differencing of a series may enable model identification and parameter estimation in the class of homogeneous stationary autoregressive-moving average models. The inverse transformation may later be applied to certain results of the analysis, such as forecasts and probability limits of forecasts, in order to express the results in the scale of the original data. A brief note concerning the choice of transformations in ARIMA models is given in Box and Jenkins (1976, page 328). The class of power transformations discussed by Box and Cox (1964) is defined by

$$X_t = \begin{cases} \dfrac{(Z_t + \xi)^{\lambda} - 1}{\lambda} & \lambda \neq 0 \\ \ln(Z_t + \xi) & \lambda = 0 \end{cases}$$

where $Z_t + \xi > 0$ for all t. Since

$$\lim_{\lambda \to 0} \frac{\left(Z_t + \xi\right)^{\lambda} - 1}{\lambda} = \ln\left(Z_t + \xi\right)$$

the family of power transformations is continuous.

Let λ = POWER and ξ = SHIFT; then, the computational formula utilized by routine BCTR is given by

$$X_t = \begin{cases} \frac{\left(Z_t + \xi\right)^{\lambda}}{\lambda} & \lambda \neq 0 \\ \ln\left(Z_t + \xi\right) & \lambda = 0 \end{cases}$$

where $Z_t + \xi > 0$ for all t. The computational and Box-Cox formulas differ only in the scale and the origin of the transformed data. Consequently, the general analysis of the data is unaffected (Draper and Smith 1981, page 225).

The inverse transformation is computed by

$$X_t = \begin{cases} Z_t^{1/\lambda} - \xi & \lambda \neq 0 \\ \exp(Z_t) - \xi & \lambda = 0 \end{cases}$$

where $\{Z_t\}$ now represents the result computed by **BCTR** for a forward transformation of the original data using parameters λ and ξ.

Comments

1. Informational errors

Type	Code	Description
4	1	For the specified forward transformation, the minimum element of X will underflow.
4	2	For the specified forward transformation, the maximum element of X will overflow.
4	3	For the specified inverse transformation, the maximum element of X will overflow.
4	4	For the specified inverse transformation, the minimum element of X will underflow.

2. The forward transformation is performed prior to fitting a model. Differencing of the data is done after the data are transformed.
3. The inverse transformation is performed on results such as forecasts and their corresponding probability limits.

Examples

Example 1

Consider the Airline Data (Box and Jenkins 1976, page 531) consisting of the monthly total number of international airline passengers from January 1949 through December 1960. Routine **BCTR** is used to compute a forward Box-Cox transformation of the first 12 observations. In the transformation **SHIFT** and **POWER** are each set to zero, which corresponds to taking natural logarithms of the data.

```
      USE GDATA_INT
      USE BCTR_INT

      IMPLICIT   NONE
      INTEGER IPRINT, NOBS
      PARAMETER (IPRINT=1, NOBS=12)
!
      INTEGER    NCOL, NROW
      REAL       POWER, SHIFT, X(NOBS), Z(144, 1)
!                                Airline Data
      CALL GDATA (4, Z, NROW, NCOL)
!                                Forward direction
!                                Transformation parameters
      POWER = 0.0
      SHIFT = 0.0
!                                Compute natural logarithms of
!                                first 12 observations in Z
      CALL BCTR (Z(:,1), POWER, SHIFT, X, NOBS=NOBS, IPRINT=IPRINT)
!
      END
```

Output

```
 Output from BCTR

 I        Z           X
  1      112.00      4.7185
  2      118.00      4.7707
  3      132.00      4.8828
  4      129.00      4.8598
  5      121.00      4.7958
  6      135.00      4.9053
  7      148.00      4.9972
  8      148.00      4.9972
  9      136.00      4.9127
 10      119.00      4.7791
 11      104.00      4.6444
 12      118.00      4.7707
```

Example 2

The estimated standard errors of forecasts (lead times 1 through 12 at origin July 1957) using the transformed Airline Data (Box and Jenkins 1976, page 311) may be converted back to their original scale using routine **BCTR**. The backward Box-Cox transformation with **SHIFT** and **POWER** each set to zero corresponds to using the exponential function.

```
      USE BCTR_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=12)
!
      INTEGER    IDIR, IPRINT
      REAL       POWER, SD(NOBS), SHIFT, X(NOBS)
!                                Standard errors of forecasts
      DATA SD/3.7, 4.3, 4.8, 5.3, 5.8, 6.2, 6.6, 6.9, 7.2, 7.6, 8.0, &
          8.2/
!
      SD=SD * 1.0E-2
!                                Backward direction
      IDIR = 1
!                                Transformation parameters
      POWER = 0.0
      SHIFT = 0.0
!                                Transform standard errors from
!                                log scale to original scale
      IPRINT = 1
      CALL BCTR (SD, POWER, SHIFT, X, IPRINT=IPRINT, IDIR=IDIR)
!
      END
```

Output

```
 Output from BCTR

 I        Z           X
 1    0.037000      1.0377
 2    0.043000      1.0439
 3    0.048000      1.0492
 4    0.053000      1.0544
 5    0.058000      1.0597
 6    0.062000      1.0640
 7    0.066000      1.0682
 8    0.069000      1.0714
 9    0.072000      1.0747
10    0.076000      1.0790
11    0.080000      1.0833
12    0.082000      1.0855
```

DIFF

Differences a time series.

Required Arguments

Z — Vector of length **NOBSZ** containing the time series. (Input)

IPER — Vector of length **NDIFF** containing the periods at which **Z** is to be differenced. (Input)
The elements of **IPER** must be greater than or equal to one.

IORD — Vector of length **NDIFF** containing the order of each difference given in **IPER**. (Input)
The elements of **IORD** must be greater than or equal to zero.

NOBSX — Number of observations in the differenced series **X**. (Output)
NOBSX = **NOBSZ - IMISS * NLOST**.

X — Vector of length **NOBSX** containing the differenced series. (Output)

Optional Arguments

NOBSZ — Number of observations in the time series **Z**. (Input)
NOBSZ must be greater than or equal to one.
Default: **NOBSZ** = size (**Z**,1).

NDIFF — Number of differences to perform. (Input)
NDIFF must be greater than or equal to one.
Default: **NDIFF** = size (**IPER**,1).

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the number of observations lost because of differencing Z, the number of observations in the differenced series X, and the differenced series X

IMISS — Missing value option. (Input)
Default: **IMISS** = 0.

IMISS	Action
0	Include missing values in X.
1	Exclude missing values from X.

NLOST — Number of observations lost because of differencing the time series Z. (Output)
NLOST = IPER(1) * IORD(1) + … + IPER(NDIFF) * IORD(NDIFF).

FORTRAN 90 Interface

Generic: CALL DIFF (Z, IPER, IORD, NOBSX, X [, …])
Specific: The specific interface names are S_DIFF and D_DIFF.

FORTRAN 77 Interface

Single: CALL DIFF (NOBSZ, Z, NDIFF, IPER, IORD, IPRINT, IMISS, NLOST, NOBSX, X)
Double: The double precision name is DDIFF.

Description

Routine DIFF performs m = NDIFF successive backward differences of period s_i= IPER(i) and order d_i = IORD(i) for i = 1, …, m on the n = NOBSZ observations $\{Z_t\}$ for t = 1, 2, …, n.

Consider the backward shift operator B given by

$$B^k Z_t = Z_{t-k}, \text{ for all } k$$

Then, the *backward difference operator* with period s is defined by

$$\nabla_s Z_t = (1 - B^s)Z_t = Z_t - Z_{t-s}, s \geq 0$$

Note that $B^S Z_t$ and $\nabla_S Z_t$ are defined only for $t = (s + 1), \ldots, n$. Repeated differencing with period s is simply

$$\nabla_s^d Z_t = (1 - B^s)^d Z_t = \sum_{j=0}^{d} \frac{d!}{j!(d-j)!}(-1)^j B^{sj} Z_t$$

where $d \geq 0$ is the order of differencing. Note that

$$\nabla_s^d Z_t$$

is defined only for $t = (sd + 1), \ldots, n$.

The general difference formula used in routine **DIFF** is given by

$$X_t = \begin{cases} \text{NaN} & t = 1, \ \ldots, n_L \\ \nabla_{s_1}^{d_1} \nabla_{s_2}^{d_2} \ldots \nabla_{s_m}^{d_m} Z_t & t = n_L + 1, \ \ldots, n \end{cases}$$

where n_L = **NLOST** represents the number of observations "lost" because of differencing and NaN (not a number) represents the missing value code. See the routine **AMACH**; in the "Machine-Dependent Constants" section of the Reference Material. Note that $n_L = \Sigma_j s_j d_j$.

A homogeneous stationary time series may be arrived at by appropriately differencing a homogeneous nonstationary time series (Box and Jenkins 1976, page 85). Preliminary application of an appropriate transformation followed by differencing of a series may enable model identification and parameter estimation in the class of homogeneous stationary autoregressive-moving average models.

Comments

1. Workspace may be explicitly provided, if desired, by use of **D2FF/DD2FF**. The reference is:

   ```
   CALL D2FF (NOBSZ, Z, NDIFF, IPER, IORD, IPRINT, IMISS, NLOST, NOBSX, X, XWK)
   ```

   The additional argument is:

   XWK — Work vector of length equal to **NOBSZ**.

2. A value is considered to be missing if it is not itself in the data set or if it is the result of an operation involving missing value(s). In differencing, missing values occur at the beginning of the differenced series since **X**(*i*) = **Z**(*i*) **-** **Z**(*i* **-** *k*) is not defined for *k* greater than or equal to *i*.

Example

Consider the Airline Data (Box and Jenkins 1976, page 531) consisting of the monthly total number of international airline passengers from January 1949 through December 1960. Routine **DIFF** is used to compute

$$X_t = \nabla_1 \nabla_{12} X_t = (Z_t - Z_{t-12}) - (Z_{t-1} - Z_{t-13})$$

For the first invocation of **DIFF** with **IMISS** = 0, $X_1, X_2, \ldots, X_{13}$ are set to the missing value code (NaN) and the equation is applied for t = 14, 15, ..., 24. For the second invocation of **DIFF** with **IMISS** = 1, the missing values are excluded from the output array containing the differenced series.

```
      USE GDATA_INT
      USE DIFF_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, NDIFF, NOBSZ
```

```
      PARAMETER  (IPRINT=1, NDIFF=2, NOBSZ=24)
!
      INTEGER    IMISS, IORD(NDIFF), IPER(NDIFF), NCOL, NLOST, NOBSX, &
                 NROW
      REAL       X(NOBSZ), Z(144, 1)
!                                Periods of differencing
      DATA IPER/1, 12/
!                                Orders of differencing
      DATA IORD/1, 1/
!                                Airline Data
      CALL GDATA (4, Z, NROW, NCOL)
!                                 Nonseasonal and seasonal difference
!                                 first 24 observations in Z
!
!                                 Include missing values in result X
!     USE Default IMISS = 0
      CALL DIFF (Z(:, 1), IPER, IORD, NOBSX, X, NOBSZ=NOBSZ, IPRINT=IPRINT)
!                                 Exclude missing values in result X
      IMISS = 1
      CALL DIFF (Z(:, 1), IPER, IORD, NOBSX, X, IPRINT=IPRINT, &
                 NOBSZ=NOBSZ, IMISS=IMISS)
!
      END
```

Output

```
 Output from DIFF/D2FF

 NLOST = 13
 NOBSX = 24
  I     Z(I)     X(I)
  1   112.00      NaN
  2   118.00      NaN
  3   132.00      NaN
  4   129.00      NaN
  5   121.00      NaN
  6   135.00      NaN
  7   148.00      NaN
  8   148.00      NaN
  9   136.00      NaN
 10   119.00      NaN
 11   104.00      NaN
 12   118.00      NaN
 13   115.00      NaN
 14   126.00    5.000
 15   141.00    1.000
 16   135.00   -3.000
 17   125.00   -2.000
 18   149.00   10.000
 19   170.00    8.000
 20   170.00    0.000
 21   158.00    0.000
 22   133.00   -8.000
 23   114.00   -4.000
 24   140.00   12.000

 Output from DIFF/D2FF
```

```
    NLOST = 13
    NOBSX = 11

     I     Z(I)     X(I)
     1   112.00    5.000
     2   118.00    1.000
     3   132.00   -3.000
     4   129.00   -2.000
     5   121.00   10.000
     6   135.00    8.000
     7   148.00    0.000
     8   148.00    0.000
     9   136.00   -8.00
    10   119.00   -4.000
    11   104.00   12.000
    12   118.00
    13   115.00
    14   126.00
    15   141.00
    16   135.00
    17   125.00
    18   149.00
    19   170.00
    20   170.00
    21   158.00
    22   133.00
    23   114.00
    24   140.00
```

ESTIMATE_MISSING

more...

Estimates missing values in a time series.

Required Arguments

ITIME_POINTS — Vector of length **NOBSW** containing the reference time points t_1, ..., t_{NOBSW} at which the time series was observed. (Input)
The reference time points must be of data type integer and in strictly increasing order. Reference time points for missing values must lie in the open interval (t_1, t_{NOBSW}) It is assumed that the time series, after estimation of missing values, contains values at equidistant time points where the distance between two consecutive time points is one.

W — Vector of length **NOBSW** containing the time series values. (Input)
The values must be ordered in accordance with the values in vector **ITIME_POINTS**. It is assumed that the time series, after estimation of missing values, contains values at equidistant time points where the distance between two consecutive time points is one. If the non-missing time series values are observed at time points t_1, ..., t_{NOBSW}, then missing values between t_i and t_{i+1}, i = 1, ..., **NOBSW** - 1, exist if t_{i+1} - t_i > 1. The size of the gap between t_i and t_{i+1} is then t_{i+1} - t_i - 1. The total length of the time series with non-missing and estimated missing values is t_{NOBSW} - t_i + 1, or **ITIME_POINTS(NOBSW) - ITIME_POINTS(1)** + 1.

For example, a time series with six observations with the fourth observation missing would appear as follows:

ITIME_POINTS	W
$t_1 = 1$	.0019
$t_2 = 2$	.0018
$t_3 = 3$	.0019
$t_4 = 5$	.0021
$t_5 = 6$	.0022
$t_6 = 7$	.002

In this example, NOBSW = 6 and the length of the time series, Z, including missing values is $t_{NOBSW} - t_i + 1 = 7$.

Z — Allocatable array which on return will contain the time series values together with estimates for the missing values. (Output)

Optional Arguments

NOBSW — Number of observations in time series W. (Input)
Default: NOBSW = size(W).

IMETH — Method used for missing value estimation. (Input)
Default: IMETH = 3.

IMETH	Method
0	Median.
1	Cubic Spline Interpolation.
2	AR(1) model.
3	AR(*p*) *model*.

If IMETH = 2 and the first gap begins at t_2 or t_3, the corresponding time series values are estimated using method Median. If IMETH = 3 is chosen and the first gap starts at t_2, then the values of this gap are also estimated by method Median. If the length of the series before a gap, denoted *len*, is greater than 1 and less than 2·MAXLAG, then MAXLAG is reduced to *len*/2 for the estimate of the missing values within this gap.

MAXLAG — Maximum number of autoregressive parameters, *p*, in the AR(*p*) model when **IMETH** = 3 is chosen. (Input)
See **AUTO_UNI_AR** for further information.
Default: **MAXLAG** = 10.

MAXBC — Maximum length of backcasting in the least-squares algorithm when **IMETH** = 2 is chosen. (Input)
MAXBC must be greater than or equal to zero. See **NSLSE** for further information.
Default: **MAXBC** = 0.

TOLBC — Tolerance level used to determine convergence of the backcast algorithm used when **IMETH** = 2 is chosen. (Input)
Backcasting terminates when the absolute value of a backcast is less than **TOLBC**. Typically, **TOLBC** is set to a fraction of *wstdev* where *wstdev* is an estimate of the standard deviation of the time series.
See **NSLSE** for further information.
Default: **TOLBC** = 0.01 * *wstdev*.

TOLSS — Tolerance level used to determine convergence of the nonlinear least-squares algorithm when **IMETH** = 2 is chosen. (Input)
TOLSS represents the minimum relative decrease in the sum of squares between two iterations required to determine convergence. Hence, **TOLSS** must be greater than zero and less than one.
See **NSLSE** for further information.
The default value is: $\max\{10^{-10}, EPS^{2/3}\}$ for single precision and
$\max\{10^{-20}, EPS^{2/3}\}$ for double precision,
where *EPS* = **AMACH**(4) for single precision and *EPS* = **DMACH**(4) for double precision. See the documentation for routine **AMACH** in the Reference Material.

RELERR— Stopping criterion for use in the nonlinear equation solver when **IMETH** = 2 is chosen. (Input)
See **NSLSE** for further information.
Default: **RELERR**= 100.0 * **AMACH**(4) for single precision,
RELERR= 100.0 * **DMACH**(4) for double precision.
See the documentation for routine **AMACH** in the Reference Material.

MAXIT — Maximum number of iterations allowed in the nonlinear equation solver when **IMETH** = 2 is chosen. (Input)
See **NSLSE** for further information.
Default: **MAXIT**= 200.

WMEAN — Estimate of the mean of time series **W**. (Input)
Default: **WMEAN** = 0.0.

NOBSZ — Number of elements in the time series **Z**, with estimated missing values, **NOBSZ = ITIME_POINTS(NOBSW) – ITIME_POINTS(1) + 1**. (Output)

ITIME_POINTS_FULL — Vector of length **NOBSZ** containing the reference time points of the time series **Z**. (Output)

MISS_INDEX — Vector of length **NOBSZ-NOBSW** containing the indices for the missing values in vector **ITIME_POINTS_FULL**. (Output)

FORTRAN 90 Interface

Generic: **CALL ESTIMATE_MISSING** (**ITIME_POINTS**, **W**, **Z** [, ...])

Specific: The specific interface names are **S_ESTIMATE_MISSING** and **D_ESTIMATE_MISSING**.

Description

Traditional time series analysis, as described by Box, Jenkins and Reinsel (1994), requires the observations be made at equidistant time points with no missing values. When observations are missing, the problem of determining suitable estimates occurs. Routine **ESTIMATE_MISSING** offers four methods for estimating missing values from an equidistant time series.

The Median method, **IMETH** = 0, estimates the missing observations in a gap by the median of the last four time series values before and the first four values after the gap. If enough values are not available before or after the gap then the number is reduced accordingly. This method is very fast and simple, but its use is limited to stationary ergodic series without outliers and level shifts.

The Cubic Spline Interpolation method, **IMETH** = 1, uses a cubic spline interpolation method to estimate missing values. Here the interpolation is again done over the last four time series values before and the first four values after the gap. The missing values are estimated by the resulting interpolant. This method gives smooth transitions across missing values.

The AR(1) method, **IMETH** = 2, assumes that the time series before the gap can be well described by an AR(1) process. If the last observation prior to the gap is made at time point t_m then it uses the time series values at t_1, $t_1 + 1$, ..., t_m to compute the one-step-ahead forecast at origin t_m. This value is taken as an estimate for the missing value at time point $t_m + 1$. If the value at $t_m + 2$ is also missing then the values at time points t_1, $t_1 + 1$, ..., $t_m + 1$ are used to recompute the AR(1) model, estimate the value at $t_m + 2$ and so on. The coefficient ϕ_1 in the AR(1) model is computed internally by the method of least squares from routine **NSLSE**.

The AR(*p*) method, **IMETH** = 3 ,uses an AR(*p*) model to estimate missing values by a one-step-ahead forecast First, routine **AUTO_UNI_AR** , applied to the time series prior to the missing values, is used to determine the optimum *p* from the set {0, 1, ..., **MAXLAG**} of possible values and to compute the parameters $\phi_1, \ldots, \phi_p$ of the resulting AR(*p*) model. The parameters are estimated by the method of moments. Denoting the mean of the series $w_{t_1}, w_{t_1+1}, \ldots, w_{t_m}$ by μ the one-step-ahead forecast at origin t_m, $\hat{w}_{t_m}(1)$, can be computed by the formula

$$\hat{w}_{t_m}(1) = \mu(1 - \sum_{j=1}^{p} \phi_j) + \sum_{j=1}^{p} \phi_j w_{t_m+1-j} \ .$$

This value is used as an estimate for the missing value. The procedure, starting with AUTO_UNI_AR, is then repeated for every further missing value in the gap.

All four estimation methods treat gaps of missing values in increasing time order.

Example

Consider the AR(1) process

$$W_t = \phi_1 \ W_{t-1} + a_t, \qquad t = 1, 2, 3, \ldots$$

We assume that $\{a_t\}$ is a Gaussian white noise process, $a_t \sim N(0, \sigma^2)$. Then, $E[W_t] = 0$ and $VAR[W_t] = \sigma^2 / (1 - \phi_1^2)$ (see Anderson, p. 174).

The time series in this example was artificially generated from an AR(1) process characterized by $\phi_1 = -0.7$ and $\sigma^2 = 1 - \phi_1^2 = 0.51$. This process is stationary with $VAR[W_t] = 1$. An initial value, $W_0 := a_0$ was used. The sequence $\{a_t\}$ was generated by a random number generator.

From the original series, we remove the observations at time points *t* = 130, *t* = 140, *t* = 141, *t* = 160, *t* = 175, and *t* = 176. Then, **ESTIMATE_MISSING** is used to compute estimates for the missing values by all 4 estimation methods available. The estimated values are compared with the actual values.

```
      use estimate_missing_int
    !
      implicit none
      integer                              :: i, j, k, nout
      integer, dimension(200)              :: times_1, times_2
      integer                              :: n_obs, n_miss
      integer                              :: ntimes, miss_ind
      integer, dimension(:), allocatable :: times, missing_index
      real, dimension(200)                 :: x_1, x_2
      real, dimension(:), allocatable      :: result

      real, dimension(200) :: y = (/ 1.30540,-1.37166,1.47905, &
          -0.91059,1.36191,-2.16966,3.11254, -1.99536,2.29740, &
```

```
          -1.82474,-0.25445,0.33519,-0.25480,-0.50574,-0.21429,&
          -0.45932,-0.63813,0.25646,-0.46243,-0.44104,0.42733, &
           0.61102,-0.82417,1.48537,-1.57733,-0.09846,0.46311, &
           0.49156,-1.66090,2.02808,-1.45768,1.36115,-0.65973, &
           1.13332,-0.86285,1.23848,-0.57301,-0.28210,0.20195, &
           0.06981,0.28454,0.19745, -0.16490,-1.05019,0.78652, &
          -0.40447,0.71514,-0.90003,1.83604, -2.51205,1.00526, &
          -1.01683,1.70691,-1.86564,1.84912,-1.33120, 2.35105, &
          -0.45579,-0.57773,-0.55226,0.88371,0.23138,0.59984,  &
           0.31971,0.59849,0.41873,-0.46955,0.53003,-1.17203,  &
           1.52937,-0.48017,-0.93830,1.00651,-1.41493,-0.42188,&
          -0.67010,0.58079, -0.96193,0.22763,-0.92214,1.35697, &
          -1.47008,2.47841,-1.50522, 0.41650,-0.21669,-0.90297,&
           0.00274,-1.04863,0.66192,-0.39143, 0.40779,-0.68174,&
          -0.04700,-0.84469,0.30735,-0.68412,0.25888, -1.08642,&
           0.52928,0.72168,-0.18199,-0.09499,0.67610,0.14636,  &
           0.46846,-0.13989,0.50856,-0.22268,0.92756,0.73069,  &
           0.78998,-1.01650,1.25637,-2.36179,1.99616,-1.54326, &
           1.38220,0.19674,-0.85241,0.40463,0.39523,-0.60721,  &
           0.25041,-1.24967,0.26727,1.40042,-0.66963,1.26049,  &
          -0.92074,0.05909,-0.61926,1.41550, 0.25537,-0.13240, &
          -0.07543,0.10413,1.42445,-1.37379,0.44382, -1.57210, &
           2.04702,-2.22450,1.27698,0.01073,-0.88459,0.88194,  &
          -0.25019,0.70224,-0.41855,0.93850,0.36007,-0.46043,  &
           0.18645,0.06337,0.29414,-0.20054,0.83078,-1.62530,  &
           2.64925,-1.25355,1.59094,-1.00684,1.03196,-1.58045, &
           2.04295,-2.38264,1.65095,-0.33273,-1.29092,0.14020, &
          -0.11434,0.04392,0.05293,-0.42277,0.59143,-0.03347,  &
          -0.58457,0.87030,0.19985,-0.73500,0.73640, 0.29531,  &
           0.22325,-0.60035,1.42253,-1.11278,1.30468,-0.41923, &
          -0.38019,0.50937,0.23051,0.46496,0.02459,-0.68478,   &
           0.25821,1.17655,-2.26629,1.41173,-0.68331 /)

      integer, dimension(200) :: tpoints=(/1,2,3,4,5,6,7,8,9,10,&
          11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,&
          29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,&
          47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,&
          65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,&
         83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,&
        101,102,103,104,105,106,107,108,109,110,111,112,113,114,&
        115,116,117,118,119,120,121,122,123,124,125,126,127,128,&
        129,130,131,132,133,134,135,136,137,138,139,140,141,142,&
        143,144,145,146,147,148,149,150,151,152,153,154,155,156,&
        157,158,159,160,161,162,163,164,165,166,167,168,169,170,&
        171,172,173,174,175,176,177,178,179,180,181,182,183,184,&
        185,186,187,188,189,190,191,192,193,194,195,196,197,198,&
        199,200 /)

      n_miss = 0
      times_1(1) = tpoints(1)
      times_2(1) = tpoints(1)
      x_1(1) = y(1)
      x_2(1) = y(1)
      k = 0

      DO i=1,199
         times_1(i+1) = tpoints(i+1)
         x_1(i+1) = y(i+1)
         !        Generate series with missing values
         IF ( i/=129 .AND. i/=139 .AND. i/=140 .AND. i/=159 &
```

```
                .AND. i/=174 .AND. i/=175 ) THEN
            k = k+1
            times_2(k+1) = times_1(i+1)
            x_2(k+1) = x_1(i+1)
         END IF
      END DO
      !
      n_obs = k + 1
      ntimes = tpoints(200) - tpoints(1) + 1
      n_miss = ntimes - n_obs
   !
      ALLOCATE(times(ntimes), missing_index(n_miss))

      DO j=0,3
         IF (j <= 2) THEN
            CALL estimate_missing(times_2, x_2, result, NOBSW=n_obs, &
                                  IMETH=j, ITIME_POINTS_FULL=times,  &
                                  MISS_INDEX=missing_index)
         ELSE
            CALL estimate_missing(times_2, x_2, result, NOBSW=n_obs, &
                                  IMETH=j, ITIME_POINTS_FULL=times,  &
                                  MAXLAG=20, MISS_INDEX=missing_index)
         END IF

         CALL UMACH (2, nout)
         IF (j == 0) WRITE (nout,*) "Method: Median"
         IF (j == 1) WRITE (nout,*) "Method: Cubic Spline Interpolation"
         IF (j == 2) WRITE (nout,*) "Method: AR(1) Forecast"
         IF (j == 3) WRITE (nout,*) "Method: AR(p) Forecast"

         WRITE(nout,99998)

         DO i=0,n_miss-1
           miss_ind = missing_index(i+1)
           WRITE(nout,99999) times(miss_ind), x_1(miss_ind),      &
                     result(miss_ind),                            &
                     ABS(x_1(miss_ind)-result(miss_ind))

         END DO
         WRITE(nout,*) ""
   !
         IF (ALLOCATED(result)) DEALLOCATE(result)
      END DO
   !
      IF (ALLOCATED(times)) DEALLOCATE(times)
      IF (ALLOCATED(missing_index)) DEALLOCATE(missing_index)
   !
   99998  FORMAT("time",6x,"actual",6x,"predicted",6x,"difference")
   99999  FORMAT(I4,6x,F6.3,8x,F6.3,7x,F6.3)
   END
```

Output

```
   Method: Median
   time      actual      predicted      difference
    130      -0.921         0.261        1.182
    140       0.444         0.057        0.386
    141      -1.572         0.057        1.630
```

```
 160       2.649         0.047         2.602
 175      -0.423         0.048         0.471
 176       0.591         0.048         0.543

Method: Cubic Spline Interpolation
time      actual      predicted      difference
 130      -0.921         1.541         2.462
 140       0.444        -0.407         0.851
 141      -1.572         2.497         4.069
 160       2.649        -2.947         5.596
 175      -0.423         0.251         0.673
 176       0.591         0.380         0.211

Method: AR(1) Forecast
time      actual      predicted      difference
 130      -0.921        -0.916         0.005
 140       0.444         1.019         0.575
 141      -1.572        -0.714         0.858
 160       2.649         1.228         1.421
 175      -0.423        -0.010         0.413
 176       0.591         0.037         0.555

Method: AR(p) Forecast
time      actual      predicted      difference
 130      -0.921        -0.901         0.020
 140       0.444         1.024         0.580
 141      -1.572        -0.706         0.867
 160       2.649         1.233         1.417
 175      -0.423        -0.002         0.421
 176       0.591         0.039         0.553
```

SEASONAL_FIT

more...

Estimates the optimum differencing for a non-stationary time series using an autoregressive model, AR(*p*), to adjust the series for seasonality.

Required Arguments

Z — Vector containing the time series. (Input)

MAXLAG — Maximum lag allowed when fitting an AR(*p*) model,
`1 <= MAXLAG <= NOBSZ/2`. (Input)

IPERA — `NDIFF` by *nipera* matrix containing the seasonal differences to test. (Input)
Here, *nipera* = `SIZE(IPERA, 2)` must be greater than or equal to one.
All elements of `IPERA` must be greater than or equal to one.

W — Allocatable array which, on return, will contain the differenced series. (Output)

Optional Arguments

NOBSZ— Number of observations in time series `Z`. (Input)
Default: `NOBSZ = SIZE(Z)`.

NDIFF — Number of differences to use. (Input)
`NDIFF` must be greater than or equal to one.
Default: `NDIFF = SIZE(IPERA, 1)`.

IORDA — `NDIFF` by *niorda* matrix containing the possible orders of each difference given in `IPERA`. (Input)
Here, *niorda* = `SIZE(IORDA, 2)` must be greater than or equal to one.
All elements of `IORDA` must be non-negative.
Default: `IORDA` is an `NDIFF` by 1 matrix, where `IORDA(1:NDIFF, 1) =1`.

IMISS — Missing value option. (Input)
Default: `IMISS = 0`

IMISS	Action
0	Include missing values in W.
1	Exclude missing values from W.

NOBSW — Number of observations in the differenced series **W**. (Output)
`NOBSW = NOBSZ - IMISS * NLOST`

NLOST — Number of observations lost because of differencing the time series **Z**. (Output)

IOPT_PERA — Vector of length `NDIFF` containing the column from matrix `IPERA` which produced the differenced series represented in **W**. (Output)

IOPT_ORDA — Vector of length `NDIFF` containing the column from matrix `IORDA` which produced the differenced series represented in **W**. (Output)

NPAR — The optimum value for the autoregressive lag. (Output)

AIC — Akaike's Information Criterion (AIC) for the optimum seasonally adjusted model. (Output)

FORTRAN 90 Interface

Generic: `CALL SEASONAL_FIT (Z, MAXLAG, IPERA, W [, ...])`
Specific: The specific interface names are `S_SEASONAL_FIT` and `D_SEASONAL_FIT`.

Description

Many time series contain seasonal trends and cycles that can be modeled by first differencing the series. For example, if the correlation is strong from one period to the next, the series might be differenced by a lag of 1. Instead of fitting a model to the series Z_t, the model is fitted to the transformed series: $W_t = Z_t - Z_{t-1}$. Higher order lags or differences are warranted if the series has, for example, a cycle every 4 or 13 weeks.

Routine `SEASONAL_FIT` does not center the original series. For every combination of columns in `IPERA` and `IORDA`, the series $Z_t$is converted to the seasonally adjusted series using the following computation

$$W_t(s,d) = \Delta_{s_1}^{d_1}\Delta_{s_2}^{d_2}\cdots\Delta_{s_m}^{d_m}Z_t = \prod_{i=1}^{m}(1-B^{s_i})^{d_i}Z_t = \prod_{i=1}^{m}\sum_{j=0}^{d_i}\binom{d_i}{j}(-1)^j B^{j\cdot s_i}Z_t$$

where s:= (s_1,... ,s_m) , d:= (d_1,... ,d_m) represent specific columns of arrays **IPERA** and **IORDA** respectively, and m = **NDIFF**.

This transformation of the series Z_t to $W_t(s,d)$ is accomplished using routine **DIFF**. After this transformation, a call is made to **AUTO_UNI_AR** to automatically determine the optimum lag for an AR(p) representation for $W_t(s,d)$. This procedure is repeated for every possible combination of columns of **IPERA** and **IORDA**. The series with the minimum AIC is identified as the optimum representation and returned in vector **W**.

Example

Consider the Airline Data (Box, Jenkins and Reinsel 1994, p. 547) consisting of the monthly total number of international airline passengers from January 1949 through December 1960. Routine **SEASONAL_FIT** is used to compute the optimum seasonality representation of the adjusted series

$$W_t(s,d) = \Delta_{s_1}^{d_1}\Delta_{s_2}^{d_2}Z_t = (1-B^{s_1})^{d_1}(1-B^{s_2})^{d_2}Z_t,$$

where $s = (1,1)$ or $s = (1,12)$ and $d = (1,1)$.

A differenced series with minimum AIC,

$$W_t = \Delta_1^1\Delta_{12}^1 Z_t = (Z_t - Z_{t-12}) - (Z_{t-1} - Z_{t-13}),$$

is in Figure 15, "Differenced Series."

Note: The numerical output may be viewed by removing the comments from the WRITE statement lines in the following example.

```
USE GDATA_INT
USE SEASONAL_FIT_INT

IMPLICIT NONE

integer :: nout
integer, parameter :: nobs = 144, ndiff = 2, maxlag = 10
integer :: nlost, npar, i
integer :: NROW, NCOL
```

```
    real, dimension(144,1) :: z
    real :: aic
    integer, dimension(2,2) :: ipera
    integer :: nobsw
    real, dimension(:), allocatable :: w
    integer, dimension(ndiff) :: iopt_pera, iopt_orda

    ipera(1,1) = 1; ipera(1,2) = 1
    ipera(2,1) = 1; ipera(2,2) = 12

    CALL GDATA (4, z, NROW, NCOL)

    CALL seasonal_fit(z(:,1), maxlag, ipera, w, NDIFF = ndiff, NLOST = nlost, &
                          IOPT_PERA = iopt_pera, NOBSW = nobsw, &
                          IOPT_ORDA = iopt_orda, NPAR = npar, AIC = aic)

    CALL UMACH(2,nout)
    ! WRITE (nout,*) "nlost =", nlost
    ! WRITE (nout,*) "iopt_pera = (", iopt_pera(1),",", iopt_pera(2), ")"
    ! WRITE (nout,*) "iopt_orda = (", iopt_orda(1),",", iopt_orda(2), ")"
    ! WRITE (nout,*) "Order of optimum AR process: ", npar
    ! WRITE (nout,*) "aic =", aic

    ! WRITE (nout,*) " "
    ! WRITE (nout,*) "Size of w:", nobsw
    ! WRITE (nout,*) "i  z(i,1)   w(i)"

    ! DO  i=1,nobs
    !    WRITE (nout,*) i, z(i,1), w(i)
    ! END DO

    IF (ALLOCATED(w)) DEALLOCATE(w)

    END PROGRAM
```

Output

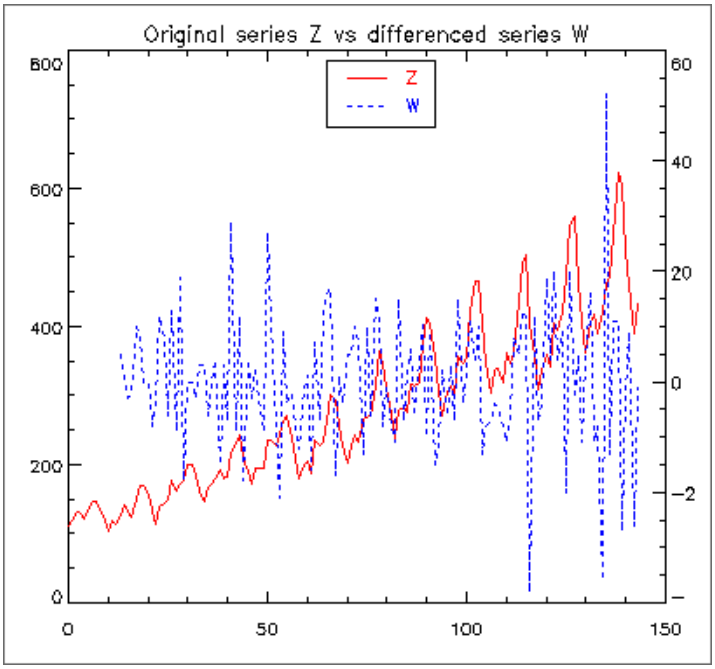


Figure 15, Differenced Series

ACF

Computes the sample autocorrelation function of a stationary time series.

Required Arguments

X — Vector of length NOBS containing the time series. (Input)

MAXLAG — Maximum lag of autocovariances, autocorrelations, and standard errors of autocorrelations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than NOBS.

AC — Vector of length MAXLAG + 1 containing the autocorrelations of the time series X. (Output)
AC(0) = 1. AC(*k*) contains the autocorrelation of lag *k* where *k* = 1, ..., MAXLAG.

Optional Arguments

NOBS — Number of observations in the time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the mean and variance.
2	Prints the mean, variance, and autocovariances.
3	Prints the mean, variance, autocovariances, autocorrelations, and standard errors of autocorrelations.

ISEOPT — Option for computing standard errors of autocorrelations. (Input)
Default: ISEOPT = 0.

ISEOPT	Action
0	No standard errors of autocorrelations are computed.
1	Computes standard errors of autocorrelations using Bartlett's formula.
2	Computes standard errors of autocorrelations using Moran's formula.

IMEAN — Option for computing the mean. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN is user specified.
1	XMEAN is set to the arithmetic mean of X.

XMEAN — Estimate of the mean of time series X. (Input, if IMEAN = 0; output, if IMEAN = 1)

ACV — Vector of length MAXLAG + 1 containing the variance and autocovariances of the time series X. (Output)
ACV(0) contains the variance of the series X. ACV(*k*) contains the autocovariance of lag *k* where *k* = 1, ..., MAXLAG.

SEAC — Vector of length MAXLAG containing the standard errors of the autocorrelations of the time series X. (Output)

The standard error of AC(*k*) is SEAC(*k*) where *k* = 1, ..., MAXLAG. If ISEOPT = 0, then SEAC may be dimensioned of length 1.

FORTRAN 90 Interface

Generic: CALL ACF (X, MAXLAG, AC [, ...])
Specific: The specific interface names are S_ACF and D_ACF.

FORTRAN 77 Interface

Single: CALL ACF (NOBS, X, IPRINT, ISEOPT, IMEAN, XMEAN, MAXLAG, ACV, AC, SEAC)
Double: The double precision name is DACF.

Description

Routine **ACF** estimates the autocorrelation function of a stationary time series given a sample of *n* = **NOBS** observations $\{X_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\hat{\mu} = \text{XMEAN}$$

be the estimate of the mean **μ** of the time series $\{X_t\}$ where

$$\hat{\mu} = \begin{cases} \mu, & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu \text{ unknown} \end{cases}$$

The autocovariance function **σ**(*k*) is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu})(X_{t+k} - \hat{\mu}), \quad k = 0,1, \ \ldots, K$$

where *K* = **MAXLAG**. Note that

$$\hat{\sigma}(0)$$

is an estimate of the sample variance. The autocorrelation function **ρ**(*k*) is estimated by

$$\hat{\rho}(k) = \frac{\hat{\sigma}(k)}{\hat{\sigma}(0)}, \quad k = 0,1, \ \ldots, K$$

Note that

$$\hat{\rho}(0) \equiv 1$$

by definition.

The standard errors of the sample autocorrelations may be optionally computed according to argument **ISEOPT**. One method (Bartlett 1946) is based on a general asymptotic expression for the variance of the sample autocorrelation coefficient of a stationary time series with independent, identically distributed normal errors. The theoretical formula is

$$\text{var}\left\{\hat{\rho}(\text{k})\right\} = \frac{1}{n}\sum_{i=-\infty}^{\infty}\left[\rho^2(i) + \rho(i-k)\rho(i+k) - 4\rho(i)\rho(k)\rho(i-k) + 2\rho^2(i)\rho^2(k)\right]$$

where

$$\hat{\rho}(k)$$

assumes **μ** is unknown. For computational purposes, the autocorrelations **ρ**(*k*) are replaced by their estimates

$$\hat{\rho}(k)$$

for $|k| \le K$, and the limits of summation are bounded because of the assumption that **ρ**(*k*) = 0 for all *k* such that $|k| > K$.

A second method (Moran 1947) utilizes an exact formula for the variance of the sample autocorrelation coefficient of a random process with independent, identically distributed normal errors. The theoretical formula is

$$\text{var}\left\{\hat{\rho}(k)\right\} = \frac{n-k}{n(n+2)}$$

where **μ** is assumed to be equal to zero. Note that this formula does not depend on the autocorrelation function.

Example

Consider the Wolfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine **ACF** computes the estimated autocovariances, estimated autocorrelations, and estimated standard errors of the autocorrelations.

```
          USE GDATA_INT
          USE ACF_INT

          IMPLICIT   NONE
          INTEGER    IPRINT, MAXLAG, NOBS
          PARAMETER  (IPRINT=3, MAXLAG=20, NOBS=100)
    !
          INTEGER    IMEAN, ISEOPT, NCOL, NROW
          REAL       AC(0:MAXLAG), ACV(0:MAXLAG), RDATA(176,2), &
                    SEAC(MAXLAG), X(NOBS), XMEAN
    !
          EQUIVALENCE (X(1), RDATA(22,2))
    !                                  Wolfer Sunspot Data for
    !                                  years 1770 through 1869
          CALL GDATA (2, RDATA, NROW, NCOL)
    !                                  Compute standard errors
          ISEOPT = 1
    !                                  Center on arithmetic mean
    !                                  USE DEFAULT IMEAN = 1
    !                                  Compute sample ACF
```

```
      CALL ACF (X, MAXLAG, AC, IPRINT=IPRINT, ISEOPT=ISEOPT)
!
      END
```

Output

```
 Output from ACF/A2F

 Mean     =     46.976
 Variance =     1382.9

 Lag         ACV           AC          SEAC

  0         1382.9      1.00000
  1         1115.0      0.80629      0.03478
  2          592.0      0.42809      0.09624
  3           95.3      0.06891      0.15678
  4         -236.0     -0.17062      0.20577
  5         -370.0     -0.26756      0.23096
  6         -294.3     -0.21278      0.22899
  7          -60.4     -0.04371      0.20862
  8          227.6      0.16460      0.17848
  9          458.4      0.33146      0.14573
 10          567.8      0.41061      0.13441
 11          546.1      0.39491      0.15068
 12          398.9      0.28848      0.17435
 13          197.8      0.14300      0.19062
 14           26.9      0.01945      0.19549
 15          -77.3     -0.05588      0.19589
 16         -143.7     -0.10394      0.19629
 17         -202.0     -0.14610      0.19602
 18         -245.4     -0.17743      0.19872
 19         -230.8     -0.16691      0.20536
 20         -142.9     -0.10332      0.20939
```

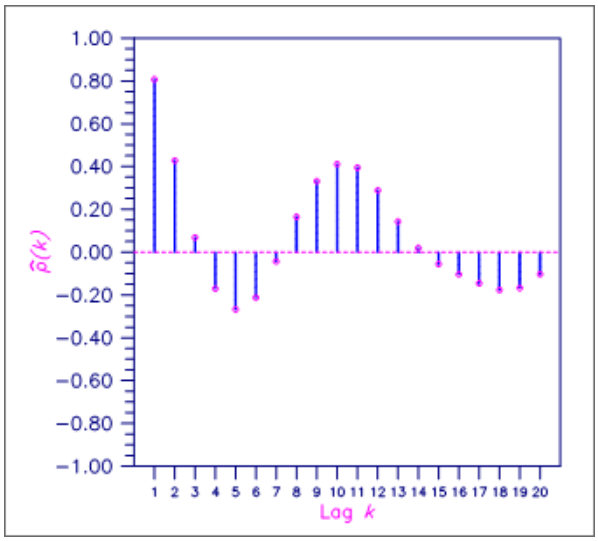


Figure 16, Sample Autocorrelation Function

PACF

Computes the sample partial autocorrelation function of a stationary time series.

Required Arguments

MAXLAG — Maximum lag of partial autocorrelations to be computed. (Input)

AC — Vector of length MAXLAG+ 1 containing the autocorrelations of the time series X. (Input)
AC(0) = 1. AC(*k*) contains the autocorrelation of lag *k* where *k* = 1, ..., MAXLAG.

PAC — Vector of length MAXLAG containing the partial autocorrelations of the time series X. (Output)
The partial autocorrelation of lag *k* corresponds to PAC(*k*) where *k* = 1, ..., MAXLAG.

FORTRAN 90 Interface

Generic: CALL PACF (MAXLAG, AC, PAC)
Specific: The specific interface names are S_PACF and D_PACF.

FORTRAN 77 Interface

Single: CALL PACF (MAXLAG, AC, PAC)
Double: The double precision name is DPACF.

Description

Routine PACF estimates the partial autocorrelations of a stationary time series given the *K* = MAXLAG sample autocorrelations

$$\hat{\rho}(k)$$

for *k* = 0, 1, ..., *K*. Consider the AR(*k*) process defined by

$$X_t = \phi_{k1}X_{t-1} + \phi_{k2}X_{t-2} + \cdots + \phi_{kk}X_{t-k} + A_t$$

where ϕ_{kj} denotes the *j*-th coefficient in the process. The set of estimates

$$\left\{\hat{\phi}_{kk}\right\}$$

for $k = 1, \ldots, K$ is the sample partial autocorrelation function. The autoregressive parameters

$$\left\{\hat{\phi}_{kj}\right\}$$

for $j = 1, \ldots, k$ are approximated by Yule-Walker estimates for successive AR(k) models where $k = 1, \ldots, K$. Based on the sample Yule-Walker equations

$$\hat{\rho}(j) = \hat{\phi}_{k1}\hat{\rho}(j-1) + \hat{\phi}_{k2}\hat{\rho}(j-2) + \cdots + \hat{\phi}_{kk}\hat{\rho}(j-k), \quad j = 1,2, \ldots, k$$

a recursive relationship for $k = 1, \ldots, K$ was developed by Durbin (1960). The equations are given by

$$\hat{\phi}_{kk} = \begin{cases} \hat{\rho}(1) & k = 1 \\ \dfrac{\hat{\rho}(k) - \sum_{j=1}^{k-1}\hat{\phi}_{k-1,j}\hat{\rho}(k-j)}{1 - \sum_{j=1}^{k-1}\hat{\phi}_{k-1,j}\hat{\rho}(j)} & k = 2, \ \ldots, K \end{cases}$$

and

$$\hat{\phi}_{kj} = \begin{cases} \hat{\phi}_{k-1,j} - \hat{\phi}_{kk}\hat{\phi}_{k-1,k-j} & j = 1,2, \ \ldots, k-1 \\ \hat{\phi}_{kk} & j = k \end{cases}$$

This procedure is sensitive to rounding error and should not be used if the parameters are near the nonstationarity boundary. A possible alternative would be to estimate $\{\phi_{kk}\}$ for successive AR(k) models using least squares (IMSL routine NSLSE) or maximum likelihood. Based on the hypothesis that the true process is AR(p), Box and Jenkins (1976, page 65) note

$$\text{var}\{\hat{\phi}_{kk}\} \simeq \frac{1}{n} \quad k \geq p+1$$

See Box and Jenkins (1976, pages 82–84) for more information concerning the partial autocorrelation function.

Comments

Workspace may be explicitly provided, if desired, by use of **P2CF/DP2CF**. The reference is:

```
CALL P2CF (MAXLAG, AC, PAC, WK)
```

The additional argument is:

WK — Work vector of length 2 * MAXLAG.

Example

Consider the Wolfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine PACF to used to compute the estimated partial autocorrelations.

```
      USE GDATA_INT
      USE ACF_INT
      USE PACF_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    IMEAN, IPRINT, ISEOPT, MAXLAG, NOBS
      PARAMETER  (IMEAN=1, IPRINT=0, ISEOPT=0, MAXLAG=20, NOBS=100)
!
      INTEGER    NCOL, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAC(MAXLAG), &
                 RDATA(176,2), SEAC(1), X(NOBS), XMEAN
      CHARACTER  CLABEL(2)*4, RLABEL(1)*6
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NUMBER'/, CLABEL/'Lag ', 'PACF'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACF
      CALL ACF (X, MAXLAG, AC)
!                                Compute sample PACF
      CALL PACF (MAXLAG, AC, PAC)
!                                Print results
      CALL WRRRL (' ', PAC, RLABEL, CLABEL, FMT= '(F8.3)')
!
      END
```

Output

```
 Lag      PACF
  1     0.806
  2    -0.635
  3     0.078
  4    -0.059
  5    -0.001
  6     0.172
  7     0.109
  8     0.110
  9     0.079
 10     0.079
 11     0.069
 12    -0.038
 13     0.081
 14     0.033
```

```
    15    -0.035
    16    -0.131
    17    -0.155
    18    -0.119
    19    -0.016
    20    -0.004
```

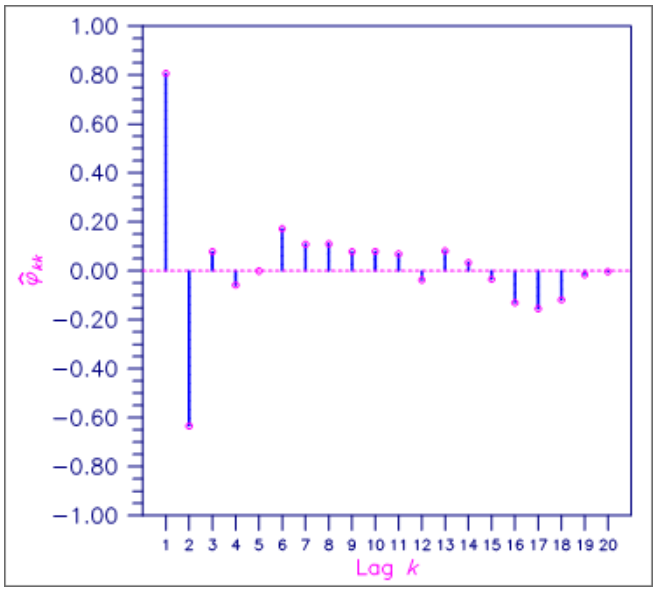


Figure 17, Sample Partial Autocorrelation Function

CCF

Computes the sample cross-correlation function of two stationary time series.

Required Arguments

X — Vector of length NOBS containing the first time series. (Input)
NOBS must be greater than or equal to two.

Y — Vector of length NOBS containing the second time series. (Input)

MAXLAG — Maximum lag of cross-covariances and cross-correlations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than NOBS.

CC — Vector of length 2 * MAXLAG + 1 containing the cross-correlations between the time series X and Y. (Output)
The cross-correlation between X and Y at lag *k* corresponds to CC(*k*) where
k = -MAXLAG, ..., -1, 0, 1, ..., MAXLAG.

Optional Arguments

XMEAN — Estimate of the mean of time series X. (Input, if IMEAN = 0; output, if IMEAN = 1)
Default: XMEAN = 0.0.

YMEAN — Estimate of the mean of time series Y. (Input, if IMEAN = 0; output, if IMEAN = 1)
Default: YMEAN = 0.0.

XVAR — Variance of the time series X. (Output)

YVAR — Variance of the time series Y. (Output)

CCV — Vector of length 2 * MAXLAG + 1 containing the cross-covariances between the time series X and Y. (Output)

The cross-covariance between X and Y at lag *k* corresponds to CCV(*k*) where
k = -MAXLAG, ..., -1, 0, 1, ..., MAXLAG.

NOBS — Number of observations in each time series. (Input)
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: `IPRINT` = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the means and variances.
2	Prints the means, variances, and cross-covariances.
3	Prints the means, variances, cross-covariances, cross-correlations, and standard errors of cross-correlations.

ISEOPT — Option for computing standard errors of cross correlations. (Input)
Default: `ISEOPT` = 0.

ISOPT	Action
0	No standard errors of cross-correlations are computed.
1	Compute standard errors of cross-correlations using Bartlett's formula.
2	Compute standard errors of cross-correlations using Bartlett's formula with the assumption of no cross-correlation.

IMEAN — Option for computing the mean. (Input)
Default: `IMEAN` = 1.

IMEAN	Action
0	`XMEAN` and `YMEAN` are user specified.
1	`XMEAN` and `YMEAN` are set to the arithmetic means of `X` and `Y`.

SECC — Vector of length 2 * `MAXLAG` + 1 containing the standard errors of the crosscorrelations between the time series `X` and `Y`. (Output)
The standard error of `CC`(*k*) is `SECC`(*k*) where *k* = `-MAXLAG`, ..., `-`1, 0, 1, ..., `MAXLAG`.

FORTRAN 90 Interface

Generic: `CALL CCF` (`X`, `Y`, `MAXLAG`, `CC`[, ...])
Specific: The specific interface names are `S_CCF` and `D_CCF`.

FORTRAN 77 Interface

Single: CALL CCF (NOBS, X, Y, MAXLAG, IPRINT, ISEOPT, IMEAN, XMEAN, YMEAN, XVAR, YVAR, CCV, CC, SECC)

Double: The double precision name is DCCF.

Description

Routine **CCF** estimates the cross-correlation function of two jointly stationary time series given a sample of n = **NOBS** observations $\{X_t\}$ and $\{Y_t\}$ for t = 1, 2, ..., n.

Let

$$\hat{\mu}_x = \text{XMEAN}$$

be the estimate of the mean μ_X of the time series $\{X_t\}$ where

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The autocovariance function of $\{X_t\}$, $\sigma_X(k)$, is estimated by

$$\hat{\sigma}_X(k) = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu}_X)(X_{t+k} - \hat{\mu}_X), \quad k = 0,\ 1,\ \ \ldots\ ,\ K$$

where K = **MAXLAG**. Note that

$$\hat{\sigma}_X(0)$$

is equivalent to the sample variance **XVAR**. The autocorrelation function $\rho_X(k)$ is estimated by

$$\hat{\rho}_X(k) = \frac{\hat{\sigma}_X(k)}{\hat{\sigma}_X(0)} \quad k = 0,1,\ \ \ldots\ ,\ K$$

Note that

$$\hat{\rho}_X(0) \equiv 1$$

by definition. Let

$$\hat{\mu}_Y = \text{YMEAN},\ \hat{\sigma}_Y(k),\ \text{and}\ \hat{\rho}_Y(k)$$

be similarly defined.

The cross-covariance function $\sigma_{XY}(k)$ is estimated by

$$\hat{\sigma}_{XY}(k) = \begin{cases} \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \hat{\mu}_X)(Y_{t+k} - \hat{\mu}_Y) & k = 0,1,\ \ldots,\ K \\ \frac{1}{n}\sum_{t=1-k}^{n}(X_t - \hat{\mu}_X)(Y_{t+k} - \hat{\mu}_Y) & k = -1,-2,\ \ldots,\ -K \end{cases}$$

The cross-correlation function $\rho_{XY}(k)$ is estimated by

$$\hat{\rho}_{XY}(k) = \frac{\hat{\sigma}_{XY}(k)}{\left[\hat{\sigma}_X(0)\hat{\sigma}_Y(0)\right]^{1/2}} \quad k = 0, \pm 1,\ \ldots,\ \pm K$$

The standard errors of the sample cross-correlations may be optionally computed according to argument `ISEOPT`. One method is based on a general asymptotic expression for the variance of the sample cross-correlation coefficient of two jointly stationary time series with independent, identically distributed normal errors given by Bartlett (1978, page 352). The theoretical formula is

$$\begin{aligned} var\left\{\hat{\rho}_{XY}(k)\right\} = \frac{1}{n-k}\sum_{i=-\infty}^{\infty}[&\rho_X(i)\rho_Y(i) + \rho_{XY}(i-k)\rho_{XY}(i+k) \\ &-2\rho_{XY}(k)\left\{\rho_X(i)\rho_{XY}(i+k) + \rho_{XY}(-i)\rho_Y(i+k)\right\} \\ &+\rho_{XY}^2(k)\left\{\rho_X(i) + \tfrac{1}{2}\rho_X^2(i) + \tfrac{1}{2}\rho_Y^2(i)\right\}] \end{aligned}$$

For computational purposes, the autocorrelations $\rho_X(k)$ and $\rho_Y(k)$ and the cross-correlations $\rho_{XY}(k)$ are replaced by their corresponding estimates for $|k| \le K$, and the limits of summation are equal to zero for all k such that $|k| > K$.

A second method evaluates Bartlett's formula under the additional assumption that the two series have no cross-correlation. The theoretical formula is

$$\text{var}\left\{\hat{\rho}_{XY}(k)\right\} = \frac{1}{n-k}\sum_{i=-\infty}^{\infty}\rho_X(i)\rho_Y(i) \quad k \ge 0$$

For additional special cases of Bartlett's formula, see Box and Jenkins (1976, page 377).

An important property of the cross-covariance coefficient is $\sigma_{XY}(k) = \sigma_{YX}(-k)$ for $k \ge 0$. This result is used in the computation of the standard error of the sample cross-correlation for lag $k < 0$. In general, the cross-covariance function is not symmetric about zero so both positive and negative lags are of interest.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2F/DC2F. The reference is:

   ```
   CALL C2F (NOBS, X, Y, MAXLAG, IPRINT, ISEOPT, IMEAN, XMEAN, YMEAN, XVAR, YVAR,
       CCV, CC, SECC, ACX, ACY)
   ```

   The additional arguments are as follows:

   ACX — Work vector of length equal to MAXLAG + 1.

   ACY — Work vector of length equal to MAXLAG + 1.

2. If ISEOPT = 0, then no workspace is needed and SECC, ACX, and ACY can be dimensioned with length 1.

3. Autocovariances, autocorrelations, and standard errors of autocorrelations may be obtained by setting the first and second time series equal.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where *X* is the input gas rate in cubic feet/minute and *Y* is the percent CO_2 in the outlet gas. Routine CCF is used to computed the cross-covariances and cross-correlations between time series *X* and *Y* with lags from -MAXLAG = -10 through lag MAXLAG = 10. In addition, the estimated standard errors of the estimated cross-correlations are computed. In the first invocation with ISEOPT = 1, the standard errors are based on the assumption that autocorrelations and cross-correlations for lags greater than MAXLAG or less than -MAXLAG are zero. In the second invocation with ISEOPT = 2, the standard errors are based on the additional assumption that all cross-correlations for *X* and *Y* are zero.

```
      USE GDATA_INT
      USE CCF_INT

      IMPLICIT   NONE

      INTEGER    IPRINT, MAXLAG, NOBS
      PARAMETER  (IPRINT=3, MAXLAG=10, NOBS=296)
!
      INTEGER    IMEAN, ISEOPT, NCOL, NROW
      REAL       CC(-MAXLAG:MAXLAG), CCV(-MAXLAG:MAXLAG), &
                RDATA(296,2), SECC(-MAXLAG:MAXLAG), X(NOBS), XMEAN, &
                XVAR, Y(NOBS), YMEAN, YVAR
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                USE Default Option to estimate means.

!                                Bartlett's formula (general case)
      ISEOPT = 1
!                                Compute cross correlation function
      CALL CCF (X, Y, MAXLAG, CC, IPRINT=IPRINT, ISEOPT=ISEOPT)
```

```
!                                  Bartlett's formula (independent case)
      ISEOPT = 2
!                                  Compute cross correlation function
      CALL CCF (X, Y, MAXLAG, CC, IPRINT=IPRINT, ISEOPT=ISEOPT)
!
      END
```

Output

```
 Output from CCF/C2F

 Mean of series X     =  -0.056834
 Variance of series X =     1.1469

 Mean of series Y     =     53.509
 Variance of series Y =     10.219

 Lag          CCV           CC         SECC

 -10      -0.40450     -0.11815     0.158148
  -9      -0.50849     -0.14853     0.155750
  -8      -0.61437     -0.17946     0.152735
  -7      -0.70548     -0.20607     0.149087
  -6      -0.77617     -0.22672     0.145055
  -5      -0.83147     -0.24287     0.141300
  -4      -0.89132     -0.26035     0.138421
  -3      -0.98060     -0.28643     0.136074
  -2      -1.12477     -0.32854     0.132159
  -1      -1.34704     -0.39347     0.123531
   0      -1.65853     -0.48445     0.107879
   1      -2.04865     -0.59841     0.087341
   2      -2.48217     -0.72503     0.064141
   3      -2.88541     -0.84282     0.046946
   4      -3.16536     -0.92459     0.044097
   5      -3.25344     -0.95032     0.048234
   6      -3.13113     -0.91459     0.049155
   7      -2.83919     -0.82932     0.047562
   8      -2.45302     -0.71652     0.053478
   9      -2.05269     -0.59958     0.071566
  10      -1.69466     -0.49500     0.093933

 Output from CCF/C2F

 Mean of series X     =  -0.056834
 Variance of series X =     1.1469

 Mean of series Y     =     53.509
 Variance of series Y =     10.219

 Lag          CCV           CC         SECC

 -10      -0.40450     -0.11815      0.16275
  -9      -0.50849     -0.14853      0.16247
  -8      -0.61437     -0.17946      0.16219
  -7      -0.70548     -0.20607      0.16191
  -6      -0.77617     -0.22672      0.16163
  -5      -0.83147     -0.24287      0.16135
  -4      -0.89132     -0.26035      0.16107
  -3      -0.98060     -0.28643      0.16080
```

-2	-1.12477	-0.32854	0.16052
-1	-1.34704	-0.39347	0.16025
0	-1.65853	-0.48445	0.15998
1	-2.04865	-0.59841	0.16025
2	-2.48217	-0.72503	0.16052
3	-2.88541	-0.84282	0.16080
4	-3.16536	-0.92459	0.16107
5	-3.25344	-0.95032	0.16135
6	-3.13113	-0.91459	0.16163
7	-2.83919	-0.82932	0.16191
8	-2.45302	-0.71652	0.16219
9	-2.05269	-0.59958	0.16247
10	-1.69466	-0.49500	0.16275

MCCF

Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.

Required Arguments

X — NOBSX by NCHANX matrix containing the first time series. (Input)
Each row of X corresponds to an observation of a multivariate time series and each column of X corresponds to a univariate time series.

Y — NOBSY by NCHANY matrix containing the second time series. (Input)
Each row of Y corresponds to an observation of a multivariate time series and each column of Y corresponds to a univariate time series.

MAXLAG — Maximum lag of cross-covariances and cross-correlations to be computed. (Input)
MAXLAG must be greater than or equal to one and less than the minimum of NOBSX and NOBSY.

CC — Array of size NCHANX by NCHANY by 2 * MAXLAG + 1 containing the cross-correlations between the channels of X and Y. (Output)
The cross-correlation between channel *i* of the X series and channel *j* of the Y series at lag *k* corresponds to CC(*i*, *j*, *k*) where *i* = 1, ..., NCHANX, *j* = 1, ..., NCHANY, and *k* = -MAXLAG, ..., -1, 0, 1, ..., MAXLAG.

Optional Arguments

NOBSX — Number of observations in each channel of the first time series X. (Input)
NOBSX must be greater than or equal to two.
Default: NOBSX = size (X,1).

NCHANX — Number of channels in the first time series X. (Input)
NCHANX must be greater than or equal to one.
Default: NCHANX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
LDX must be greater than or equal to NOBSX.
Default: LDX = size (X,1).

NOBSY — Number of observations in each channel of the second time series Y. (Input)
NOBSY must be greater than or equal to two.
Default: NOBSY = size (Y,1).

NCHANY — Number of channels in the second time series Y. (Input)
NCHANY must be greater than or equal to one.
Default: NCHANY = size (Y,2).

LDY — Leading dimension of Y exactly as specified in the dimension statement of the calling program. (Input)
LDY must be greater than or equal to NOBSY.
Default: LDY = size (Y,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the means and variances.
2	Prints the means, variances, and cross-covariances.
3	Prints the means, variances, cross-covariances, and cross-correlations.

IMEAN — Option for computing the means. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN and YMEAN are user-specified.
1	XMEAN and YMEAN are set to the arithmetic means of their respective channels.

XMEAN — Vector of length NCHANX containing the means of the channels of X. (Input, if IMEAN = 0; output, if IMEAN = 1)

YMEAN — Vector of length NCHANY containing the means of the channels of Y. (Input, if IMEAN = 0; output, if IMEAN = 1)

XVAR — Vector of length NCHANX containing the variances of the channels of X. (Output)

YVAR — Vector of length NCHANY containing the variances of the channels of Y. (Output)

CCV — Array of size NCHANX by NCHANY by 2 * MAXLAG + 1 containing the cross-covariances between the channels of X and Y. (Output)
The cross-covariance between channel *i* of the X series and channel *j* of the Y series at lag *k* corresponds to CCV(*i*, *j*, *k*) where *i* = 1, ..., NCHANX, *j* = 1, ..., NCHANY, and *k* = -MAXLAG, ..., -1, 0, 1, ..., MAXLAG.

LDCCV — Leading dimension of CCV exactly as specified in the dimension statement in the calling program. (Input)
LDCCV must be greater than or equal to NCHANX.
Default: LDCCV = size (CCV,1).

MDCCV — Middle dimension of CCV exactly as specified in the dimension statement in the calling program. (Input)
MDCCV must be greater than or equal to NCHANY.
Default: MDCCV = size (CCV,2).

LDCC — Leading dimension of CC exactly as specified in the dimension statement in the calling program. (Input)
LDCC must be greater than or equal to NCHANX.
Default: LDCCV = size (CC,1).

MDCC — Middle dimension of CC exactly as specified in the dimension statement in the calling program. (Input)
MDCC must be greater than or equal to NCHANY.
Default: MDCCV = size (CC,2).

FORTRAN 90 Interface

Generic: CALL MCCF (X, Y, MAXLAG, CC [, ...])

Specific: The specific interface names are S_MCCF and D_MCCF.

FORTRAN 77 Interface

Single: CALL MCCF (NOBSX, NCHANX, X, LDX, NOBSY, NCHANY, Y, LDY, MAXLAG, IPRINT, IMEAN, XMEAN, YMEAN, XVAR, YVAR, CCV, LDCCV, MDCCV, CC, LDCC, MDCC)

Double: The double precision name is DMCCF.

Description

Routine MCCF estimates the multichannel cross-correlation function of two mutually stationary multichannel time series. Define the multichannel time series *X* by

$$X = (X_1, X_2, \ldots, X_p)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T, \qquad j = 1, 2, \ldots, p$$

with *n* = **NOBSX** and *p* = **NCHANX**. Similarly, define the multichannel time series *Y* by

$$Y = (Y_1, Y_2, \ldots, Y_q)$$

where

$$Y_j = (Y_{1j}, Y_{2j}, \ldots, Y_{mj})^T, \qquad j = 1, 2, \ldots, q$$

with *m* = **NOBSY** and *q* = **NCHANY**. The columns of *X* and *Y* correspond to individual channels of multichannel time series and may be examined from a univariate perspective. The rows of *X* and *Y* correspond to observations of *p*-variate and *q*-variate time series, respectively, and may be examined from a multivariate perspective. Note that an alternative characterization of a multivariate time series *X* considers the columns to be observations of the multivariate time series while the rows contain univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

Let

$$\hat{\mu}_X = \texttt{XMEAN}$$

be the row vector containing the means of the channels of *X*. In particular,

$$\hat{\mu}_X = \left(\hat{\mu}_{X_1}, \hat{\mu}_{X_2}, \ \ldots, \hat{\mu}_{X_p}\right)$$

where for *j* = 1, 2, ..., *p*

$$\hat{\mu}_{X_j} = \begin{cases} \mu_{X_j} & \mu_{X_j} \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_{tj} & \mu_{X_j} \text{ unknown} \end{cases}$$

Let

$$\hat{\mu}_Y = \texttt{YMEAN}$$

be similarly defined. The cross-covariance of lag *k* between channel *i* of *X* and channel *j* of *Y* is estimated by

$$\hat{\sigma}_{X_i Y_j}(k) = \begin{cases} \frac{1}{N}\sum_t (X_{ti} - \hat{\mu}_{X_i})(Y_{t+k,j} - \hat{\mu}_{Y_j}) & k = 0, 1, \ \ldots, K \\ \frac{1}{N}\sum_t (X_{ti} - \hat{\mu}_{X_i})(Y_{t+k,j} - \hat{\mu}_{Y_j}) & k = -1, -2, \ \ldots, -K \end{cases}$$

where $i = 1, \ldots, p$, $j = 1, \ldots, q$, and K = **MAXLAG**. The summation on *t* extends over all possible cross-products with *N* equal to the number of cross-products in the sum.

Let

$$\hat{\sigma}_X(0) = \text{XVAR}$$

be the row vector consisting of the estimated variances of the channels of *X*. In particular,

$$\hat{\sigma}_X(0) = (\hat{\sigma}_{X_1}(0), \hat{\sigma}_{X_2}(0), \ \ldots, \hat{\sigma}_{X_p}(0))$$

where

$$\hat{\sigma}_{X_j}(0) = \frac{1}{n}\sum_{t=1}^{n} X_{tj} - \hat{\mu}_{X_j})^2 \quad j = 1, 2, \ \ldots, p$$

Let

$$\hat{\sigma}_Y(0) = \text{YVAR}$$

be similarly defined. The cross-correlation of lag *k* between channel *i* of X and channel *j* of Y is estimated by

$$\hat{\rho}_{X_i Y_j}(k) = \frac{\hat{\sigma}_{X_i Y_j}(k)}{\left[\hat{\sigma}_{X_i}(0)\hat{\sigma}_{Y_j}(0)\right]^{1/2}} \quad k = 0, \pm 1, \ \ldots, \pm K$$

Comments

1. For a given lag *k*, the multichannel cross-covariance coefficient is defined as the array of dimension **NCHANX** by **NCHANY** whose components are the single-channel cross-covariance coefficients **CCV**(*i*, *j*, *k*). A similar definition holds for the multichannel cross-correlation coefficient.
2. Multichannel autocovariances and autocorrelations may be obtained by setting the first and second time series equal.

Example

Consider the Wolfer Sunspot Data (Y) (Box and Jenkins 1976, page 530) along with data on northern light activity (X_1) and earthquake activity (X_2) (Robinson 1967, page 204) to be a three-channel time series. Routine **MCCF** is used to computed the cross-covariances and cross-correlations between X_1 and Y and between X_2 and Y with lags from **-MAXLAG** = **-**10 through lag **MAXLAG** = 10:

```
      USE GDATA_INT
      USE MCCF_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCC, LDCCV, LDX, LDY, MAXLAG, MDCC, MDCCV, &
                 NCHANX, NCHANY, NOBSX, NOBSY
      PARAMETER  (IPRINT=3, MAXLAG=10, NCHANX=2, NCHANY=1, NOBSX=100, &
                 NOBSY=100, LDCC=NCHANX, LDCCV=NCHANX, LDX=NOBSX, &
                 LDY=NOBSY, MDCC=NCHANY, MDCCV=NCHANY)
!
      INTEGER    IMEAN, NCOL, NROW
      REAL       CC(LDCC,MDCC,-MAXLAG:MAXLAG), CCV(LDCCV,MDCCV,- &
                 MAXLAG:MAXLAG), RDATA(100,4), X(LDX,NCHANX), &
                 XMEAN(NCHANX), XVAR(NCHANX), Y(LDY,NCHANY), &
                 YMEAN(NCHANY), YVAR(NCHANY)
!
      EQUIVALENCE (X(1,1), RDATA(1,3)), (X(1,2), RDATA(1,4))
      EQUIVALENCE (Y(1,1), RDATA(1,2))
!
      CALL GDATA (8, RDATA, NROW, NCOL)
!                                USE Default Option to estimate
!                                channel means
!                                Compute multichannel CCVF and CCF
      CALL MCCF (X, Y, MAXLAG, CC, IPRINT=IPRINT)
!
      END
```

Output

```
 Channel means of X from MCCF
             1       2
         63.43   97.97

 Channel variances of X
           1        2
      2643.7   1978.4

 Channel means of Y from MCCF
             46.94

 Channel variances of Y
           1383.8

 Multichannel cross-covariance between X and Y from MCCF

 Lag K =     -10
     1  -20.51
     2   70.71
```

```
    Lag K =      -9
        1   65.02
        2   38.14

    Lag K =      -8
        1   216.6
        2   135.6

    Lag K =      -7
        1   246.8
        2   100.4

    Lag K =      -6
        1   142.1
        2    45.0

    Lag K =      -5
        1   50.70
        2  -11.81

    Lag K =      -4
        1   72.68
        2   32.69

    Lag K =      -3
        1   217.9
        2   -40.1

    Lag K =      -2
        1   355.8
        2  -152.6

    Lag K =      -1
        1   579.7
        2  -213.0

    Lag K =       0
        1   821.6
        2  -104.8

    Lag K =       1
        1   810.1
        2    55.2

    Lag K =       2
        1   628.4
        2    84.8

    Lag K =       3
        1   438.3
        2    76.0

    Lag K =       4
        1   238.8
        2   200.4

    Lag K =       5
        1   143.6
        2   283.0
```

```
    Lag K =       6
        1   253.0
        2   234.4

    Lag K =       7
        1   479.5
        2   223.0

    Lag K =       8
        1   724.9
        2   124.5

    Lag K =       9
        1   925.0
        2   -79.5

    Lag K =      10
        1   922.8
        2  -279.3

    Multichannel cross-correlation between X and Y from MCCF

    Lag K =     -10
       1  -0.01072
       2   0.04274

    Lag K =      -9
       1   0.03400
       2   0.02305

    Lag K =      -8
       1   0.1133
       2   0.0819

    Lag K =      -7
       1   0.1290
       2   0.0607

    Lag K =      -6
       1   0.07431
       2   0.02718

    Lag K =      -5
       1   0.02651
       2  -0.00714

    Lag K =      -4
       1   0.03800
       2   0.01976

    Lag K =      -3
      1   0.1139
      2  -0.0242

    Lag K =      -2
      1   0.1860
      2  -0.0923

    Lag K =      -1
      1   0.3031
```

```
      2  -0.1287

    Lag K =       0
      1   0.4296
      2  -0.0633

    Lag K =       1
      1   0.4236
      2   0.0333

    Lag K =       2
      1   0.3285
      2   0.0512

    Lag K =       3
      1   0.2291
      2   0.0459

    Lag K =       4
      1   0.1248
      2   0.1211

    Lag K =       5
      1   0.0751
      2   0.1710

    Lag K =       6
      1   0.1323
      2   0.1417

    Lag K =       7
      1   0.2507
      2   0.1348

    Lag K =       8
      1   0.3790
      2   0.0752

    Lag K =       9
      1   0.4836
      2  -0.0481

    Lag K =      10
      1   0.4825
      2  -0.1688
```

ARMME

more...

Computes method of moments estimates of the autoregressive parameters of an ARMA model.

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances of the time series *W*. (Input)
MAXLAG must be greater than or equal to NPAR + NPMA.

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. (Input)
The *k*-th sample autocovariance of *W* is denoted by ACV(*k*), *k* = 0, 1, ..., MAXLAG.

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to one.

PAR — Vector of length NPAR containing the estimates of the autoregressive parameters. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the autoregressive parameters.

FORTRAN 90 Interface

Generic: CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR [, ...])

Specific: The specific interface names are **S_ARMME** and **D_ARMME**.

FORTRAN 77 Interface

Single: **CALL ARMME** (**MAXLAG**, **ACV**, **IPRINT**, **NPMA**, **NPAR**, **PAR**)

Double: The double precision name is **DARMME**.

Description

Routine **ARMME** determines the autoregressive parameters of an ARMA process using the extended Yule-Walker equations given the K = **MAXLAG** autocovariances $\sigma(k)$ for $k = 1, \ldots, K$.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p, q) model

$$W_t = \theta_0 + \phi_1 W_{t-1} + \ldots + \phi_p W_{t-p} + A_t - \theta_1 A_{t-1} - \ldots - \theta_q A_{t-q}, \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where p = **NPAR** and q = **NPMA**. Since W_t depends only on the innovations A_t that have occurred up through time t, the p autoregressive parameters are related to the autocovariances of lags $k = q + 1, \ldots, q + p$ by the set of equations

$$\sigma(q+1) = \phi_1\sigma(q) + \phi_2\sigma(q-1) + \ldots + \phi_p\sigma(q-p+1)$$

$$\sigma(q+2) = \phi_1\sigma(q+1) + \phi_2\sigma(q) + \ldots + \phi_p\sigma(q-p+2)$$

$$\vdots$$

$$\sigma(q+p) = \phi_1\sigma(q+p-1) + \phi_2\sigma(q+p-2) + \ldots + \phi_p\sigma(q)$$

This general system of linear equations is called the extended Yule-Walker equations. For $q = 0$, the system is referred to as the Yule-Walker equations. The equivalent matrix version is given by

$$\Sigma\phi = \sigma$$

where

$$\phi = \left(\phi_1, \ldots, \phi_p\right)^T$$

$$\sum_{ij} = \sigma(|q + i - j|) \quad i,j = 1, \ldots, p$$

$$\sigma_i = \sigma(q + i) \quad i = 1, \ldots, p$$

The overall constant θ_0 is defined by

$$\theta_0 = \begin{cases} \mu & p = 0 \\ \mu\left(1 - \sum_{i=1}^{p} \phi_i\right) & p > 0 \end{cases}$$

where **μ** is the mean of W_t.

In practice, the autocovariance function is estimated by the sample autocovariances

$$\hat{\sigma}(k)$$

for $k = 1, \ldots, K$. The solution of the extended Yule-Walker equations using these sample moments yields the *method of moments* estimates of the autoregressive parameters. The overall constant may then be estimated given an estimate of **μ**. Note that the extended Yule-Walker equations may be analogously defined in terms of autocorrelations instead of autocovariances. See Box and Jenkins (1976, pages 189–191) for some comments concerning the initial estimation of autoregressive parameters using the Yule-Walker equations.

Comments

1. Workspace may be explicitly provided, if desired, by use of A2MME/DA2MME. The reference is:

   CALL A2MME (MAXLAG, ACV, IPRINT, NPMA, NPAR, PAR, A, FACT, IPVT, WK)

   The additional arguments are as follows:

   A — Work vector of length equal to NPAR^2.

   FACT — Work vector of length equal to NPAR^2.

   IPVT — Work vector of length equal to NPAR.

   WK — Work vector of length equal to NPAR.

2. Informational error

Type	Code	Description
4	1	The problem is ill-conditioned. Transformation of the data or increased precision in the calculations may be appropriate.

3. The sample autocovariance function may be obtained using the routine ACF.

4. The first element of ACV must be the sample variance of the time series.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine **ARMME** is invoked first to compute the method of moments estimates for the autoregressive parameters of an ARMA(2, 0) model given the sample autocovariances computed from routine ACF. Then, **ARMME** is invoked a second time to compute estimated autoregressive parameters for an ARMA(2, 1) model.

```
      USE UMACH_INT
      USE GDATA_INT
      USE ACF_INT
      USE ARMME_INT

      IMPLICIT   NONE
      INTEGER    IMEAN, IPRINT, ISEOPT, MAXLAG, NOBS
      PARAMETER  (IMEAN=1, IPRINT=1, ISEOPT=0, MAXLAG=4, NOBS=100)
!
      INTEGER    NCOL, NOUT, NPAR, NPMA, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAR(2), RDATA(176,2), &
                 SEAC(1), W(100), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!
      CALL UMACH (2, NOUT)
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACV
      CALL ACF (W, MAXLAG, AC, ACV=ACV)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,0) model
!                                (Box and Jenkins, page 83)
      WRITE (NOUT,*) 'ARMA(2,0) Model'
      NPAR = 2
      NPMA = 0
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR, IPRINT=IPRINT)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,1) model
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'ARMA(2,1) Model'
      NPMA = 1
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR, IPRINT=IPRINT)
!
      END
```

Output

```
 ARMA(2,0) Model

   Output PAR
     1       2
 1.318  -0.635
```

```
    ARMA(2,1) Model

      Output PAR
        1       2
    1.244  -0.575

```

MAMME

Computes method of moments estimates of the moving average parameters of an ARMA model.

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances of the time series *W*. (Input)
MAXLAG must be greater than or equal to NPAR + NPMA.

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. (Input)
The *k*-th sample autocovariance of *W* is denoted by ACV(*k*), *k* = 0, 1, ..., MAXLAG.

PAR — Vector of length NPAR containing the estimates of the autoregressive parameters. (Input)

PMA — Vector of length NPMA containing the estimates of the moving average parameters. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the moving average parameters.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If RELERR = 0.0, then the default value RELERR = 100.0 * AMACH(4) is used. See the documentation for routine AMACH in the Reference Material.
Default: RELERR = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If MAXIT = 0, then the default value MAXIT = 200 is used.
Default: MAXIT = 0.

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to one.
Default: NPMA = size (PMA,1).

FORTRAN 90 Interface

Generic: CALL MAMME (MAXLAG, ACV, PAR, PMA [, ...])
Specific: The specific interface names are S_MAMME and D_MAMME.

FORTRAN 77 Interface

Single: CALL MAMME (MAXLAG, ACV, IPRINT, NPAR, PAR, RELERR, MAXIT, NPMA, PMA)
Double: The double precision name is DMAMME.

Description

Routine MAMME estimates the moving average parameters of an ARMA process based on a system of nonlinear equations given K = MAXLAG autocovariances $\sigma(k)$ for $k = 1, \ldots, K$ and p = NPAR autoregressive parameters ϕ_i for $i = 1, \ldots, p$.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p,q) model

$$\phi(B)W_t = \theta_0 + \theta(B)A_t, \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where p = NPAR and q = NPMA Let

$$W_t' = \phi(B)W_t$$

then the autocovariances of the *derived* moving average process $W_t = \theta(B)A_t$ are given by

$$\sigma'(k) = \begin{cases} \sigma(k) & p = 0 \\ \sum_{i=0}^{p}\sum_{j=0}^{p}\phi_i\phi_j\sigma(\,|k+i-j|\,) & p \geq 1, \phi_0 \equiv -1 \end{cases}$$

where $\sigma(k)$ denotes the autocovariance function of the original W_t process. The iterative procedure for determining the moving average parameters is based on the relation

$$\sigma'(k) = \begin{cases} (1 + \theta_1^2 + \quad \cdots \quad + \theta_q^2)\sigma_A^2 & k = 0 \\ (-\theta_k + \theta_1\theta_{k+1} + \quad \cdots \quad + \theta_{q-k}\theta_q)\sigma_A^2 & k \geq 1 \end{cases}$$

Let $\tau = (\tau_0, \tau_1, \ldots, \tau_q)^T$ and $f = (f_0, f_1, \ldots, f_q)^T$ where

$$\tau_j = \begin{cases} \sigma_A & j = 0 \\ -\theta_j / \tau_0 & j = 1, \ \ldots, q \end{cases}$$

and

$$f_j = \sum_{i=0}^{q-j} \tau_i \tau_{i+j} - \sigma'(j) \quad j = 0, 1, \ \ldots, q$$

Then, the value of τ at the (i + 1)-th iteration is determined by

$$\tau^{i+1} = \tau^i - (T^i)^{-1} f^i$$

The estimation procedure begins with the initial value

$$\tau^0 = (\sqrt{\sigma'(0)}, \ 0, \ \ldots, 0)^T$$

and terminates at iteration i when either $\|f^i\|$ is less than **RELERR** or i equals **MAXIT**. The moving average parameters are determined from the final estimate of τ by setting θ_j = - τ_j / τ_0 for j = 1, ..., q.

The random shock variance is determined according to

$$\sigma_A^2 = \begin{cases} \sigma(0) - \sum_{i=1}^{p} \phi_i \sigma(i) & q = 0 \\ \tau_0^2 & q \geq 0 \end{cases}$$

In practice, both the autocovariances and the autoregressive parameters are estimated. The solution of the system of nonlinear equations using these sample moments yields the method of moments estimates of the moving average parameters and the random shock variance. Note that autocorrelations $\rho(k)$ may be used instead of autocovariances $\sigma(k)$ to compute $\sigma'(k)$ for k = 1, ..., K. See Box and Jenkins (1976, pages 203–204) for additional motivation concerning the initial estimation of moving average parameters using a Newton-Raphson algorithm.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2MME/DM2MME**. The reference is:

   ```
   CALL M2MME (MAXLAG, ACV, IPRINT, NPAR, PAR, RELERR, MAXIT, NPMA, PMA, PARWK,
       ACVMOD, TAUINI, TAU, FVEC, FJAC, R, QTF, WKNLN)
   ```

   The additional arguments are as follows:

   PARWK — Work vector of length equal to **NPAR** + 1.

   ACVMOD — Work vector of length equal to **NPMA** + 1.

   TAUINI — Work vector of length equal to **NPMA** + 1.

TAU — Work vector of length equal to NPMA + 1.

FVEC — Work vector of length equal to NPMA + 1.

FJAC — Work vector of length equal to (NPMA + 1)^2.

R — Work vector of length equal to (NPMA + 1) * (NPMA + 2)/2.

QTF — Work vector of length equal to NPMA + 1.

WKNLN — Work vector of length equal to 5 * (NPMA + 1).

2. Informational error

Type	Code	Description
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The sample autocovariance function may be computed using the routine ACF.

4. The autoregressive parameter estimates may be computed using the routine ARMME.

Example

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. Routine **MAMME** is invoked to compute the method of moments estimates for the moving average parameter of an ARMA(2,1) model given the sample autocovariances computed from routine ACF and given the estimated autoregressive parameters computed from routine ARMME.

```
      USE GDATA_INT
      USE ACF_INT
      USE ARMME_INT
      USE MAMME_INT

      IMPLICIT   NONE
      INTEGER    IMEAN, IPRINT, ISEOPT, LDX, MAXLAG, NDX, NOBS, &
                 NOPRIN, NPAR, NPMA
      PARAMETER  (IMEAN=1, IPRINT=1, ISEOPT=0, LDX=176, MAXLAG=4, &
                 NDX=2, NOBS=100, NOPRIN=0, NPAR=2, NPMA=1)
!
      INTEGER    MAXIT, NCOL, NROW
      REAL       AC(0:MAXLAG), ACV(0:MAXLAG), PAR(2), PMA(1), &
                 RDATA(LDX,NDX), RELERR, SEAC(1), W(100), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute sample ACV
      CALL ACF (W, MAXLAG, AC, ACV=ACV)
!                                Compute estimates of autoregressive
!                                parameters for ARMA(2,1) model
      CALL ARMME (MAXLAG, ACV, NPMA, NPAR, PAR)
!                                Convergence parameters
!                                Compute estimate of moving average
!                                parameter for ARMA(2,1) model
      CALL MAMME (MAXLAG, ACV, PAR, PMA, IPRINT=IPRINT)
!
      END
```

Output

```
 Output PMA from MAMME/M2MME
 -0.1241
```

NSPE

more...

Computes preliminary estimates of the autoregressive and moving average parameters of an ARMA model.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

CNST — Estimate of the overall constant. (Output)

PAR — Vector of length NPAR containing the autoregressive parameter estimates. (Output)

PMA — Vector of length NPMA containing the moving average parameter estimates. (Output)

AVAR — Estimate of the random shock variance. (Output)

Optional Arguments

NOBS — Number of observations in the stationary time series *W*. (Input)
NOBS must be greater than NPAR + NPMA + 1.
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the mean of the time series, the estimate of the overall constant, the estimates of the autoregressive parameters, the estimates of the moving average parameters, and the estimate of the random shock variance.

IMEAN — Option for centering the time series X. (Input)
Default: IMEAN = 1

IMEAN	Action
0	WMEAN is user specified.
1	WMEAN is set to the arithmetic mean of X.

WMEAN — Constant used to center the time series **X**. (Input, if **IMEAN** = 0; output, if **IMEAN** = 1)
Default: **WMEAN** = 0.0.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: **NPAR** = size (**PAR**,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: **NPMA** = size (**PMA**,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If **RELERR** = 0.0, then the default value **RELERR** = 100.0 * **AMACH**(4) is used. See the documentation for routine **AMACH** in the Reference Material.
Default: **RELERR** = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If **MAXIT** = 0, then the default value **MAXIT** = 200 is used.
Default: **MAXIT** = 0.

FORTRAN 90 Interface

Generic: CALL NSPE (W, CNST, PAR, PMA, AVAR [, ...])
Specific: The specific interface names are S_NSPE and D_NSPE.

FORTRAN 77 Interface

Single: CALL NSPE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, NPMA, RELERR, MAXIT, CNST, PAR, PMA, AVAR)
Double: The double precision name is DNSPE.

Description

Routine **NSPE** computes preliminary estimates of the parameters of an ARMA process given a sample of n = **NOBS** observations $\{W_t\}$ for t = 1, 2, ..., n.

Suppose the time series $\{W_t\}$ is generated by an ARMA(p,q) model of the form

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator,

$$\phi(B) = 1 - \phi_1(B) - \phi_2(B)^2 - \ldots - \phi_p(B)^p$$
$$\theta(B) = 1 - \theta_1(B) - \theta_2(B)^2 - \ldots - \theta_q(B)^q$$

p = **NPAR** and q = **NPMA**. Let

$$\hat{\mu} = \text{WMEAN}$$

be the estimate of the mean of the time series $\{W_t\}$ where

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} W_t & \mu \text{ unknown} \end{cases}$$

The autocovariance function $\sigma(k)$ is estimated by

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}(W_t - \hat{\mu})(W_{t+k} - \hat{\mu}) \quad k = 0, 1, \ldots, K$$

where $K = p + q$. Note that

$$\hat{\sigma}(0)$$

is an estimate of the sample variance.

Given the sample autocovariances, the routine **ARMME** is used to compute the method of moments estimates of the autoregressive parameters using the extended Yule-Walker equations

$$\hat{\sum}\hat{\phi} = \hat{\sigma}$$

where

$$\hat{\phi} = (\hat{\phi}_1, \ldots, \hat{\phi}_p)^T$$
$$\hat{\sum}_{ij} = \hat{\sigma}(|q + i - j|) \quad i,j = 1, \ldots, p$$
$$\hat{\sigma}_i = \hat{\sigma}(q + i) \quad i = 1, \ldots, p$$

The overall constant θ_0 is estimated by

$$\hat{\theta}_0 = \begin{cases} \hat{\mu} & p = 0 \\ \hat{\mu}(1 - \sum_{i=1}^{p} \hat{\phi}_i) & p > 0 \end{cases}$$

The moving average parameters are estimated using the routine MAMME. Let

$$W'_t = \phi(B) W_t$$

then the autocovariances of the *derived* moving average process

$$W'_t = \theta(B) A_t$$

are estimated by

$$\hat{\sigma}'(k) = \begin{cases} \hat{\sigma}(k) & p = 0 \\ \sum_{i=0}^{p} \sum_{j=0}^{p} \hat{\phi}_i \hat{\phi}_j \hat{\sigma}(\mid k + i - j \mid) & p \geq 1, \hat{\phi}_0 \equiv -1 \end{cases}$$

The iterative procedure for determining the moving average parameters is based on the relation

$$\sigma'(k) = \begin{cases} (1 + \theta_1^2 + \cdots + \theta_q^2)\sigma_A^2 & k = 0 \\ (-\theta_k + \theta_1 \theta_{k+1} + \cdots + \theta_{q-k}\theta_q)\sigma_A^2 & k \geq 1 \end{cases}$$

where $\sigma(k)$ denotes the autocovariance function of the original W_t process.

Let $\tau = (\tau_0, \tau_1, \ldots, \tau_q)^T$ and $f = (f_0, f_1, \ldots, f_q)^T$ where

$$\tau_j = \begin{cases} \sigma_A & j = 0 \\ -\theta_j / \tau_0 & j = 1, \ \ldots, q \end{cases}$$

and

$$f_j = \sum_{i=0}^{q-j} \tau_i \tau_{i+j} - \hat{\sigma}'(j) \quad j = 0, 1, \ \ldots, q$$

Then, the value of τ at the $(i + 1)$-th iteration is determined by

$$\tau^{i+1} = \tau^i - (T^i)^{-1} f^i$$

The estimation procedure begins with the initial value

$$\tau^0 = (\sqrt{\hat{\sigma}'(0)}, 0, \ \ldots, 0)^T$$

and terminates at iteration *i* when either $\|f^i\|$ is less than **RELERR** or *i* equals **MAXIT**. The moving average parameter estimates are obtained from the final estimate of τ by setting

$$\hat{\theta}_j = -\tau_j / \tau_0 \quad \text{for } j = 1, \ldots, q$$

The random shock variance is estimated by

$$\hat{\sigma}_A^2 = \begin{cases} \hat{\sigma}(0) - \sum_{i=1}^{p} \hat{\phi}_i \hat{\sigma}(i) & q = 0 \\ \tau_0^2 & q \geq 0 \end{cases}$$

See Box and Jenkins (1976, pages 498–500) for a description of a similar routine.

Comments

1. Workspace may be explicitly provided, if desired, by use of **N2PE/DN2PE**. The reference is:

   ```
   CALL N2PE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, NPMA, RELERR, MAXIT, CONST, PAR,
              PMA, AVAR, ACV, PARWK, AVCMOD, TAUINI, TAU, FVEC, FJAC, R,
              QTF, WKNLN, A, FAC, IPVT, WKARMM)
   ```

   The additional arguments are as follows:

   ACV — Work vector of length equal to **NPAR** + **NPMA** + 1.

   PARWK — Work vector of length equal to **NPAR** + 1.

   ACVMOD — Work vector of length equal to **NPMA** + 1.

   TAUINI — Work vector of length equal to **NPMA** + 1.

   TAU — Work vector of length equal to **NPMA** + 1.

   FVEC — Work vector of length equal to **NPMA** + 1.

   FJAC — Work vector of length equal to (**NPMA** + 1)^2.

   R — Work vector of length equal to (**NPMA** + 1) * (**NPMA** + 2)/2.

   QTF — Work vector of length equal to **NPMA** + 1.

   WKNLN — Work vector of length equal to 5 * (**NPMA** + 1).

   A — Work vector of length equal to **NPAR**^{2}.

   FAC — Work vector of length equal to **NPAR**^{2}.

   IPVT — Work vector of length equal to **NPAR**.

   WKARMM — Work vector of length equal to **NPAR**.

2. Informational error

Type	Code	Description
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The value of **WMEAN** is used in the computation of the sample autocovariances of **W** in the process of obtaining the preliminary autoregressive parameter estimates. Also, **WMEAN** is used to obtain the value of **CNST**.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine NSPE is used to compute preliminary estimates

$$\hat{\theta}_0 \quad (\text{output in CNST})$$

$$\hat{\phi}_1, \hat{\phi}_2, \quad (\text{output in PAR})$$

$$\hat{\theta}_1 \quad (\text{output in PMA})$$

$$\hat{\sigma}_A^2 \quad (\text{output in AVAR})$$

for the following ARMA (2, 1) model

$$w_t = \theta_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 A_{t-1} + A_t$$

where the errors A_t are independently distributed each normal with mean zero and variance

$$\sigma_A^2$$

```
      USE GDATA_INT
      USE NSPE_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDX, NDX, NOBS, NOPRIN, NPAR, NPMA
      PARAMETER  (IPRINT=1, LDX=176, NDX=2, NOBS=100, NOPRIN=0, NPAR=2, &
                 NPMA=1)
!
      INTEGER    IMEAN, MAXIT, NCOL, NROW
      REAL       AVAR, CNST, PAR(NPAR), PMA(NPMA), RDATA(LDX,NDX), &
                 RELERR, W(NOBS), WMEAN
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
```

```
      CALL GDATA (2, RDATA, NROW, NCOL )
!                                 USE Default Convergence parameters
!                                 Compute preliminary parameter
!                                 estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR, IPRINT=IPRINT)
!
      END
```

Output

```
 Results from NSPE/N2PE

 WMEAN =     46.9760
 CONST =     15.5440
 AVAR  =     287.242

      PAR
        1       2
  1.244  -0.575

   PMA
 -0.1241
```

NSLSE

more...

Computes least-squares estimates of parameters for a nonseasonal ARMA model.

Required Arguments

W — Vector of length **NOBS** containing the stationary time series. (Input)

PAR — Vector of length **NPAR** containing the autoregressive parameters.(Input/ Output)

On input, **PAR** contains the preliminary estimate. On output, **PAR** contains the final estimate.

LAGAR — Vector of length **NPAR** containing the order of the autoregressive parameters. (Input)
The elements of **LAGAR** must be greater than or equal to one.

PMA — Vector of length **NPMA** containing the moving average parameters.(Input/Output)
On input, **PMA** contains the preliminary estimate. On output, **PMA** contains the final estimate.

LAGMA — Vector of length **NPMA** containing the order of the moving average parameters. (Input)
The elements of **LAGMA** must be greater than or equal to one.

MAXBC — Maximum length of backcasting. (Input)
MAXBC must be greater than or equal to zero.

CNST — Estimate of the overall constant. (Output)
For **IMEAN** = 0, **CNST** is set to zero. For **IMEAN** = 1,
CNST = **WMEAN** * (1 - **PAR**(1) - **PAR**(2) - ... - **PAR**(**NPAR**)).

COV — **NP** by **NP** variance-covariance matrix of the estimates of the parameters where **NP** = **IMEAN** + **NPAR** + **NPMA**. (Output)
The ordering of variables in **COV** is **WMEAN** (if defined), **PAR**, and **PMA**. **NP** must 1 or more.

AVAR — Estimate of the random shock variance. (Output)
AVAR = (A(1)^2 + ... + A(NA)^2)/(**NOBS** - **IMEAN** - **NPAR** - **NPMA**).

Optional Arguments

NOBS — Number of observations in the stationary time series W. (Input)
NOBS must be greater than IARDEG + IMADEG where IARDEG = max(LAGAR(*i*)) and IMADEG = max(LAGMA(*j*)).
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the least-squares estimates of the parameters, their associated standard errors, and the residual sum of squares at the final iteration.
2	Prints the least-squares estimates of the parameters and the residual sum of squares at each iteration and at the final iteration. Print the standard errors of the parameters at the final iteration.

IMEAN — Option for centering the time series W. (Input)
Default: IMEAN = 0.

IMEAN	Action
0	W is not centered.
1	W is centered about WMEAN. Centering the time series W about WMEAN is equivalent to inclusion of the overall constant in the model.

WMEAN — Estimate of the mean of the time series W. (Input/Output, if IMEAN = 1; not used if IMEAN = 0)
For IMEAN = 1, on input, WMEAN contains the preliminary estimate, on output, WMEAN contains the final estimate.
Default: WMEAN = 0.0.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

TOLBC — Tolerance level used to determine convergence of the backcast algorithm. (Input)
Backcasting terminates when the absolute value of a backcast is less than **TOLBC**. Typically, **TOLBC** is set to a fraction of **WSTDEV** where **WSTDEV** is an estimate of the standard deviation of the time series. If **TOLBC** = 0.0, then **TOLBC** = 0.01 * **WSTDEV** is used.
Default: **TOLBC** = 0.0.

TOLSS — Tolerance level used to determine convergence of the nonlinear least-squares algorithm. (Input)
Default: **TOLSS** = 0.0.
TOLSS represents the minimum relative decrease in sum of squares between two iterations required to determine convergence. Hence, **TOLSS** must be greater than or equal to zero and less than one where **TOLSS** = 0.0 specifies the default value is to be used. The default value is

$\max\{10^{-10}, \mathbf{EPS}^{2/3}\}$ for single precision and

$\max\{10^{-20}, \mathbf{EPS}^{2/3}\}$ for double precision

where **EPS** = **AMACH**(4). See the documentation for routine **AMACH** in the Reference Material.

LDCOV — Leading dimension of **COV** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOV** = size (**COV**,1).

NA — Number of residuals computed (including backcasts). (Output)
If **NB** values of the time series are backcast, then **NA** = **NOBS - IARDEG** + **NB**.

A — Vector of length **NOBS - IARDEG** + **MAXBC** containing the residuals (including backcasts) at the final parameter estimate **point** in the first **NA** locations. (Output)

FORTRAN 90 Interface

Generic: CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, AVAR [, ...])
Specific: The specific interface names are S_NSLSE and D_NSLSE.

FORTRAN 77 Interface

Single: CALL NSLSE (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, MAXBC, TOLBC, TOLSS, CNST, COV, LDCOV, NA, A, AVAR)
Double: The double precision name is DNSLSE.

Description

Routine **NSLSE** computes least-squares estimates of parameters for a nonseasonal ARMA model given a sample of n = **NOBS** observations $\{W_t\}$ for $t = 1, 2, \ldots, n$.

Suppose the time series $\{W_t\}$ is generated by a nonseasonal ARMA model of the form

$$\phi(B)(W_t - \mu) = \theta(B)A_t \quad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator, **μ** is the mean of W_t,

$$\phi(B) = 1 - \phi_1 B^{l_\phi(1)} - \phi_2 B^{l_\phi(2)} - \cdots - \phi_p B^{l_\phi(p)} \quad p \geq 0$$
$$\theta(B) = 1 - \theta_1 B^{l_\theta(1)} - \theta_2 B^{l_\theta(2)} - \cdots - \theta_q B^{l_\theta(q)} \quad q \geq 0$$

with p = **NPAR** and q = **NPMA**. Without loss of generality, we assume

$$1 \leq l_\phi(1) \leq l_\phi(2) \leq \cdots \leq l_\phi(p)$$
$$1 \leq l_\theta(1) \leq l_\theta(2) \leq \cdots \leq l_\theta(q)$$

so that the nonseasonal ARMA model is of order (p', q') where $p' = l_\phi(p)$ and $q' = l_\theta(q)$. Note that the usual hierarchal model assumes

$$l_\phi(i) = i \quad 1 \leq i \leq p$$
$$l_\theta(j) = j \quad 1 \leq j \leq q$$

Consider the sum of squares function

$$S_T(\mu, \phi, \theta) = \sum_{-T+1}^{n} [A_t]^2$$

where

$$[A_t] = E[A_t | \mu, \phi, \theta, W]$$

and T is the *backward origin*. The random shocks $\{A_t\}$ are assumed to be independent and identically distributed

$$N(0, \sigma_A^2)$$

random variables. Hence, the *log-likelihood function* is given by

$$l(\mu,\phi,\theta,\sigma_A) = f(\mu,\phi,\theta) - n \ln \sigma_A - \frac{S_T(\mu,\phi,\theta)}{2\sigma_A^2}$$

where $f(,\mu,\phi,\theta)$ is a function of μ, ϕ, and θ.

For $T = 0$, the log-likelihood function is *conditional* on the past values of both W_t and A_t required to initialize the model. The method of selecting these initial values usually introduces transient bias into the model (Box and Jenkins 1976, pages 210–211). For T = ∞, this dependency vanishes, and the estimation problem concerns maximization of the *unconditional* log-likelihood function. Box and Jenkins (1976, page 213) argue that

$$S_\infty(\mu,\phi,\theta) / 2\sigma_A^2$$

dominates

$$l\left(\mu,\phi,\theta,\sigma_A^2\right)$$

The parameter estimates that minimize the sum of squares function are called *least-squares estimates*. For large *n*, the unconditional least-squares estimates are approximately equal to the maximum likelihood estimates.

In practice, a finite value of *T* will enable sufficient approximation of the unconditional sum of squares function. The values of $[A_t]$ needed to compute the unconditional sum of squares are computed iteratively with initial values of W_t obtained by back-forecasting. The residuals (including backcasts), estimate of random shock variance, and covariance matrix of the final parameter estimates are also computed. Note that application of an appropriate transformation using routine BCTR followed by differencing using routine DIFF allows for fitting of nonseasonal ARIMA models. The algorithm for nonseasonal ARIMA models is developed in Chapter 7 of Box and Jenkins (1976). The extension to multiplicative seasonal ARIMA models is given in Box and Jenkins (1976, pages 500–504).

Comments

1. Workspace may be explicitly provided, if desired, by use of N2LSE/DN2LSE. The reference is:

   ```
   CALL N2LSE  (NOBS, W, IPRINT, IMEAN, WMEAN, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA,
      MAXBC, TOLBC, TOLSS, CNST, COV, LDCOV, NA, A, AVAR, XGUESS, XSCALE, FSCALE, X,
      FVEC, FJAC, LDFJAC, RWKUNL, IWKUNL, WKNSRE, AI, FCST)
   ```

   The additional arguments are as follows:

   XGUESS — Work vector of length NP.

   XSCALE — Work vector of length NP.

   FSCALE — Work vector of length M.

   X — Work vector of length NP.

   FVEC — Work vector of length M.

FJAC — Work vector of length `M * NP`.

LDFJAC — Integer scalar equal to `M`.

RWKUNL — Work vector of length 10 `* NP` + 2 `* M -` 1.

IWKUNL — Work vector of length `NP`.

WKNSRE — Work vector of length `NOBS` + `MAXBC`.

AI — Work vector of length `IMADEG`.

FCST — Work vector of length `MAXBC`.

2 Informational error

Type	Code	Description
3	1	Least-squares estimation of the parameters has failed to converge. Increase `MAXBC` and/or `TOLBC` and/or `TOLSS`. The estimates of the parameters at the last iteration may be used as new starting values.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine NSPE is first invoked to compute preliminary estimates for an ARMA(2, 1) model. Then, NSLSE is invoked with the preliminary estimates as input in order to compute the least-squares estimates

$$\hat{\theta}_0 \quad (\text{output in CNST})$$

$$\hat{\phi}_1, \hat{\phi}_2, \quad (\text{output in PAR})$$

$$\hat{\theta}_1 \quad (\text{output in PMA})$$

$$\hat{\sigma}_A^2 \quad (\text{output in AVAR})$$

for the ARMA(2, 1) model

$$w_t = \theta_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 A_{t-1} + A_t$$

where the errors A_t are independently distributed each normal with mean zero and variance

$$\sigma_A^2$$

Note at the end of the output a warning error appears. Most of the time this error message can be ignored. There are three general reasons this error can occur.

1. Convergence was declared using the criterion based on **TOLSS**, but the gradient of the residual sum of squares function was nonzero. This occurred in this example. Either the message can be ignored or **TOLSS** can be reduced to allow more iterations and a slightly more accurate solution.
2. Convergence is declared based on the fact that a very small step was taken, but the gradient of the residual sum of squares function was nonzero. The message can usually be ignored. However, sometimes the algorithm is making very slow progress and is not near a minimum.
3. Convergence is not declared after 100 iterations.

Examination of the history of iterations using **IPRINT** = 2 and trying a smaller value for **TOLSS** can help you determine what caused the error message.

```
      USE GDATA_INT
      USE NSPE_INT
      USE NSLSE_INT

      IMPLICIT   NONE
      INTEGER    IARDEG, IMEAN, LDCOV, LDX, MAXBC, MDX, NOBS, NP, &
                 NPAR, NPMA
      PARAMETER  (IARDEG=2, IMEAN=1, LDX=176, MAXBC=10, MDX=2, &
                 NOBS=100, NPAR=2, NPMA=1, NP=NPAR+NPMA+IMEAN, &
                 LDCOV=NP)
!
      INTEGER    IPRINT, LAGAR(NPAR), LAGMA(NPMA), MAXIT, NA, NCOL, &
                 NROW
      REAL       A(NOBS-IARDEG+MAXBC), AVAR, CNST, COV(LDCOV,NP), &
                 PAR(NPAR), PMA(NPMA), RELERR, TOLBC, TOLSS, W(NOBS), &
                 WMEAN, X(LDX,MDX)
!
      EQUIVALENCE (W(1), X(22,2))
!
      DATA LAGAR/1, 2/, LAGMA/1/
!                                 Wolfer Sunspot Data for
!                                 years 1770 through 1869
      CALL GDATA (2, X, NROW, NCOL)
!                                USE Default Convergence parameters
!                                Compute preliminary parameter
!                                estimates for ARMA(2,1) model
      IPRINT = 1
      CALL NSPE (W, CNST, PAR, PMA, AVAR, IPRINT=IPRINT, WMEAN=WMEAN)
!
      TOLBC  = 0.0
      TOLSS  = 0.125
      IPRINT = 2
!
      CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, &
                 AVAR, IMEAN=IMEAN,  WMEAN=WMEAN, TOLSS=TOLSS, &
                 IPRINT=IPRINT)
!
      END
```

Output

```
Results from NSPE/N2PE

WMEAN =     46.9760
CONST =     15.5440
AVAR  =     287.242

     PAR
    1       2
1.244  -0.575

PMA
-0.1241
------------------------------------------------------------------------
Iteration       1

WMEAN =   52.638233185

     PAR
    1       2
1.264  -0.606

  PMA
-0.1731

Residual SS (including backcasts) =  23908.66210937500
Number of residuals               =     108
Number of backcasts               =      10
------------------------------------------------------------------------
Iteration       2

WMEAN =   54.756504059

     PAR
    1       2
1.360  -0.688

  PMA
-0.1411

Residual SS (including backcasts) =  23520.71484375000
Number of residuals               =     108
Number of backcasts               =      10
------------------------------------------------------------------------
Final Results, Iteration       3

Parameter         Estimate            Std. Error            t-ratio

WMEAN             53.9187279            5.5178852            9.7716293

                                  PAR
1                  1.3925704            0.0960639           14.4962845
2                 -0.7329484            0.0866115           -8.4624796

                                  PMA
1                 -0.1375125            0.1223797           -1.1236545

CNST =         18.3527489
```

```
    AVAR  =        243.4830170

    Residual SS (including backcasts) =      23374.3691406
    Number of residuals               =        108

    Residual SS (excluding backcasts) =      20931.7519531
    Number of residuals               =         98

    *** WARNING  ERROR 1 from NSLSE.  Least squares estimation of the parameters
    ***          has failed to converge.  Increase MAXBC and/or TOLBC and/or
    ***          TOLSS.  The estimates of the parameters at the last iteration
    ***          may be used as new starting values.
```

MAX_ARMA

Exact maximum likelihood estimation of the parameters in a univariate ARMA (auto-regressive, moving average) time series model.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

PAR — Vector of length NPAR. On input PAR contains initial estimates for the autoregressive parameters. On output these are replaced by the exact maximum likelihood estimates for the autoregressive parameters. (Input/Output)

PMA — Vector of length NPMA. On input PMA contains initial estimates for the moving average parameters. On output these are replaced by the exact maximum likelihood estimates for the moving average parameters. (Input/Output)

Optional Arguments

NOBS — Number of values in the time series. (Input)
Default: NOBS = size(W,1).

NPAR — Number of autoregressive parameters. (Input)
Default: NPAR = size(PAR,1).

NPMA — Number of moving average parameters. (Input)
Default: NPMA = size(PMA,1).

WMEAN — Estimate of the mean of the time series W. (Input)
Default: WMEAN = arithmetic mean of W.

IPRINT — Printing option. (Input)

IPRINT	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

MAXIT — Maximum number of estimation iterations. (Input)
Default: MAXIT = 500.

CNST — Estimate of the constant term θ_0 in the model. (Output)

AVAR — Estimate of the noise variance. (Output)

F — Value of –2*(ln(likelihood)) for fitted model. (Output)

EWS — Array of length NOBS containing the residuals of the requested ARMA fit. (Output)

FORTRAN 90 Interface

Generic: CALL MAX_ARMA (W, PAR, PMA [, ...])
Specific: The specific interface names are S_MAX_ARMA and D_MAX_ARMA.

Description

Routine MAX_ARMA is derived from the maximum likelihood estimation algorithm described by Akaike, Kitagawa, Arahata and Tada (1979), and the XSARMA routine published in the TIMSAC-78 Library.

Using the notation developed in the introduction to this chapter, the stationary time series W_t with mean μ can be represented by the nonseasonal autoregressive moving average (ARMA) model by the following relationship:

$$\phi(B)(W_t - \mu) = \theta(B)A_t$$

where

$$t \in ZZ = \{ \cdots, -2, -1, 0, 1, 2, \cdots \}$$

B is the backward shift operator defined by

$$B^k W_t = W_{t-k}$$

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_{\mathrm{NPAR}} B^{\mathrm{NPAR}}, NPAR \geq 0$$

and

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_{NPMA} B^{NPMA}, NPMA \geq 0$$

MAX_ARMA estimates the coefficients

$$\phi_1, \phi_2, \cdots, \phi_{NPAR}$$

and

$$\theta_1,\theta_2, \cdots ,\theta_{NPMA}$$

using maximum likelihood estimation.

MAX_ARMA checks the initial estimates for the autoregressive coefficients to ensure that they represent a stationary series. If

$$\phi_1,\phi_2, \cdots ,\phi_{NPAR}$$

are the initial estimates for a stationary series then all (complex) roots of the following polynomial will fall outside the unit circle:

$$1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_{NPAR} z^{NPAR}$$

MAX_ARMA computes the roots of this polynomial for the initial estimates supplied in the vector **PAR**. If these estimates represent a non-stationary series, **MAX_ARMA** issues a warning message and replaces **PAR** with initial values that are stationary.

Initial estimates can be obtained using **NSPE** and **NSLSE** procedures for calculating the autoregressive and moving average parameters of a series.

MAX_ARMA also validates its final estimates to ensure that they too represent a stationary series. This is done to guard against the possibility that **MAX_ARMA** converged to a non-stationary solution. If non-stationary estimates are encountered, **MAX_ARMA** quits and issues a fatal message. Routines **IERCD** and **N1RTY** (see the Reference Material section of this manual) can be used to verify that the stationary condition was met.

The ARMA process

$$\phi(B)(W_t - \mu) = \theta(B)A_t$$

can equivalently be written in the form

$$\phi(B)W_t = \theta_0 + \theta(B)A_t$$

where the constant term θ_0 is defined by $\left(1 - \sum_{i=1}^{NPAR}\phi_i\right)\mu$.

MAX_ARMA estimates μ always by the sample mean of the series.

For model selection, the ARMA model with the minimum value for **AIC** might be preferred

$$\mathrm{AIC} = F+2p$$

where p = **NPAR** + **NPMA**.

Comments

Informational errors

Type	Code	Description
3	1	Input values for autoregressive coefficients are invalid. They do not represent a stationary time series. New values have been generated.
4	1	Maximum number of iterations exceeded. Try increasing `MAXIT` or use double precision.
4	2	Estimation process converged to a non-stationary solution.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins, 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. In this example, **MAX_ARMA** is used to fit the following ARMA model:

$$w_t - \mu = \phi_1(w_{t-1} - \mu) + \phi_2(w_{t-2} - \mu) - \theta_1 a_{t-1} + a_t$$

For these data, **MAX_ARMA** calculated the following estimates:

$$w_t - \mu = 1.22(w_{t-1} - \mu) - 0.56(w_{t-2} - \mu) + 0.38a_{t-1} + a_t$$

Letting $\theta_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right)$, we can obtain the following equivalent representations:

$$w_t = \theta_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} - \theta_1 a_{t-1} + a_t \text{ and,}$$
$$w_t = 0.33\mu + 1.22w_{t-1} - 0.56w_{t-2} + 0.38a_{t-1} + a_t$$

```
      USE MAX_ARMA_INT
      USE GDATA_INT
      USE NSPE_INT
      IMPLICIT NONE
!                                 SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    I
      REAL(KIND(1E0)) PAR(2), PMA(1), AVAR, F
      REAL(KIND(1E0)) X(176,2)
      REAL(KIND(1E0)) CONST
      INTEGER         NCOL, NROW
!                                 Get Wolfer Sunspot Data
      CALL GDATA(2,X,NROW,NCOL)
!                                 Get preliminary PAR and PMA estimates
      CALL NSPE(X(22:,2),CONST, PAR, PMA, AVAR, NOBS=100)
!                                 TEST #1: DOCUMENT EXAMPLE
      CALL MAX_ARMA(x(22:,2), PAR, PMA, nobs=100, MAXIT=12000, &
                    AVAR=AVAR, F=F)
      WRITE(*,99994) SIZE(PAR)
```

```
      WRITE (*,99996) (PAR(I),I=1,SIZE(PAR))
      WRITE(*,99995) SIZE(PMA)
      WRITE(*,99996) (PMA(I),I=1,SIZE(PMA))

      WRITE(*,*) "-2*LN(MAXIMUM LOG LIKELIHOOD) = ", F
      WRITE(*,*) "WHITE NOISE VARIANCE = ", AVAR
99994 FORMAT(//1H ,5('-'),2X,'FINAL PAR(I)',2X,'NPAR=',I3,2X,5('-'))
99995 FORMAT(//1H ,5('-'),2X,'FINAL PMA(I)',2X,'NPMA =',I3,2X,5('-'))
99996 FORMAT(1H ,5E20.10,/(1H ,5E20.10))

      END
```

Output

```
 -----  FINAL PAR(I)  NPAR=  2  -----
     0.1224243164E+01   -0.5600821972E+00


 -----  FINAL PMA(I)  NPMA =  1  -----
    -0.3847315013E+00
 -2*LN(MAXIMUM LOG LIKELIHOOD) =  539.5841
 WHITE NOISE VARIANCE =  214.50406
```

REG_ARIMA

more...

Fits a univariate, non-seasonal ARIMA time series model with the inclusion of one or more regression variables.

Required Arguments

Y — Array of length **NOBS** containing the time series. (Input)

IMODEL — Array of length 3 containing the model order parameters. (Input)

IMODEL I	Description
1	Order of the autoregressive part, p, where $p \geq 0$.
2	Order of the non-seasonal difference operator, d, where $d \geq 0$.
3	Order of the moving average part, q, where $q \geq 0$.

If $p = 0$ and $q = 0$, only regression is performed.

PARMA —Array of length $1 + p + q$ containing the estimated autoregressive (AR) and moving average (MA) parameters of the ARIMA(p,d,q) model. **PARMA**(1) is the estimated AR constant parameter, **PARMA**(2: (p+1)) contains the AR parameter estimates and **PARMA**((p+2):), contains the MA parameter estimates. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = `size`(**Y**).

X — Array of size **NOBS** by K containing the regression data, where K = `size`(**X**,2) is the number of user supplied regression variables. (Input)
Specific columns in **X** may be selected using the **INDX** argument. Otherwise, all columns of **X** are used.
Default: No regression variables are included.

XLEAD — Array of size MXLEAD by *K* containing the regression data to be used in obtaining forecasts, where *K* = size(X,2) is the number of user supplied regression variables. (Input)
Specific columns in XLEAD may be selected using the INDX argument. Otherwise all columns of XLEAD are used.
Note: If MXLEAD > 0 and optional argument *X* is present, XLEAD is required.

INDX — Index array containing the column numbers in X and XLEAD that are to be used for the regression variables. (Input)
Default: All columns of X and XLEAD are used.

TREND — Logical. If .TRUE., the routine will include a trend variable. (Input)
Note: Setting TREND = .TRUE. has the effect of fitting an intercept term in the regression. If the difference operator IMODEL(2) = *d* > 0, the effect on the model in the original, undifferenced space is polynomial of order *d*.
Default: TREND = .TRUE..

MXLEAD — Maximum lead time for forecasts. (Input)
Note: If MXLEAD > 1, forecasts are returned for *t* = NOBS + 1, .NOBS + 2, ..., NOBS + MXLEAD
Default: MXLEAD = 0 (no forecasts).

MAXIT — Maximum number of iterations. (Input)
Default: MAXIT = 50.

IPRINT — Printing option. (Input)

IPRINT	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

PREG — Array of length *K* + *t* containing the estimated regression coefficients, where *t*=0 if TREND=.FALSE. or *t*=1 if TREND=.TRUE.. (Output)

SE — Array of length *p* + *q* containing the standard errors of the ARMA parameter estimates. (Output)

AVAR — White noise variance estimate. (Output)
Note: If IMODEL(0)+IMODEL(2)= 0 and *K*>0, AVAR is the mean squared regression residual.

REGSE — Array of length *K* + *t* containing the standard errors of the regression estimates, where *t*=0 if TREND=.FALSE. or *t*=1 if TREND=.TRUE.. (Output)

PREGVAR — Array of length (*K* + *t*) by (*K* + *t*) containing the variances and covariances of the regression coefficients, where *t*=0 if TREND=.FALSE. or *t*=1 if TREND=.TRUE.. (Output)

AIC — Akaike's Information Criterion for the fitted ARMA model. (Output)

LLIKE — Value of –2(ln(likelihood)) for fitted model. (Output)

FCST — Array of length **MXLEAD** containing the forecasts for time points *t* = **NOBS** + 1, .**NOBS** + 2, …, **NOBS** + **MXLEAD**. (Output)

FCSTVAR — Array of length **MXLEAD** containing the forecast variances for time points *t* = **NOBS** + 1, .**NOBS** + 2, …, **NOBS** + **MXLEAD**. (Output)

Fortran 90 Interface

Generic: **CALL REG_ARIMA** (**Y**, **IMODEL**, **PARMA** [, …])
Specific: The specific interface names are **S_REG_ARIMA** and **D_REG_ARIMA**.

Description

Routine **REG_ARIMA** fits an ARIMA(*p*, *d*, *q*) to a univariate time series with the possible inclusion of one or more regression variables.

Suppose Y_t, t = 1, …, *N*, is a time series such that the *d*-th difference is stationary. Further, suppose a_t is a series of uncorrelated, mean 0 random variables with variance σ_a^2.

The Auto-Regressive Integrated Moving Average (ARIMA) model for $\{Y_t, a_t\}$ can be expressed as

$$\phi(B)(1-B)^{\mathrm{d}}Y_{\mathrm{t}} = \theta(B)a_{\mathrm{t}}$$

where *B* is the backshift operator, $Bz_t = z_{t-1}, B^2 z_t = z_{t-2},$

$$\phi(B) = (1 - \phi_1 B - \phi_2 B^2 + \cdots - \phi_p B^p),$$

and

$$\theta(B) = (1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q)$$

The notation for this model is ARIMA(*p*, *d*, *q*) where *p* is the order of the autoregressive polynomial $\phi(B)$, *d* is the order of the differencing needed to make Y_t stationary, and *q* is the order of the moving-average polynomial $\theta(B)$.

The ARIMA model can be extended to include K regression variables $X_{1t}, X_{2t}, \ldots, X_{Kt}$, by using the residuals (of the multiple regression of Y_t on $X_{1t}, X_{2t}, \ldots, X_{Kt}$) in place of Y_t in the above ARIMA model.

$$\phi(B)\left(1-B\right)^{d}(Y_t-\sum_{i=1}^{K}\beta_i X_{it})=\theta\left(B\right)a_t$$

Equivalently,

$$\phi(B)w_t=\theta(B)a_t$$

where

$$w_t=(1-B)^{d}(Y_t-\sum_{i=1}^{K}\beta_i X_{it})$$

is the differenced residual series.

To estimate the (*p* + *q* + *K*) parameters of the specified regression ARIMA model, **REG_ARIMA** uses the iterative generalized least squares method (IGLS) as described in Otto, Bell, and Burman (1987).

The IGLS method iterates between two steps, one step to estimate the regression parameters via generalized least squares (GLS) and the second step to estimate the ARMA parameters. In particular, at iteration *m*, the first step finds

$$\hat{\beta}_m=(\hat{\beta}_{m1},\ ...,\ \hat{\beta}_{mK})'$$

by solving the GLS problem with weight matrix

$$\mathbf{V}(i,j)=\gamma_w(j-i)\quad i<j=1,\ ...,\ N$$

where

$$\gamma_w(j-i)=E[w_{t-j}w_{t-i}|\hat{\phi}_{m-1},\hat{\theta}_{m-1}]$$

That is, $\hat{\beta}_m$ minimizes $\left(Y-X'\beta\right)'V^{-1}\left(Y-X'\beta\right)$, where $Y=\left(Y_1,...Y_N\right)'$, X is an *N* by *K* matrix with *i*-th column, $X_i=\left(X_{i1},...X_{iN}\right)'$, and V is an *N* by *N* weight matrix defined using the theoretical autocovariances of the series $w_{m-1,t}=(1-B)^{d}(Y_t-\sum_{i=1}^{K}\hat{\beta}_{m-1,i}X_{it})$. The series $w_{m-1,t}$ is modeled as an ARMA(p,q) with parameters $\hat{\phi}_{m-1}=(\hat{\phi}_{m-1,1},\ \ldots\ ,\hat{\phi}_{m-1,p})'$ and $\hat{\theta}_{m-1}=(\hat{\theta}_{m-1,1},\ \ldots\ ,\hat{\theta}_{m-1,q})'$. At iteration *m*, the second step is then to obtain new estimates of $\hat{\phi}_m$ and $\hat{\theta}_m$ for the updated series,$w_{m,t}$. To find the estimates $\hat{\phi}_m$ and $\hat{\theta}_m$, **REG_ARIMA** uses the exact likelihood method as described in Akaike, Kitagawa, Arahata and Tada (1979) and used in routine, **MAX_ARMA**.

Comments

When forecasts are requested (**MXLEAD** > 0), **REG_ARIMA** requires that future values of the independent variables are provided in optional argument **XLEAD**. In effect, **REG_ARIMA** assumes the future X's are known without error, which is valid for any deterministic function of time such as a seasonal indicator. Also, in economics, certain factors that are considered to be leading indicators are treated as deterministic for the purpose of predicting changes in the economy. Users may consider using a more general transfer function model if this is an unreasonable assumption. **REG_ARIMA** calculates forecast variances using the asymptotic result found in Fuller (1996) , Theorem 2.9.4. To obtain the standard errors of the ARMA parameters, **REG_ARIMA** calls routine **NSLSE** for the final *w* series.

Examples

Example 1

The data set consists of annual mileage per passenger vehicle and annual US population (in 1000's) spanning the years 1980 to 2006 (U.S. Energy Information Administration, 2008). Consider modeling the annual mileage using US population as a regression variable.

```
          use reg_arima_int
          use umach_int
          implicit none

          integer, parameter :: mxlead=5, idep=2
          integer :: i, nout, nobs, n
          integer :: imodel(3)=(/1,0,0/), indind(1)=(/1/)
          real(kind(1e0)) :: y(29), parma(2), xlead(mxlead, 2)
          real(kind(1e0)) :: preg(2), regses(2)
          real(kind(1e0)) :: ses(1), fcst(mxlead), fcstvar(mxlead)
          real(kind(1e0)) :: avar,llike
    !                               US mileage and population (1980-2008)
    !                               Source: U.S. Energy Information
    !                               Administration (Oct 2008).

          real(kind(1e0)) :: x(29,2)
          data x / &
              22722.4681, 22946.5714, 23166.4458, 23379.1990,           &
              23582.4902, 23792.3795, 24013.2887, 24228.8918,           &
              24449.8982, 24681.923, 24962.2814, 25298.0941, 25651.4224,&
              25991.8588, 26312.5820999999, 26627.8393, 26939.4284,     &
              27264.6925, 27585.4104, 27904.0168, 28217.1936,           &
              28503.9803, 28772.6647, 29021.0914, 29289.2127,           &
              29556.0549, 29836.2973, 30129.0332, 30405.9724,           &
              9062.0, 8813.0, 8873.0, 9050.0, 9118.0, 9248.0, 9419.0,   &
              9464.0, 9720.0, 9972.0, 10157.0, 10504.0, 10571.0,        &
              10857.0, 10804.0, 10992.0, 11203.0, 11330.0, 11581.0,     &
              11754.0, 11848.0, 11976.0, 11831.0, 12202.0, 12325.0,     &
              12460.0, 12510.0, 12485.0, 12293.0/

          call umach(2,nout)
```

```
!                                  Example 1
!                                  The first column is the scaled US
!                                  population and the second column is
!                                  the annual mileage per vehicle
      n =  size(x,1)
      nobs = n - mxlead
      call scopy(nobs,x(1:n,idep),1,y,1)
      call reg_arima(y, imodel ,parma, nobs=nobs, iprint=1, x=x,   &
               indx=indind, avar=avar, llike=llike, preg=preg,     &
               regse=regses, mxlead=mxlead, xlead=x(nobs+1:n,1:2), &
               trend=.true., fcst=fcst, fcstvar=fcstvar, se=ses)
      end
```

Output

```
 Final results for regarima model (p,d,q) =  1 0 0

  Final AR parameter estimates/ std errors

           0.73063        0.13499

 -2*ln(maximum log likelihood) =   231.8354

  White noise variance 10982.5654

  Regression estimates:

                 coef          reg. se
     1      -3481.22607      689.02661
     2          0.54237        0.02673

  Forecasts with standard deviation

  t         Y fcst          Y fcst std dev
  25      12360.40137         153.89659
  26      12514.61035         171.89198
  27      12673.53320         180.76634
  28      12837.36719         185.32974
  29      12991.26855         187.72037
```

Example 2

The data set consists of simulated weekly observations containing a strong annual seasonality. The seasonal variables are constructed and sent into REG_ARIMA as regression variables.

```
      use reg_arima_int
      use umach_int
      use const_int
      implicit none

      integer, parameter :: mxlead = 4
      integer :: nobs, i,n,idep,nout
      integer :: imodel(3) =(/2,0,0/)
      real(kind(1e0)) :: PI
      real(kind(1e0)) :: preg(3),regses(3),parma(3)
      real(kind(1e0)) :: ses(2),fcst(mxlead),fcstvar(mxlead)
      real(kind(1e0)) :: avar,llike,aic,tmpr
```

```
      real(kind(1e0)) :: x(100,2), xlead(mxlead,2)
      real(kind(1e0)) :: y(104) =(/ &
          32.27778,32.63300,33.13768,34.4517,34.63824, &
          37.31262,37.35704,37.03092,36.39894,35.75541,&
          35.10829,34.70107,34.69592,32.75326,30.85370,&
          31.10936,29.47493,29.14361,28.50466,30.09714,&
          28.49403,27.23268,23.49674,22.71225,21.42798,&
          18.68601,17.40035,16.06832,15.31862,14.75179,&
          13.40089,13.01101,12.44863,11.27890,11.51770,&
          14.31982,14.67036,14.76331,15.35644,17.04353,&
          18.39931,18.21919,18.72777,19.61794,22.31733,&
          23.79600,25.41326,25.60497,27.93579,29.21765,&
          29.60981,28.46994,28.78081,30.96402,35.49537,&
          35.75124,36.18933,37.2627,35.02454,33.57089, &
          35.00683,34.83886,34.19827,33.73966,34.49709,&
          34.07127,32.74709,31.97856,31.3029,30.21916, &
          27.46015,26.78431,25.32815,23.97863,21.83837,&
          21.00647,20.58846,19.94578,17.38271,17.12572,&
          16.71847,17.45425,16.15050,13.07448,12.54188,&
          12.42137,13.51771,14.84232,14.28870,13.39561,&
          15.48938,16.47175,17.62758,16.57677,18.20737,&
          20.8491,20.15616,20.93857,23.73973,25.30449, &
          26.51106,29.43261,32.02672,32.18846/)


      call umach(2,nout)
      PI = const('PI')

!                                The data are simulated weekly
!                                observations with an annual
!                                seasonal cycle
      n = size(y)
      nobs = n - mxlead
!                                Create the seasonal variables
      do i = 1, nobs
        x(i,1)=sin(2*PI*i/float(52))
        x(i,2) = cos(2*PI*i/float(52))
      end do
      do i=1, mxlead
        xlead(i,1)=sin(2*PI*(i+nobs)/float(52))
        xlead(i,2) = cos(2*PI*(i+nobs)/float(52))
      end do

      call reg_arima(y, imodel, parma, iprint=1, x=x,  &
              nobs=nobs, avar=avar, llike=llike,       &
              preg=preg, regse=regses, mxlead=mxlead,  &
              xlead=xlead, trend=.true., fcst=fcst,    &
              fcstvar=fcstvar, se=ses)
      end
```

Output

```
 Final results for regarima model (p,d,q) =  2 0 0

  Final AR parameter estimates/ std errors

           0.71860        0.09836

          -0.25991        0.09827
```

```
-2*ln(maximum log likelihood) =   -13.6209

 White noise variance     0.8783

 Regression estimates:

                coef          reg. se
     1         24.81010        0.17175
     2          9.68013        0.23993
     3          5.72305        0.24751

 Forecasts with standard deviation

 t        Y fcst          Y fcst std dev
101         26.74492          1.31358
102         28.07805          1.47616
103         29.33707          1.49560
104         30.53160          1.49560
```

GARCH

Computes estimates of the parameters of a GARCH(p,q) model.

Required Arguments

W — Vector of length NOBS containing the observed time series data. (Input)

NP — Number of GARCH parameters, p. (Input)

NQ — Number of ARCH parameters, q. (Input)

XGUESS — Vector of length NP+NQ +1 containing the initial values for the parameter vector X. (Input)

X — Vector of length NP+NQ+1 containing the estimates for σ^2, the ARCH parameters and the GARCH parameters. X(1) contains the estimate for σ^2, X(2)…X(NQ+1) contain the ARCH estimates, X(NQ+2)…X(NP+NQ+1) contain the GARCH estimates. (Output)

Optional Arguments

SIG2MAX— Upperbound for σ^2 , the first element of X. (Input)
Default: SIG2MAX = 10.

NOBS — Length of the observed time series. (Input)
Default: NOBS = size(W).

A — Value of Log-likelihood function evaluated at X. (Output)

AIC — Akaike's Information Criterion evaluated at X. (Output)

VAR — (NP+NQ+1) by (NP+NQ+1) matrix containing the variance-covariance matrix. (Output)

NDIM — Column dimension (NP+NQ+1) of VAR. (Input) Default: NDIM = NP+NQ+1.

FORTRAN 90 Interface

Generic: CALL GARCH (W, NP, NQ, XGUESS, X [, …])
Specific: The specific interface names are S_GARCH and D_GARCH.

Description

The Generalized Autoregressive Conditional Heteroskedastic (GARCH) model for a time series $\{w_t\}$ is defined as

$$w_t = z_t \sigma_t$$
$$\sigma_t^2 = \sigma^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i w_{t-i}^2,$$

where z_t's are independent and identically distributed standard normal random variables,

$$0 < \sigma^2 < SIG2MAX, \quad \beta_i \geq 0, \quad \alpha_i \geq 0 \quad \text{and}$$
$$\sum_{i=2}^{p+q+1} x(i) = \sum_{i=1}^{p} \beta_i + \sum_{i=1}^{q} \alpha_i < 1.$$

The above model is denoted as GARCH(p,q). The β_i and α_i coeffecients will be referred to as GARCH and ARCH coefficents, respectively. When $\beta_i = 0$, $i = 1,2,\ldots,p$, the above model reduces to ARCH(q) which was proposed by Engle (1982). The nonnegativity conditions on the parameters imply a nonnegative variance and the condition on the sum of the β_i's and α_i's is required for wide sense stationarity.

In the empirical analysis of observed data, GARCH(1,1) or GARCH(1,2) models have often found to appropriately account for conditional heteroskedasticity (Palm 1996). This finding is similar to linear time series analysis based on ARMA models.

It is important to notice that for the above models positive and negative past values have a symmetric impact on the conditional variance. In practice, many series may have strong asymmetric influence on the conditional variance. To take into account this phenomena, Nelson (1991) put forward Exponential GARCH (EGARCH). Lai (1998) proposed and studied some properties of a general class of models that extended linear relationship of the conditional variance in ARCH and GARCH into nonlinear fashion.

The maximum likelihood method is used in estimating the parameters in GARCH(p,q). The log-likelihood of the model for the observed series $\{w_t\}$ with length m = nobs is

$$\log(L) = -\frac{m}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{m} y_t^2/\sigma_t^2 - \frac{1}{2}\sum_{t=1}^{m}\log\sigma_t^2,$$
$$\text{where} \quad \sigma_t^2 = \sigma^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i w_{t-i}^2.$$

Thus log(L) is maximized subject to the constraints on the α_i, β_i, and σ.

In this model, if q = 0, the GARCH model is singular since the estimated Hessian matrix is singular.

The initial values of the parameter vector **x** entered in vector **xguess** must satisfy certain constraints. The first element of **xguess** refers to σ^2 and must be greater than zero and less than **sig2max**. The remaining **p+q** initial values must each be greater than or equal to zero and sum to a value less than one.

To guarantee stationarity in model fitting,

$$\sum_{i=2}^{p+q+1} x(i) = \sum_{i=1}^{p} \beta_i + \sum_{i=1}^{q} \alpha_i < 1$$

is checked internally. The initial values should selected from values between zero and one.

AIC is computed by

$$-2\log(L) + 2(p+q+1),$$

where log(*L*) is the value of the log-likelihood function.

In fitting the optimal model, the routine **NNLPF** as well as its associated subroutines are modified to find the maximum likelihood estimates of the parameters in the model. Statistical inferences can be performed outside the routine **GARCH** based on the output of the log-likelihood function **(A)**, the Akaike Information Criterion (**AIC**), and the variance-covariance matrix **(VAR)**.

Example

The data for this example are generated to follow a **GARCH**(2,1) process by using a standard normal random number generation routine **WG2RCH** . The data set is analyzed and estimates of sigma, the GARCH parameters, and the ARCH parameters are returned. The values of the Log-likelihood function and Akaike's Information Criterion are returned from the optional arguments **A** and **AIC**.

```
      USE GARCH_INT
      USE RNSET_INT
      IMPLICIT NONE

      INTERFACE
      SUBROUTINE WG2RCH (W, NP, NQ, NOBS, X, Z, Y0, SIGMA)
      INTEGER    NP, NQ, NOBS
      REAL(KIND(1D0))       W(:), X(:), Z(:), Y0(:), SIGMA(:)
      END SUBROUTINE
      END INTERFACE

      INTEGER :: NP, NQ, NOBS, N
      PARAMETER  (NP=2, NQ=1, NOBS=1000)
      PARAMETER  (N=NP+NQ+1)
      REAL(KIND(1D0)) :: A, AIC, Z(NOBS + 1000), Y0(NOBS + 1000), &
      X0=(/1.3,0.2,0.3,0.4/)
      XGUESS = (/1.0,0.1,0.2,0.3/)
      CALL RNSET (182198625)
      CALL WG2RCH (W, NP, NQ, NOBS, X0, Z, Y0, SIGMA)
```

```
      CALL GARCH(W, NP, NQ, XGUESS, X, NOBS=NOBS, A=A, AIC=AIC)
      WRITE(*,*)"Variance estimate is ", x(1)
      WRITE(*,*)"ARCH(1) estimate is ", x(2)
      WRITE(*,*)"GARCH(1) estimate is ", x(3)
      WRITE(*,*)"GARCH(2) estimate is ", x(4)
      WRITE(*,*)"Log-likelihood function is ", A
      WRITE(*,*)"Akaike's Information Criterion is ", AIC
      END

      SUBROUTINE WG2RCH (W, NP, NQ, NOBS, X, Z, Y0, SIGMA)
      USE RNNOR_INT
      INTEGER    NP, NQ, NOBS
      REAL(KIND(1D0))       W(:), X(:), Z(:), Y0(:), SIGMA(:)
      INTEGER    I, J, L
      REAL(KIND(1D0))       S1, S2, S3

!     RNNOR GENERATES STANDARD NORMAL OBSERVATIONS
      CALL RNNOR(Z, NOBS+1000)
!     INITIAL VALUES
      L = MAX(NP,NQ)
      L = MAX(L,1)
      DO I=1, L
         Y0(I) = Z(I)*X(1)
      END DO
!     COMPUTE THE INITIAL VALUE OF SIGMA
      S3 = 0.0;
      IF (MAX(NP,NQ) .GE. 1) THEN
         DO I=1, NP + NQ
            S3 = S3 + X(I+1)
         END DO
      END IF
      DO I=1, L
         SIGMA(I) = X(1)/(1.0-S3)
      END DO
      DO  I=L + 1, NOBS + 1000
         S1 = 0.0
         S2 = 0.0
         IF (NQ .GE. 1) THEN
            DO J=1, NQ
               S1 = S1 + X(J+1)*Y0(I-J)*Y0(I-J)
            END DO
         END IF
         IF (NP .GE. 1) THEN
            DO J=1, NP
               S2 = S2 + X(NQ+1+J)*SIGMA(I-J)
            END DO
         END IF
         SIGMA(I) = X(1) + S1 + S2
         Y0(I) = Z(I)*SQRT(SIGMA(I))
      END DO
! DISCARD THE FIRST 1000 SIMULATED OBSERVATIONS
      DO I=1, NOBS
         W(I) = Y0(1000+I)
      END DO
      RETURN
      END
```

Output

```
    Variance estimate is  1.6915576416511892
    ARCH(1) estimate is  0.24499571998823416
    GARCH(1) estimate is  0.3372325349834042
    GARCH(2) estimate is  0.3095905689822821
    Log-likelihood function is  -2707.072433499691
    Akaike's Information Criterion is  5422.144866999382
```

SPWF

Computes the Wiener forecast operator for a stationary stochastic process.

Required Arguments

W — Vector of length NOBS containing the stationary time series. (Input)

WNADJ — White noise adjustment factor. (Input)
WNADJ must be greater than or equal to zero.

EPS — Bound on the normalized mean square error. (Input)
EPS must be in the range (0, 1) inclusive.

MLFOP — Maximum length of the forecast operator. (Input)
MLFOP must be greater than or equal to one and less than NOBS.

LFOP — Length of the estimated forecast operator. (Output)

FOP — Vector of length LFOP containing the estimated forecast operator coefficients. (Output)

Optional Arguments

NOBS — Number of observations in the stationary time series W. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (W,1).

IWMEAN — Option for estimation of the mean of W. (Input)
Default: IWMEAN = 1.

IWMEAN	Action
0	WMEAN is user specified.
1	WMEAN is set equal to the arithmetic mean of W.

WMEAN — Estimate of the mean of the time series W. (Input, if IWMEAN = 0; output, if IWMEAN = 1)
WMEAN is used to center the time series W prior to estimation of the forecast operator.
Default: WMEAN = 0.0.

FORTRAN 90 Interface

Generic: CALL SPWF (W, WNADJ, EPS, MLFOP, LFOP, FOP [, ...])
Specific: The specific interface names are S_SPWF and D_SPWF.

FORTRAN 77 Interface

Single: CALL SPWF (NOBS, W, IWMEAN, WMEAN, WNADJ, EPS, MLFOP, LFOP, FOP)
Double: The double precision name is DSPWF.

Description

Routine SPWF performs least-squares estimation of parameters for successive autoregressive models of a stationary stochastic process given a sample of n = NOBS observations $\{W_t\}$ for $t = 1, \ldots, n$.

Let

$$\hat{\mu} = \text{WMEAN}$$

be the estimate of the mean μ of the stochastic process $\{W_t\}$ where

$$\hat{\mu} = \begin{cases} \mu & \mu \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} W_t & \mu \text{ unknown} \end{cases}$$

Consider the autoregressive model of order k defined by

$$\phi_k(B)\widetilde{W}_t = A_t \quad k \geq 0$$

where

$$\widetilde{W}_t = W_t - \hat{\mu}$$

and

$$\phi_k(B) = 1 - \phi_{1k}B - \phi_{2k}B^2 - \quad \cdots \quad - \phi_{kk}B^k \quad k \geq 1$$

Successive AR(k) models are fit to the centered data using Durbin's algorithm (1960) based on the sample autocovariances

$$\hat{\sigma}(k) = \frac{1}{n}\sum_{t=1}^{n-k}\left(W_t - \hat{\mu}\right)\left(W_{t+k} - \hat{\mu}\right) \quad k \geq 0$$

Note that the variance

$$\hat{\sigma}^*(0)$$

used in the fitting algorithm is adjusted by the amount δ = **WNADJ** according to

$$\hat{\sigma}^*(0) = (1+\delta)\hat{\sigma}(0)$$

See Robinson (1967, page 96).

Iteration to the next higher order model terminates when either the expected mean square error of the model is less than **EPS** or when *k* = **MLFOP**. The forecast operator $\phi = (\phi_1, \phi_2, \ldots, \phi_k^*)^T$ for *k** = **LFOP** is contained in **FOP**. See also Craddock (1969).

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2WF/DS2WF**. The reference is:

   ```
   CALL S2WF (NOBS, W, IWMEAN, WMEAN, WNADJ, EPS, MLFOP, LFOP, FOP, CW, WK)
   ```

   The additional arguments are as follows:

   CW — Vector of length **NOBS** containing the centered time series **W**. (Output)

   WK — Vector of length 2 * **MLFOP** + 1. (Output)

2. Informational error

Type	Code	Description
3	5	No operator could be found of length less than or equal to MLFOP that produced a normalized mean square error less than EPS.

3 The length of the forecast operator is determined by the arguments **EPS** and **MLFOP**. Iteration to a longer forecast operator stops when either the normalized mean square error is less than **EPS**, or the operator reaches the maximum allowable length, **MLFOP**.

4. The white noise adjustment factor, **WNADJ**, is used to modify the the estimate of the variance of the time series **W** used in the computation of the autocorrelation function of **W**. In the absence of white noise, **WNADJ** should be set to zero.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine SPWF to these data produces the following results:

```
      USE GDATA_INT
      USE SPWF_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    MLFOP, NOBS, NOUT, IMEAN, I
      PARAMETER  (MLFOP=1, NOBS=100)
!     INTEGER    I, IMEAN, LFOP, NCOL, NOUT, NROW

      REAL       EPS, FOP(MLFOP), RDATA(176,2), W(NOBS), WMEAN, WNADJ
      REAL       NROW, NCOL, LFOP
!
      EQUIVALENCE (W(1), RDATA(22,2))
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
      IMEAN = 0
      WMEAN = 46.976
!                                White noise adjustment
      WNADJ = 0.0
!                                Bound on normalized MSE
      EPS = 0.1
!                                Determine autoregressive model
      CALL SPWF (W, WNADJ, EPS, MLFOP, LFOP, FOP, IWMEAN=IMEAN, &
           WMEAN=WMEAN)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99997) LFOP
99997 FORMAT (/, 1X, 'Forecast operator length, LFOP = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (/, 1X, ' I           FOP(I)')
      DO 10  I=1, LFOP
         WRITE (NOUT,99999) I, FOP(I)
99999    FORMAT (1X, I2, 2X, F12.4)
   10 CONTINUE
!
      END
```

Output

```
*** WARNING  ERROR 5 from SPWF.  No operator could be found of length less
***          than or equal to 1 which produced a normalized mean square
***          error less than 1.000000E-01.

Forecast operator length, LFOP =  1

I           FOP(I)
1         0.8063
```

NSBJF

Computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model.

Required Arguments

W — Vector of length NOBS containing the time series. (Input)

PAR — Vector of length NPAR containing the autoregressive parameters. (Input)

LAGAR — Vector of length NPAR containing the order of the autoregressive parameters. (Input)
The elements of LAGAR must be greater than zero.

PMA — Vector of length NPMA containing the moving average parameters. (Input)

LAGMA — Vector of length NPMA containing the order of the moving average parameters. (Input)
The elements of LAGMA must be greater than zero.

ICNST — Option for including the overall constant in the model. (Input)

ICNST	Action
0	No overall constant is included.
1	The overall constant is included.

CNST — Estimate of the overall constant. (Input)

AVAR — Estimate of the random shock variance. (Input)
AVAR must be greater than 0.

ALPHA — Value in the exclusive interval (0, 1) used to specify the 100(1 - ALPHA)% probability limits of the forecasts. (Input)
Typical choices for ALPHA are 0.10, 0.05, and 0.01.

MXBKOR — Maximum backward origin. (Input)
MXBKOR must be greater than or equal to zero and less than or equal to NOBS - max(MAXAR, MAXMA) where MAXAR = max(LAGAR(*i*)) and MAXMA = max(LAGMA(*j*)). Forecasts at origins NOBS - MXBKOR through NOBS are generated.

MXLEAD — Maximum lead time for forecasts. (Input)
MXLEAD must be greater than zero.

FCST — MXLEAD by (MXBKOR + 3) matrix defined below. (Output)

Column	Content
j	Forecasts for lead times *l* = 1, ..., MXLEAD at origins NOBS - MXBKOR - 1 + *j*, *j* = 1, ..., MXBKOR + 1.
MXBKOR + 2	Deviations from each forecast that give the 100(1 - ALPHA)% probability limits.
MXBKOR + 3	Psi weights of the infinite order moving average form of the model.

Optional Arguments

NOBS — Number of observations in the time series W. (Input)
NOBS must be greater than ICONST + max(LAGAR(*i*)) + max(LAGMA(*j*)).
Default: NOBS = size (W,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the forecasts for lead times *l* = 1, ..., MXLEAD at each origin *t* = (NOBS - MXBKOR), ..., NOBS, the 100(1 - ALPHA)% probability limit deviations, and the psi weights.

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

LDFCST — Leading dimension of FCST exactly as specified in the dimension statement in the calling program. (Input)
LDFCST must be greater than or equal to MXLEAD.
Default: LDFCST = size (FCST,1).

FORTRAN 90 Interface

Generic: CALL NSBJF (W, PAR, LAGAR, PMA, LAGMA, ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST [, ...])
Specific: The specific interface names are S_NSBJF and D_NSBJF.

FORTRAN 77 Interface

Single: CALL NSBJF (NOBS, W, IPRINT, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST, LDFCST)

Double: The double precision name is DNSBJF.

Description

Routine NSBJF computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model given a sample of n = NOBS observations $\{W_t\}$ for t = 1, 2, ..., n.

Suppose the time series $\{W_t\}$ is generated by a nonseasonal ARMA model of the form

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \quad t \in \{0, \pm 1, \pm 2, \ldots\}$$

where B is the backward shift operator, θ_0 = CONST,

$$\phi(B) = 1 - \phi_1 B^{l_\phi(1)} - \phi_2 B^{l_\phi(2)} - \cdots - \phi_p B^{l_\phi(p)}$$
$$\theta(B) = 1 - \theta_1 B^{l_\theta(1)} - \theta_2 B^{l_\theta(2)} - \cdots - \theta_q B^{l_\theta(q)}$$

p = NPAR and q = NPMA. Without loss of generality, we assume

$$1 \le l_\phi(1) \le l_\phi(2) \le \cdots \le l_\phi(p)$$
$$1 \le l_\theta(1) \le l_\theta(2) \le \cdots \le l_\theta(q)$$

so that the nonseasonal ARMA model is of order (p', q') where $p' = l_\phi(p)$ and $q' = l_\theta(q)$. Note that the usual hierarchal model assumes

$$l_\phi(i) = i \quad 1 \le i \le p$$
$$l_\theta(j) = j \quad 1 \le j \le q$$

The Box-Jenkins forecast at origin t for lead time l of W_{t+l} is defined in terms of the difference equation

$$\hat{W}_t(l) = \theta_0 + \phi_1[W_{t+l-l_\phi(1)}] + \cdots + \phi_p[W_{t+l-l_\phi(p)}]$$
$$+[A_{t+l}] - \theta_1[A_{t+l-l_\theta(1)}] - \cdots - \theta_q[A_{t+l-l_\theta(q)}]$$

where

$$[W_{t+k}] = \begin{cases} W_{t+k} & k = 0, -1, -2, \ldots \\ \hat{W}_t(k) & k = 1, 2, \ldots \end{cases}$$

$$[A_{t+k}] = \begin{cases} W_{t+k} - \hat{W}_{t+k-1}(1) & k = 0, -1, -2, \ldots \\ 0 & k = 1, 2, \ldots \end{cases}$$

The 100(1 - α)% probability limits for W_{t+l} are given by

$$\hat{W}_t(l) \pm z_{\alpha/2}\left\{1 + \sum_{j=1}^{l-1} \psi_j^2\right\}^{1/2} \sigma_A$$

where $z_{a/2}$ is the 100(1 - α/2) percentile of the standard normal distribution,

$$\sigma_A^2 = \text{AVAR}$$

and { Ψ_j } are the parameters of the random shock form of the difference equation. Note that the forecasts are computed for lead times *l* = 1, 2, ..., *L* at origins *t* = (*n* - *b*), (*n* - *b* + 1), ..., *n* where *L* = MXLEAD and *b* = MXBKOR.

The Box-Jenkins forecasts minimize the mean square error

$$E[W_{t+l} - \hat{W}_t(l)]^2$$

Also, the forecasts may be easily updated according to the equation

$$\hat{W}_{t+1}(l) = \hat{W}_t(l+1) + \psi_l A_{t+1} \qquad (7)$$

This approach and others are given in Chapter 5 of Box and Jenkins (1976).

Comments

1. Workspace may be explicitly provided, if desired, by use of N2BJF/DN2BJF. The reference is:

   ```
   CALL N2BJF (NOBS, W, IPRINT, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, ICNST, CNST,
       AVAR, ALPHA, MXBKOR, MXLEAD, FCST, LDFCST, PARH, PMAH, PSIH, PSI, LAGPSI)
   ```

   The additional arguments are as follows:

   PARH — Work vector of length equal to IARDEG + 1.

   PMAH — Work vector of length equal to IMADEG + 1.

   PSIH — Work vector of length equal to MXLEAD + 1.

   PSI — Work vector of length equal to MXLEAD + 1.

   LAGPSI — Work vector of length equal to MXLEAD + 1.

2. If the **W** series has been transformed using a Box-Cox transformation with parameters **POWER** and **SHIFT**, the forecasts and probability limits for the original series may be obtained by application of routine **BCTR** with the same parameters and argument **IDIR** set equal to one.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Routine **NSBJF** is used to computed forecasts and 95% probability limits for the forecasts for an **ARMA**(2, 1) model fit using routine **NSPE**. With **MXBKOR** = 3, columns one through four of **FCST** give forecasts given the data through 1866, 1867, 1868, and 1869, respectively. Column 5 gives the deviations from the forecast for computing probability limits, and column six gives the psi weights, which can be used to update forecasts once more data is available. For example, the forecast for the 102-nd observation (year 1871) given the data through the 100-th observation (year 1869) is 77.21, and 95% probability limits are given by 77.21 ∓ 56.30. After observation 101 (W_{101} for year 1870) is available, the forecast can be updated by using equation 7 with the psi weight (Ψ_1 = 1.37) and the one-step-ahead forecast error for observation 101 (W_{101} - 83.72) to give

$$77.21 + 1.37\ (W_{101} - 83.72)$$

Since this updated forecast is one step ahead, the 95% probability limits are now given by the forecast ∓ 33.22.

```
      USE GDATA_INT
      USE NSPE_INT
      USE NSBJF_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    LDFCST, MXBKOR, MXLEAD, NOBS, NPAR, NPMA
      PARAMETER  (MXBKOR=3, MXLEAD=12, NOBS=100, NPAR=2, NPMA=1, &
                 LDFCST=MXLEAD)
!
      INTEGER    ICNST, LAGAR(NPAR), LAGMA(NPMA), NCOL, NROW
      REAL       ALPHA, AVAR, CNST, FCST(LDFCST,MXBKOR+3), PAR(NPAR), &
                 PMA(NPMA), RDATA(176,2), W(NOBS), WMEAN
      CHARACTER  CLABEL(MXBKOR+4)*40, RLABEL(1)*6
!
      EQUIVALENCE (W(1), RDATA(22,2))
!
      DATA LAGAR(1), LAGAR(2)/1, 2/
      DATA LAGMA(1)/1/
      DATA RLABEL/'NUMBER'/, CLABEL/'%/Lead%/Time', &
          '%/Forecast%/From 1866', '%/Forecast%/From 1867', &
          '%/Forecast%/From 1868', '%/Forecast%/From 1869', &
          ' Deviation %/  for 95%  %/Prob. Limits', '%/%/Psi'/
!                                 Wolfer Sunspot Data for
!                                 years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Compute preliminary parameter
!                                estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR)
```

```
!
!                                  Include constant in forecast model
      ICNST = 1
!                                  Specify 95 percent probability
!                                  limits for forecasts
      ALPHA = 0.05
!                                  Compute forecasts
      CALL NSBJF (W, PAR, LAGAR, PMA, LAGMA, &
                  ICNST, CNST, AVAR, ALPHA, MXBKOR, MXLEAD, FCST)
!                                  Print results
      CALL WRRRL ('FCST', FCST, RLABEL, CLABEL, FMT='(5F9.2, F6.3)')
!
      END
```

Output

```
                                  FCST
                                                          Deviation
 Lead    Forecast    Forecast    Forecast    Forecast      for 95%
 Time  From 1866   From 1867   From 1868   From 1869   Prob. Limits     Psi
  1       18.28       16.62       55.19       83.72          33.22   1.368
  2       28.92       32.02       62.76       77.21          56.30   1.127
  3       41.01       45.83       61.89       63.46          67.62   0.616
  4       49.94       54.15       56.46       50.10          70.64   0.118
  5       54.09       56.56       50.19       41.38          70.75  -0.208
  6       54.13       54.78       45.53       38.22          71.09  -0.326
  7       51.78       51.17       43.32       39.30          71.91  -0.286
  8       48.84       47.71       43.26       42.46          72.53  -0.169
  9       46.53       45.47       44.46       45.77          72.75  -0.045
 10       45.35       44.69       45.98       48.08          72.77   0.041
 11       45.21       44.99       47.18       49.04          72.78   0.077
 12       45.71       45.82       47.81       48.91          72.82   0.072
```

IRNSE

Computes estimates of the impulse response weights and noise series of a univariate transfer function model.

Required Arguments

X — Vector of length **NOBS** containing the input time series. (Input)

Y — Vector of length **NOBS** containing the output time series. (Input)

MWTIR — Maximum index of the impulse response weights. (Input)
MWTIR must be greater than or equal to zero and less than or equal to **NOBS** - 1.

MWTSN — Maximum index of the impulse response weights used to compute the noise series. (Input)
MWTSN must be greater than or equal to zero and less than or equal to **MWTIR**.

WTIR — Vector of length **MWTIR** + 1 containing the impulse response weight estimates. (Output)
The impulse response weight estimate of index *k* is given by **WTIR**(*k*) for
k = 0, 1, ..., **MWTIR**.

SNOISE — Vector of length **NOBS** - **MWTSN** containing the noise series based on the impulse response weight estimates. (Output)

XPW — Vector of length **NOBS** - **NPAR** containing the prewhitened input time series **X**. (Output)

YPW — Vector of length **NOBS** - **NPAR** containing the prewhitened output time series **Y**. (Output)

Optional Arguments

NOBS — Number of observations in each time series. (Input)
NOBS must be greater than or equal to two.
Default: **NOBS** = size (**X**,1).

IPRINT — Printing option. (Input)
Default: **IPRINT** = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the estimates of the impulse response weights and the noise series.

NPAR — Number of prewhitening autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1) if PAR is present otherwise NPAR = 0.

PAR — Vector of length NPAR containing the prewhitening autoregressive parameters. (Input)
Default: PAR = 0.0.

NPMA — Number of prewhitening moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1) if PAR is present, otherwise NPMA = 0.

PMA — Vector of length NPMA containing the prewhitening moving average parameters. (Input)
Default: PMA = 0.0.

FORTRAN 90 Interface

Generic: CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW [, ...])
Specific: The specific interface names are S_IRNSE and D_IRNSE.

FORTRAN 77 Interface

Single: CALL IRNSE (NOBS, X, Y, IPRINT, NPAR, PAR, NPMA, PMA, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW)
Double: The double precision name is DIRNSE.

Description

Routine IRNSE estimates the impulse response weights and noise series of a transfer function model given a sample of n = NOBS observations of the input $\{x_t\}$ and output $\{y_t\}$ for $t = 1, 2, \ldots, n$. Define $\{x_t\}$ and $\{y_t\}$, respectively, by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and output series may be obtained using the routine DIFF following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \nu(B)X_t + N_t$$

or, equivalently,

$$y_t = \nu(B)x_t + n_t$$

with transfer function

$$\nu(B) = \nu_0 + \nu_1 B + \nu_2 B^2 + \ldots$$

and differenced noise series $n_t = \nabla^d N_t$.

The prewhitened input and output series are computed for $t = (p + 1), \ldots, n$ according to

$$\alpha_t = \phi(B)x_t + \theta_1(B)\ \alpha_t$$
$$\beta_t = \phi(B)y_t + \theta_1(B)\ \beta_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$
$$\theta(B) = \theta_1 B + \theta_2 B^2 + \ldots + \theta_q B^q$$

The parameters of the prewhitening transformation may be estimated roughly using the routine NSPE or more precisely using the routine NSLSE. The correlation relationship between $\{\alpha_t\}$, $\{\beta_t\}$, and $\{n_t\}$ may be further examined using the routines ACF, PACF, and CCF.

The *impulse response weights* $\{\nu_k\}$ are estimated by

$$\hat{\nu}_k = \frac{\hat{\sigma}_\beta}{\hat{\sigma}_\alpha}\hat{\rho}_{\alpha\beta}(k) \quad k = 0, 1, \ldots, K$$

where K = MWTIR,

$$\hat{\sigma}_\alpha \text{ and } \hat{\sigma}_\beta$$

denote the standard deviation of $\{\alpha_t\}$ and $\{\beta_t\}$;

$$\hat{\rho}_{\alpha\beta}(k) \quad \text{for } k = 0, 1, \ \ldots, K$$

represents the cross-correlation function between $\{\alpha_t\}$ and $\{\beta_t\}$. The differenced noise series $\{n_t\}$ for $t = (K' + 1), \ldots, n$ is reconstructed using the model

$$n_t = y_t - \hat{v}_0 x_t - \hat{v}_1 x_{t-1} - \cdots - \hat{v}_{K'} x_{t-K'}$$

where K' = MWTSN.

Comments

1. Workspace may be explicitly provided, if desired, by use of I2NSE/DI2NSE. The reference is:

   ```
   CALL I2NSE (NOBS, X, Y, IPRINT, NPAR, PAR, NPMA, PMA, MWTIR, MWTSN, WTIR, SNOISE,
               XPW, YPW, ACPWX, ACPWY, CCPW)
   ```

   The additional arguments are as follows:

   ACPWX — Vector of length MWTIR + 1 containing the estimated autocorrelation function of PWX. (Output)

   ACPWY — Vector of length MWTIR + 1 containing the estimated autocorrelation function of PWY. (Output)

   CCPW — Vector of length 2 * MWTIR + 1 containing the estimated cross-correlation function of PWX and PWY. (Output)

2. The input series X and output series Y are assumed to be the result of transforming and differencing the raw input and output series. The routines BCTR and DIFF may be used, respectively, to perform a Box-Cox transformation and difference the raw input and output series.

3. Note that the prewhitened input and output are computed at time t = NPAR + 1 through t = NOBS. Also, the noise series is computed at time t = MWTSN + 1 through t = NOBS.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X is the input gas rate in cubic feet/minute and Y is the percent CO_2 in the outlet gas. Application of routine IRNSE to these data produces the following results:

```
      USE GDATA_INT
      USE IRNSE_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDX, MWTIR, MWTSN, NDX, NOBS, NOPRIN, NPAR, NPMA
      PARAMETER  (LDX=296, MWTIR=10, NDX=2, NOBS=296, &
                  NOPRIN=0, NPAR=3, NPMA=0, MWTSN=MWTIR)
```

```
!
      INTEGER    NCOL, NROW
      REAL       PAR(NPAR), PMA(1), RDATA(296,2), SNOISE(NOBS-MWTSN), &
                 WTIR(MWTIR+1), X(NOBS), XPW(NOBS-NPAR), Y(NOBS), &
                 YPW(NOBS-NPAR)
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Specify AR parameters for
!                                prewhitening transformation
      PAR(1) = 1.97
      PAR(2) = -1.37
      PAR(3) = 0.34
!                                Compute estimates of impulse
!                                response weights and noise series
      CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, SNOISE, XPW, YPW, PAR=PAR)
!                                Print results
      CALL WRRRN ('WTIR', WTIR, 1, 11, 1)
      CALL WRRRN ('SNOISE', SNOISE, 1, 20, 1)
!
      END
```

Output

```
                                WTIR
      1       2       3       4       5       6       7       8
-0.0355  0.0716 -0.0764 -0.5655 -0.6549 -0.8936 -0.5358 -0.3482

      9      10      11
-0.0782  0.0277 -0.1364

                               SNOISE
    1     2     3     4     5     6     7     8     9    10
53.21 53.49 53.72 54.05 53.98 53.95 53.69 53.02 52.56 52.33

   11    12    13    14    15    16    17    18    19    20
52.47 52.69 52.57 52.63 52.81 53.14 53.21 53.20 53.05 52.88
```

TFPE

more...

Computes preliminary estimates of parameters for a univariate transfer function model.

Required Arguments

NDELAY — Time delay parameter. (Input)
NDELAY must be greater than or equal to zero.

WTIR — Vector of length MWTIR + 1 containing the impulse response weight estimates. (Input)
The impulse response weight estimate of index *k* is given by WTIR(*k*) for *k* = 0, 1, ..., MWTIR.

SNOISE — Vector of length NSNOIS containing the noise series. (Input)

AVAR — Estimate of the random shock variance. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints estimates of transfer function parameters, estimates of noise model parameters, and an estimate of the random shock variance.

NPLHS — Number of left-hand side transfer function parameters. (Input)
NPLHS must be greater than or equal to zero.
Default: NPLHS = size (PLHS,1) if PLHS is present. Otherwise, NPLHS=0.

NPRHS — Number of right-hand side transfer function parameters (excluding the index 0 parameter). (Input)
NPRHS must be greater than or equal to zero.
Default: NPRHS = size (PRHS,1) – 1 if PRHS is present. Otherwise, NPRHS=0.

NPNAR — Number of noise autoregressive parameters. (Input)
NPNAR must be greater than or equal to zero.
Default: NPNAR = size (PNAR,1) if PNAR is present. Otherwise, NPNAR=0.

NPNMA — Number of noise moving average parameters. (Input)
NPNMA must be greater than or equal to zero.
Default: NPNMA = size (PNMA,1) if PNMA is present. Otherwise, NPNMA=0.

MWTIR — Maximum index of the impulse response weights. (Input)
MWTIR must be greater than or equal to NPLHS + NPRHS + NDELAY.
Default: MWTIR = size (WTIR,1) –1.

NSNOIS — Number of elements in the noise series. (Input)
NSNOIS must be greater than or equal to NPNAR + NPNMA + 1.
Default: NSNOIS = size (SNOISE,1).

RELERR — Stopping criterion for use in the nonlinear equation solver. (Input)
If RELERR = 0.0, then the default value RELERR = 100.0 * AMACH(4) is used. See the documentation for routine AMACH in the Reference Material section of this manual.
Default: RELERR = 0.0.

MAXIT — The maximum number of iterations allowed in the nonlinear equation solver. (Input)
If MAXIT = 0, then the default value MAXIT = 200 is used.
Default: MAXIT = 0.

PLHS — Vector of length NPLHS containing the estimates of the left-hand side transfer function parameters. (Output)
The LHS weight estimates are PLHS(*k*), *k* = 1, ..., NPLHS.

PRHS — Vector of length NPRHS + 1 containing the estimates of the right-hand side transfer function parameters. (Output)
The RHS weight estimates are PRHS(*k*), *k* = 0, ..., NPRHS.

PNAR — Vector of length NPNAR containing the estimates of the noise autoregressive parameters. (Output)

PNMA — Vector of length NPNMA containing the estimates of the noise moving average parameters. (Output)

FORTRAN 90 Interface

Generic: CALL TFPE (NDELAY, WTIR, SNOISE, AVAR [, ...])

Specific: The specific interface names are S_TFPE and D_TFPE.

FORTRAN 77 Interface

Single: CALL TFPE (IPRINT, NPLHS, NPRHS, NPNAR, NPNMA, NDELAY, MWTIR, WTIR, NSNOIS, SNOISE, RELERR, MAXIT, PLHS, PRHS, PNAR, PNMA, AVAR)

Double: The double precision name is DTFPE.

Description

Routine TFPE computes preliminary estimates of the parameters of a transfer function model given a sample of n = NOBS observations of the differenced input $\{x_t\}$ and differenced output $\{y_t\}$ for $t = 1, 2, \ldots, n$.

Define $\{x_t\}$ and $\{y_t\}$, respectively, by

$$x_t = \begin{cases} X_t - \hat{\mu}_X & d = 0 \\ \nabla^d X_t & d > 0 \end{cases}$$

and

$$y_t = \begin{cases} Y_t - \hat{\mu}_Y & d = 0 \\ \nabla^d Y_t & d > 0 \end{cases}$$

where $\{X_t\}$ and $\{Y_t\}$ for $t = (-d + 1), \ldots, n$ represent the undifferenced input and output series with

$$\hat{\mu}_X \text{ and } \hat{\mu}_Y$$

estimates of their respective means. The differenced input and output series may be obtained using the routine DIFF following any preliminary transformation of the data.

The transfer function model is defined by

$$Y_t = \delta^{-1}(B)\omega(B)X_{t-b} + N_t$$

or, equivalently,

$$y_t = \delta^{-1}(B)\omega(B)x_{t-b} + n_t$$

where $n_t = \nabla^d N_t$ and the left-hand side and right-hand side transfer function polynomial operators are

$$\delta(B) = 1 - \delta_1 B - \delta_2 B^2 - \ldots - \delta_r B^r$$
$$\omega(B) = \omega_0 - \omega_1 B - \omega_2 B^2 - \ldots - \omega_s B^s$$

with r = NPLHS, s = NPRHS, and b = NDELAY. The noise process $\{N_t\}$ and the input process $\{X_t\}$ are assumed to be independent with the noise process given by the ARIMA model

$$\phi(B) n_t = \theta(B) A_t$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$
$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

with p = NPNAR and q = NPNMA.

The impulse response weights and the transfer function parameters are related by

$$v_k = \begin{cases} 0 & k = 0, 1, \ldots, b-1 \\ \sum_{j=1}^{r} \delta_j v_{k-j} + \omega_0 & k = b \\ \sum_{j=1}^{r} \delta_j v_{k-j} - \omega_{k-b} & k = b+1, b+2, \ldots, b+s \\ \sum_{j=1}^{r} \delta_j v_{k-j} & k = b+s+1, b+s+2, \ldots \end{cases}$$

See Abraham and Ledolter (1983, page 341). The r left-hand side transfer function parameters are estimated using the difference equation given as the last case above. The resulting estimates

$$\hat{\delta}_1, \ldots, \hat{\delta}_r$$

are then substituted into the middle two cases to determine the s + 1 estimates

$$\hat{\omega}_0, \hat{\omega}_1, \ldots, \hat{\omega}_s$$

The noise series parameters are estimated using the routine NSPE. The impulse response weights $\{v_k\}$ and differenced noise series$\{n_t\}$ may be computed using the routine IRNSE. See Box and Jenkins (1976, pages 511–513).

Comments

1. Workspace may be explicitly provided, if desired, by use of T2PE/DT2PE. The reference is:

   ```
   CALL T2PE (IPRINT, NPLHS, NPRHS, NPNAR, NPNMA, NDELAY, MWTIR, WTIR, NSNOIS,
      SNOISE, RELERR, MAXIT, PLHS, PRHS, PNAR, PNMA, AVAR, A, FAC, IPVT, WK, ACV,
      PARWK, ACVMOD, TAUINI, TAU, FVEC, FJAC, R, QTF, WKNLN, H)
   ```

   The additional arguments are as follows:

A — Work vector of length (max(NPLHS, NPNAR))^2.

FAC — Work vector of length (max(NPLHS, NPNAR))^2.

IPVT — Work vector of length max(NPLHS, NPNAR).

WK — Work vector of length max(NPLHS, NPNAR).

ACV — Work vector of length NPNAR + NPNMA + 1.

PARWK — Work vector of length NPNAR + 1.

ACVMOD — Work vector of length NPNMA + 1.

TAUINI — Work vector of length NPNMA + 1.

TAU — Work vector of length NPNMA + 1.

FVEC — Work vector of length NPNMA + 1.

FJAC — Work vector of length (NPNMA + 1)^2.

R — Work vector of length (NPNMA + 1) * (NPNMA + 2)/2.

QTF — Work vector of length NPNMA + 1.

WKNLN — Work vector of length 5 * (NPNMA + 1).

H — Work vector of length NPLHS.

2. Informational error

Type	Code	Description
4	1	The nonlinear equation solver did not converge to RELERR within MAXIT iterations.

3. The impulse response weight estimates and the noise series may be computed using routine IRNSE.

Example

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where *X* is the input gas rate in cubic feet/minute and *Y* is the percent CO_2 in the outlet gas. The data is retrieved by routine GDATA. Routine IRNSE computes the impulse response weights. Application of routine TFPE to these weights produces the following results:

```
      USE GDATA_INT
      USE IRNSE_INT
      USE WROPT_INT
      USE TFPE_INT

      IMPLICIT   NONE
      INTEGER    MWTIR, MWTSN, NDELAY, NOBS, NPAR, NPLHS, NPMA, NPNAR, &
                 NPNMA, NPRHS, NSNOIS
      PARAMETER  (MWTIR=10, NDELAY=3, NOBS=100, NPAR=3, NPLHS=2, &
                 NPMA=0, NPNAR=2, NPNMA=0, NPRHS=2, MWTSN=MWTIR, &
                 NSNOIS=NOBS-MWTSN)
```

```
!
      INTEGER    IPRINT, ISETNG, NCOL, NROW
      REAL       AVAR, PAR(NPAR), PLHS(NPLHS), PMA(1), PNAR(NPNAR), &
                 PNMA(1), PRHS(NPRHS+1), RDATA(296,2), &
                 SNOISE(NOBS-MWTSN), WTIR(MWTIR+1), X(NOBS), &
                 XPW(NOBS-NPAR), Y(NOBS), YPW(NOBS-NPAR)
!
      EQUIVALENCE (X(1), RDATA(1,1)), (Y(1), RDATA(1,2))
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Specify AR parameters for
!                                prewhitening transformation
      PAR(1) = 1.97
      PAR(2) = -1.37
      PAR(3) = 0.34
!                                Compute estimates of impulse
!                                response weights and noise series
      CALL IRNSE (X, Y, MWTIR, MWTSN, WTIR, &
                  SNOISE, XPW, YPW, PAR=PAR)
!                                Convergence parameters
!                                Compute preliminary estimates of
!                                transfer function parameters
      ISETNG = 1
      CALL WROPT (-6, ISETNG, 1)
      IPRINT = 1
      CALL TFPE (NDELAY, WTIR, SNOISE, AVAR, IPRINT=IPRINT, NPLHS=NPLHS, &
                 NPRHS=NPRHS, NPNAR=NPNAR, PLHS=PLHS, PRHS=PRHS, PNAR=PNAR)
!
      END
```

Output

```
 PLHS from TFPE/T2PE
        1             2
 0.120342      0.326149

         PRHS from TFPE/T2PE
         1             2             3
 -0.623240      0.318698      0.362488

 PNAR from TFPE/T2PE
       1             2
 1.64679      -0.70916

 PNMA is not written since NPNMA = 0

 AVAR from TFPE/T2PE =     2.85408E-02
```

MLSE

more...

Computes least-squares estimates of a linear regression model for a multichannel time series with a specified base channel.

Required Arguments

X — NOBSX by NCHANX matrix containing the time series. (Input)
Each row of X corresponds to an observation of a multivariate time series, and each column of X corresponds to a univariate time series. The base time series or output channel is contained in the first column.

NDIFF — Vector of length NCHANX containing the order of differencing for each channel of X. (Input)
The elements of NDIFF must be greater than or equal to zero.

NDPREG — Vector of length NCHANX containing the number of regression parameters in the differenced form of the model for each channel of X. (Input) The elements of NDPREG must be greater than or equal to zero.

LAG — Vector of length NCHANX containing the amount of time that each channel of X is to lag the base series. (Input)
The elements of LAG must be greater than or equal to zero.

CNST — Estimate of the overall constant. (Output)

NPREG — Number of regression parameters in the undifferenced model. (Output)

NPREG = IADD + (NDPREG(1) + NDIFF(1)) + … + (NDPREG(NCHANX) + DIFF(NCHANX)

where

IADD = NDIFF(1), if NDPREG(1) = 0

IADD = max (0, min(LAG(1) - 1, NDIFF(1))), if NDPREG(1) > 0.

PREG — Vector of length NPREG containing the regression parameters in the undifferenced model. (Output)

The parameter estimates are concatenated as follows.

Channel 1: REG(*i*), *i* = 1, 2, ..., IADD + NDPREG(1) + NDIFF(1)

Channel *j*: PREG(*i*), *i* = I(*j*) + 1, I(*j*) + 2, ..., I(*j*) + NDPREG(*j*) + NDIFF(*j*)

where

I(*j*) = IADD + NDPREG(1) + NDIFF(1) + ... + NDPREG(*j* - 1) + NDIFF(*j* - 1)
for *j* = 2, 3, ..., NCHANX.

Optional Arguments

NOBSX — Number of observations in each channel of the time series X. (Input)
NOBSX must be less than or equal LDX and greater than max(NDPREG(*i*) + LAG(*i*)) for *i* = 1, 2, ..., NCHANX.
Default: NOBSX = size (X,1).

NCHANX — Number of channels in the time series X. (Input)
NCHANX must be greater than or equal to one.
Default: NCHANX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

IMEAN — Option for computation of the means of the channels of X. (Input)
Default: IMEAN = 1.

IMEAN	Action
0	XMEAN is user specified.
1	XMEAN is set to the vector of arithmetic means of the channels of X.

XMEAN — Vector of length NCHANX containing the means of the channels of X. (Input, if IMEAN = 0; output, if IMEAN = 1)

FORTRAN 90 Interface

Generic: CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG [, ...])
Specific: The specific interface names are S_MLSE and D_MLSE.

FORTRAN 77 Interface

Single: CALL MLSE (NOBSX, NCHANX, X, LDX, IMEAN, XMEAN, NDIFF, NDPREG, LAG, CNST, NPREG, PREG)

Double: The double precision name is DMLSE.

Description

Routine MLSE performs least-squares estimation of a linear regression model for a multichannel time series with a specified base channel.

Define the multichannel time series *X* by

$$X = (X_1, X_2, \ldots, X_m)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T \quad j = 1, 2, \ldots, m$$

with *n* = NOBSX and *m* = NCHANX. The columns of *X* correspond to individual channels of a multichannel time series and may be examined from a univariate perspective. The rows of *X* correspond to observations of an *m*-variate time series and may be examined from a multivariate perspective. Note that an alternative characterization of the multivariate time series *X* considers the columns of *X* to be observations of an *m*-variate time series with the rows of *X* containing univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

The model is formed by regressing the base series X_1 on its previous values and on the remaining channels $X_2, \ldots, X_m$. The differenced form of the model is given by

$$X_{t1} = \theta_0 + \phi_1(B)\nabla^{d_1}B^{l_1}X_{t1} + \phi_2(B)\nabla^{d_2}B^{l_2}X_{t2} + \ldots + \phi_m(B)\nabla^{d_m}B^{l_m}X_{tm}$$

where θ_0 = CNST is the overall constant, d_k = NDIFF(*k*) is the order of differencing X_k, l_k = LAG(*k*) is the amount X_k lags X_1,

$$\phi_k(B) = \phi_{1k} + \phi_{2k}B + \ldots + \phi_{p_k,k}B^{p_k-1}$$

and p_k = NDPREG(*k*) for *k* = 1, 2, ..., *m*.

The undifferenced form of the model is given by

$$X_{t1} = \theta_0 + \varphi_1(B)X_{t-l_1,1} + \varphi_2(B)X_{t-l_2,2} + \ \ldots \ + \varphi_m(B)X_{t-l_m,m}$$

where the undifferenced parameters $\varphi_k = \mathrm{PREG}(k)$ are defined by

$$\begin{aligned}\varphi_k(B) &= \phi_k(B)\nabla^{d_k} \\ &= \varphi_1 + \varphi_2 B + \ \cdots \ + \varphi_{p_k+d_k} B^{p_k+d_k-1}\end{aligned}$$

for k = 1, 2, ..., m. Note that if $l_1 \geq d_1 \geq 0$, the base series terms $X_{t-j,1}$ at lags j = 1, ..., (l_1 - 1) are omitted from the right-hand side of the above model when $d_1 \geq 1$. In the actual computations, these terms are included.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2SE/DM2SE**. The reference is:

   CALL M2SE (**NOBSX**, **NCHANX**, **X**, **LDX**, **IMEAN**, **XMEAN**, **NDIFF**, **NDPREG**, **LAG**, **CNST**, **NPREG**, **PREG**, **XWK**, **IWK**)

   The additional arguments are as follows:

   XWK — Work vector of length **NOBSX** * **NCHANX**+ 2 * **NSUM**^{2}+ max(**IADD**, **NCHANX** + **NSUM**), where **NSUM** = **NDPREG**(1) + … + **NDPREG**(**NCHANX**).

   IWK — Work vector of length **NSUM**.

2. Prior to parameter estimation, the channels of **X** are centered and/or differenced according to **XMEAN** and **NDIFF**, respectively.

3. The undifferenced predictive form of the model is

   X(*t*, 1) = **CNST** + **PREG**(1) * **X**(*t* - 1, 1) + … + **PREG**(**IADD**) * **X**(*t* - **IADD**, 1) + **PREG**(**IADD** + 1) * **X**(*t* - **LAG**(1), 1) + … + **PREG**(**IADD** + **NDPREG**(1) + **NDIFF**(1)) * **X**(*t* - **LAG**(1) + 1 - **NDPREG**(1) - **NDIFF**(1), 1) + … + **PREG**(*l*(*j*) + 1) * **X**(*t* - **LAG**(*j*), *j*) + … + **PREG**(*l*(*j*)+**NDPREG**(*j*)+**NDIFF**(*j*)) * **X**(*t* - **LAG**(*j*) + 1 - **NDPREG**(*j*) - **NDIFF**(*j*),*j*) + …

   where

   l(*j*) = **IADD** + **NDPREG**(1) + **NDIFF**(1) + … + **NDPREG**(*j* - 1) + **NDIFF**(*j* - 1)

   for *j* = 2, 3, ..., **NCHANX**.

Examples

Example 1

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light activity and earthquake activity (Robinson 1967, page 204) to be a three-channel time series. Routine MLSE is applied to these data to examine the regressive relationship between the channels.

```
      USE GDATA_INT
      USE MLSE_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LDX, NCHANX, NOBSX
      PARAMETER  (NCHANX=3, NOBSX=100, LDX=NOBSX)
!
      INTEGER    I, IMEAN, LAG(NCHANX), NCOL, NDIFF(NCHANX), &
                 NDPREG(NCHANX), NOUT, NPREG, NROW
      REAL       CNST, PREG(20), RDATA(100,4), X(LDX,NCHANX), &
                 XMEAN(NCHANX)
!
      EQUIVALENCE (X(1,1), RDATA(1,2)), (X(1,2), RDATA(1,3)), &
                (X(1,3), RDATA(1,4))
!
      DATA NDIFF(1), NDIFF(2), NDIFF(3)/1, 1, 0/
      DATA LAG(1), LAG(2), LAG(3)/1, 2, 1/
      DATA NDPREG(1), NDPREG(2), NDPREG(3)/2, 1, 3/
!
      CALL GDATA (8, RDATA, NROW, NCOL)
!     USE Default Option to estimate channel means
!                                Compute regression parameters
      CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG, XMEAN=XMEAN)
!
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99993)
99993 FORMAT (//, 1X, '     Results of MLSE/M2SE')
      WRITE (NOUT,99994)
99994 FORMAT (1X, '  I   NDIFF(I)  LAG(I)  NDPREG(I)     XMEAN(I)')
      DO 10  I=1, NCHANX
         WRITE (NOUT,99995) I, NDIFF(I), LAG(I), NDPREG(I), XMEAN(I)
99995    FORMAT (1X, 4(I3,6X), F12.4)
   10 CONTINUE
      WRITE (NOUT,99996) CNST
99996 FORMAT (1X, 'Overall constant, CNST = ', F12.4)
      WRITE (NOUT,99997) NPREG
99997 FORMAT (//, 1X, 'Total number of parameters, NPREG = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (//, 1X, ' I          PREG(I)')
      DO 20  I=1, NPREG
         WRITE (NOUT,99999) I, PREG(I)
99999    FORMAT (1X, I2, 5X, F12.4)
   20 CONTINUE
!
      END
```

Output

```
Results of MLSE/M2SE
I   NDIFF(I)  LAG(I)  NDPREG(I)     XMEAN(I)
1        1        1        2          46.9400
2        1        2        1          63.4300
3        0        1        3          97.9700
Overall constant, CNST =      -7.2698

Total number of parameters, NPREG =  8

I          PREG(I)
1          -0.1481
2          -1.3444
3           0.4925
4          -0.0302
5           0.0302
6          -0.0210
7           0.0187
8           0.0765
```

Example 2

Consider the Gas Furnace Data (Box and Jenkins 1976, pages 532–533) where X_1 is the percent CO_2 in the outlet gas and X_2 is the input gas rate in cubic feet/minute. Application of routine MLSE to these data produces the following results:

```
      USE GDATA_INT
      USE SCOPY_INT
      USE MLSE_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LDX, NCHANX, NOBSX
      PARAMETER  (NCHANX=2, NOBSX=296, LDX=NOBSX)
!
      INTEGER    I, IMEAN, LAG(NCHANX), NCOL, NDIFF(NCHANX), &
                 NDPREG(NCHANX), NOUT, NPREG, NROW
      REAL       CNST, PREG(20), RDATA(296,2), X(LDX,NCHANX), &
                 XMEAN(NCHANX)
!
      DATA NDIFF(1), NDIFF(2)/0, 0/
      DATA LAG(1), LAG(2)/1, 3/
      DATA NDPREG(1), NDPREG(2)/2, 3/
!                                Gas Furnace Data
      CALL GDATA (7, RDATA, NROW, NCOL)
!                                Multichannel X consists of
!                                Column 1: Output percent CO2
!                                Column 2: Input gas rate
      CALL SCOPY (NOBSX, RDATA(1:,2), 1, X(1:,1), 1)
      CALL SCOPY (NOBSX, RDATA(1:,1), 1, X(1:,2), 1)
!                                Option to estimate channel means
      IMEAN = 1
!                                Compute regression parameters
      CALL MLSE (X, NDIFF, NDPREG, LAG, CNST, NPREG, PREG, XMEAN=XMEAN)
```

```
!
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99993)
99993 FORMAT (1X, 'Results of MLSE/M2SE on Gas Furnace Data')
      WRITE (NOUT,99994)
99994 FORMAT (1X, '  I   NDIFF(I)  LAG(I)  NDPREG(I)       XMEAN(I)')
      DO 10  I=1, NCHANX
         WRITE (NOUT,99995) I, NDIFF(I), LAG(I), NDPREG(I), XMEAN(I)
99995    FORMAT (1X, 4(I3,6X), F12.4)
   10 CONTINUE
      WRITE (NOUT,99996) CNST
99996 FORMAT (1X, 'Overall constant, CNST = ', F12.4)
      WRITE (NOUT,99997) NPREG
99997 FORMAT (1X, 'Total number of parameters, NPREG = ', I2)
      WRITE (NOUT,99998)
99998 FORMAT (1X, ' I          PREG(I)')
      DO 20  I=1, NPREG
         WRITE (NOUT,99999) I, PREG(I)
99999    FORMAT (1X, I2, 5X, F12.4)
   20 CONTINUE
!
      END
```

Output

```
Results of MLSE/M2SE on Gas Furnace Data
I   NDIFF(I)  LAG(I)  NDPREG(I)       XMEAN(I)
1        0        1        2           53.5091
2        0        3        3           -0.0568
Overall constant, CNST =       2.6562
Total number of parameters, NPREG =  5
I          PREG(I)
1           1.6063
2          -0.6561
3          -0.4837
4          -0.1653
5           0.5052
```

MWFE

more...

Computes least-squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

Required Arguments

CXX — Array of size NCHX by NCHX by MLFIL containing the autocovariances of the input time series X. (Input)

CZX — Array of size NCHZ by NCHX by MLFIL containing the cross-covariances between the desired output time series Z and the input time series X. (Input)

EPS — Lower bound for the normalized mean square error. (Input)

TRACE — Trace of the autocovariance matrix of the desired output time series Z at time lag zero. (Input)

LFIL — Length of the Wiener filter. (Output)

FIL — Array of size NCHZ by NCHX by MLFIL containing the multichannel Wiener filter coefficients. (Output)

ENMS — Vector of length MLFIL containing the normalized mean square error corresponding to each filter length. (Output)

Optional Arguments

NCHX — Number of input channels. (Input)
NCHX must be greater than or equal to one.
Default: NCHX = size (CXX,1).

MLFIL — Maximum length of the Wiener filter. (Input)
MLFIL must be greater than or equal to one.
Default: MLFIL = size (CXX,3).

LDCXX — Leading dimension of CXX exactly as specified in the dimension statement of the calling program. (Input)
LDCXX must be greater than or equal to NCHX.
Default: LDCXX = size (CXX,1).

MDCXX — Middle dimension of CXX exactly as specified in the dimension statement of the calling program. (Input)
MDCXX must be greater than or equal to NCHX.
Default: MDCXX = size (CXX,2).

NCHZ — Number of channels in desired output time series. (Input)
NCHZ must be greater than or equal to one.
Default: NCHZ = size (CZX,1).

LDCZX — Leading dimension of CZX exactly as specified in the dimension statement of the calling program. (Input)
LDCZX must be greater than or equal to NCHZ.
Default: LDCZX = size (CZX,1).

MDCZX — Middle dimension of CZX exactly as specified in the dimension statement of the calling program. (Input)
MDCZX must be greater than or equal to NCHX.
Default: MDCZX = size (CZX,2).

LDFIL — Leading dimension of FIL exactly as specified in the dimension statement of the calling program. (Input)
LDFIL must be greater than or equal to NCHZ.
Default: LDFIL = size (FIL,1).

MDFIL — Middle dimension of FIL exactly as specified in the dimension statement of the calling program. (Input)
MDFIL must be greater than or equal to NCHX.
Default: MDFIL = size (FIL,2).

FORTRAN 90 Interface

Generic: CALL MWFE (CXX, CZX, EPS, TRACE, LFIL, FIL, ENMS [, ...])
Specific: The specific interface names are S_MWFE and D_MWFE.

FORTRAN 77 Interface

Single: CALL MWFE (NCHX, MLFIL, CXX, LDCXX, MDCXX, NCHZ, CZX, LDCZX, MDCZX, EPS, TRACE, LFIL, FIL, LDFIL, MDFIL ENMS)

Double: The double precision name is **DMWFE**.

Description

Routine **MFFE** computes least-squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

Define the multichannel time series *X* by

$$X = (X_1, X_2, \ldots, X_p)$$

where

$$X_j = (X_{1j}, X_{2j}, \ldots, X_{nj})^T \qquad j = 1, 2, \ldots, p$$

with *p* = **NCHX**. Similarly, define the multichannel time series *Z* by

$$Z = (Z_1, Z_2, \ldots, Z_q)$$

where

$$Z_j = (Z_{1j}, Z_{2j}, \ldots, Z_{mj})^T \qquad j = 1, 2, \ldots, q$$

with *q* = **NCHZ**. The columns of *X* and *Z* correspond to individual channels of multichannel time series and may be examined from a univariate perspective. The rows of *X* and *Z* correspond to observations of *p*-variate and *q*-variate time series and may be examined from a multivariate perspective. Note that an alternative characterization of a multivariate time series *X* considers the columns of *X* to be observations of a *p*-variate time series with the rows of *X* containing univariate time series. For example, see Priestley (1981, page 692) and Fuller (1976, page 14).

Let

$$\hat{\mu}_X$$

be the row vector containing the means of the channels of *X*. In particular,

$$\hat{\mu}_X = (\hat{\mu}_{X_1}, \hat{\mu}_{X_2}, \ldots, \hat{\mu}_{X_p})$$

where for *j* = 1, 2, ..., *p*

$$\hat{\mu}_{X_j} = \begin{cases} \mu_{X_j} & \mu_{X_j} \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_{tj} & \mu_{X_j} \text{ unknown} \end{cases}$$

Let

$$\hat{\mu}_Z$$

be similarly defined. In what follows, assume the channels of both *X* and *Z* have been centered about their respective means

$$\hat{\mu}_X \text{ and } \hat{\mu}_Z$$

Suppose the desired output is the multichannel time series *Z* defined by the model

$$Z_{t\bullet} = X_{t\bullet}\phi_0 + X_{(t-1)\bullet}\phi_1 + \ \ldots \ + X_{(t-K)\bullet}\phi_K$$

where

$$X_{t\bullet} = (X_{t1}, X_{t2}, \ \ldots, X_{tp})$$
$$Z_{t\bullet} = (Z_{t1}, Z_{t2}, \ \ldots, Z_{tp})$$

and $\boldsymbol{\phi}_k$ is the array of dimension $p \times q$ containing the Wiener filter coefficients

$$\phi_k = \begin{bmatrix} \phi_{11k} & \phi_{12k} & \cdots & \phi_{1qk} \\ \phi_{21k} & \phi_{22k} & \cdots & \phi_{2qk} \\ \vdots & \vdots & & \vdots \\ \phi_{p1k} & \phi_{p2k} & \cdots & \phi_{pqk} \end{bmatrix}$$

for $k = 1, \ldots, K$. The array $\boldsymbol{\phi}_k$ is the $(k + 1)$-st level of the 3-dimensional array **FIL**.

The filter coefficients are computed by solving a set of normal equations. The algorithm utilizes the block Toeplitz (or Töplitz) matrix structure of these equations and is given by Robinson (1967, pages 238–246). In particular, the required input consists of the multichannel autocovariance matrices $\boldsymbol{\Sigma}_X$, $\boldsymbol{\Sigma}_Z$, and the multichannel cross-covariance matrix $\boldsymbol{\Sigma}_{ZX}$. The routine **MCCF** may be used to estimate these covariance matrices.

Note that successively longer filters are estimated until either the normalized mean square error is less than **EPS** or the filter length K = **LFIL** equals **MLFIL**. The normalized mean square error is defined by

$$Q_k = 1 - \frac{\text{tr}\sum_{k=0}^{K} \sum_{ZX}(k)\phi_k}{\text{tr}\sum_Z(0)}$$

where tr $\boldsymbol{\Sigma}_Z(0)$ = **TRACE** is the trace of the multichannel autocorrelation coefficient of the desired output at lag zero. The values of Q_k for the successive filters of length $k = 1, 2, \ldots, K$ are contained in **ENMS**.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2FE/DM2FE**. The reference is:

```
CALL M2FE (NCHX, MLFIL, CXX, LDCXX, MDCXX, NCHZ, CZX, LDCZX, MDCZX, EPS, TRACE,
    LFIL, FIL, LDFIL, MDFIL, ENMS, IWK, WK)
```

The additional arguments are as follows:

IWK — Work vector of length NCHX.

WK — Work vector of length NCHX * NCHX * (2 * MLFIL + 12) + NCHZ.

2. The length of the filter is determined by the arguments EPS and MLFIL. Iteration to a longer filter stops when either the normalized mean square error ENMS is less than EPS, or the filter reaches the maximum allowable length, MLFIL.

3. The routine MCCF may be used to obtain the input arguments CXX, CZX, and TRACE. For TRACE, routine MCCF may be used to obtain the autocovariances of the desired output series Z. In particular, TRACE = ZVAR(1) + … + ZVAR(NCHZ).

4. For a given lag *k*, the multichannel cross-covariance coefficient between Z and X is defined as the array of size NCHZ by NCHX whose elements are the single-channel crosscovariance coefficients CZX(*i*, *j*, *k*).

Example

Consider the Wölfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light activity and earthquake activity (Robinson 1967, page 204) to be a three-channel time series. Routine MWFE applied to these data determines the following Wiener filter:

```
      USE GDATA_INT
      USE SCOPY_INT
      USE MCCF_INT
      USE MWFE_INT
      USE UMACH_INT

      IMPLICIT   NONE

      INTEGER    LDCXX, LDCZX, LDFIL, LDX, LDZ, MAXLAG, MDCXX, &
                 MDCZX, MDFIL, MLFIL, NCHANX, NCHANZ, NOBSX, NOBSZ
      REAL       SSUM
      PARAMETER  (MLFIL=3, NCHANX=3, NCHANZ=3, NOBSX=99, NOBSZ=99, &
                  LDCXX=NCHANX, LDCZX=NCHANZ, LDFIL=NCHANX, LDX=NOBSX, &
                  LDZ=NOBSZ, MAXLAG=MLFIL-1, MDCXX=NCHANX, MDCZX=NCHANX, &
                  MDFIL=NCHANZ)
!
      INTEGER    I, J, K, LFIL, NCOL, NOUT, NROW
      REAL       CVXX(LDCXX,MDCXX,-MAXLAG:MAXLAG), CVXX1(3,3,3), &
                 CVZX(LDCZX,MDCZX,-MAXLAG:MAXLAG), CVZX1(3,3,3), &
                 CXX(LDCXX,MDCXX,-MAXLAG:MAXLAG), &
                 CZX(LDCZX,MDCZX,-MAXLAG:MAXLAG), ENMS(MLFIL), EPS, &
                 FIL(LDFIL,MDFIL,MLFIL), R(0:2), RDATA(100,4), &
                 TRACE, X(LDX,NCHANX), XMEAN(NCHANX), XVAR(NCHANX), &
                 YMEAN, YVAR, Z(LDZ,NCHANZ), ZMEAN(NCHANZ), &
                 ZVAR(NCHANZ)
```

```
!
      EQUIVALENCE (CVXX(1,1,0), CVXX1(1,1,1)), (CVZX(1,1,0), CVZX1(1,1, &
                1))
!                                  Wolfer sunspot numbers
!                                  Northern lights activity
!                                  Earthquake activity
      CALL GDATA (8, RDATA, NROW, NCOL)
!
      CALL SCOPY (NOBSX, RDATA(1:,2), 1, X(1:,1), 1)
      CALL SCOPY (NOBSX, RDATA(1:,3), 1, X(1:,2), 1)
      CALL SCOPY (NOBSX, RDATA(1:,4), 1, X(1:,3), 1)
!
      CALL SCOPY (NOBSZ, RDATA(2:,2), 1, Z(1:,1), 1)
      CALL SCOPY (NOBSZ, RDATA(2:,3), 1, Z(1:,2), 1)
      CALL SCOPY (NOBSZ, RDATA(2:,4), 1, Z(1:,3), 1)
!                                  Compute multichannel ACF of Z
      CALL MCCF (Z, Z, MAXLAG, CXX, XVAR=XVAR, CCV=CVXX)
!                                  Compute TRACE
      TRACE = SSUM(NCHANZ,XVAR,1)
!                                  Compute multichannel ACF of X
      CALL MCCF (X, X, MAXLAG, CXX, CCV=CVXX)
!                                  Compute multichannel CCF of Z and X
      CALL MCCF (Z, X, MAXLAG, CZX, CCV=CVZX)
!                                  Bound normalized MSE to be positive
      EPS = 0.0
!                                  Reverse the LAG direction and scale
!                                  to agree with Robinson (1967)
      R(0)  = 99.D0
      R(1)  = 98.D0
      R(2)  = 97.D0
      TRACE = TRACE*R(0)
      DO 10  K=0, MAXLAG
         DO 10  J=1, NCHANX
            DO 10  I=1, NCHANX
               CVXX(I,J,K) = CVXX(I,J,-K)*R(K)
               CVZX(I,J,K) = CVZX(I,J,-K)*R(K)
   10 CONTINUE
!                                  Compute multichannel Wiener filter
      CALL MWFE (CVXX1, CVZX1, EPS, TRACE, LFIL, FIL, ENMS)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99994) LFIL
99994 FORMAT (1X, 'Number of filter coefficients, LFIL = ', I3)
      DO 30  K=1, LFIL
         WRITE (NOUT,99995) K
99995    FORMAT (//, 1X, 'Wiener filter coefficient of index K = ', I3)
         DO 20  I=1, NCHANX
            WRITE (NOUT,99996) (FIL(I,J,K),J=1,NCHANZ)
99996       FORMAT (1X, 3F12.4)
   20    CONTINUE
   30 CONTINUE
      WRITE (NOUT,99997)
99997 FORMAT (//, 1X, 'Normalized mean square error')
      WRITE (NOUT,99998)
99998 FORMAT (1X, ' K          ENMS(K)')
      DO 40  K=1, LFIL
         WRITE (NOUT,99999) K, ENMS(K)
99999    FORMAT (1X, I2, 5X, F12.4)
   40 CONTINUE
```

```
!
      END
```

Output

```
 Number of filter coefficients, LFIL =   3

 Wiener filter coefficient of index K =   1
  1.3834      0.0348      0.0158
  0.0599      0.8266      0.0629
 -0.1710     -0.0332     -0.1205

 Wiener filter coefficient of index K =   2
  -0.7719     -0.0183     -0.0318
 -0.0040     -0.2328      0.0484
 -0.2170      0.1912     -0.0667

 Wiener filter coefficient of index K =   3
  0.0516      0.0563     -0.0138
 -0.0568      0.1084     -0.1731
  0.0007      0.2177     -0.0152

 Normalized mean square error
  K          ENMS(K)
  1           0.6042
  2           0.5389
  3           0.5174
```

KALMN

more...

Performs Kalman filtering and evaluates the likelihood function for the state-space model.

Required Arguments

Y — Vector of length **NY** containing the observations. (Input)

Z — **NY** by **NB** matrix relating the observations to the state vector in the observation equation. (Input)

R — **NY** by **NY** matrix such that **R** * σ^2 is the variance-covariance matrix of errors in the observation equation. (Input)
σ^2 is a positive unknown scalar. Only elements in the upper triangle of **R** are referenced.

B — Estimated state vector of length **NB**. (Input/Output)
The input is the estimated state vector at time *k* given the observations thru time *k* - 1. The output is the estimated state vector at time *k* + 1 given the observations thru time *k*. On the first call to **KALMN**, the input **B** must be the prior mean of the state vector at time 1.

COVB — **NB** by **NB** matrix such that **COVB** * σ^2 is the mean squared error matrix for **B**. (Input/Output)
Before the first call to **KALMN**, **COVB** * σ^2 must equal the variance-covariance matrix of the state vector.

N — Rank of the variance-covariance matrix for all the observations. (Input/Output)
N must be initialized to zero before the first call to **KALMN**. In the usual case when the variance-covariance matrix is nonsingular, **N** equals the sum of the **NY**'s from the invocations to **KALMN**.

SS — Generalized sum of squares. (Input/Output)
SS must be initialized to zero before the first call to **KALMN**. The estimate of σ^2 is given by **SS**/**N**.

ALNDET — Natural log of the product of the nonzero eigenvalues of *P* where *P* * σ^2 is the variance-covariance matrix of the observations. (Input/Output)
Although **ALNDET** is computed, **KALMN** avoids the explicit computation of *P*. **ALNDET** must be initialized to zero before the first call to **KALMN**. In the usual case when *P* is nonsingular, **ALNDET** is the natural log of the determinant of *P*.

Optional Arguments

NY — Number of observations for current update. (Input)
If NY = 0, no update is performed.
Default: NY = size (Y,1).

NB — Number of elements in the state vector. (Input)
Default: NB = size (Z,2).

LDZ — Leading dimension of Z exactly as specified in the dimension statement in the calling program. (Input)
Default: LDZ = size (Z,1).

LDR — Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

IT — Transition matrix option. (Input)
Default: IT = 1.

IT	Action
0	T is the transition matrix in the state equation.
1	The identity is the transition matrix in the state equation.

T — NB by NB transition matrix in the state equation. (Input, if IT = 0)
If IT = 1, then T is not referenced and can be a vector of length one.

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

IQ — State equation error option. (Input)
Default: IQ = 1.

IQ	Action
0	There is an error term in the state equation.
1	There is no error term in the state equation.

Q — NB by NB matrix such that Q * σ^2 is the variance-covariance matrix of the error vector in the state equation. (Input, if IQ = 0)
σ^2 is a positive unknown scalar. If IQ = 1, then Q is not referenced and can be a 1x1 array. If IQ = 0, only the elements in the upper triangle of Q are referenced.

LDQ — Leading dimension of **Q** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDQ** = size (**Q**,1).

TOL — Tolerance used in determining linear dependence. (Input)
TOL = 100.0 * **AMACH**(4) is a common choice. See the documentation for routine **AMACH** in the Reference Material.
Default: **TOL** = 1.e-5 for single precision and 2.d –14 for double precision.

LDCOVB — Leading dimension of **COVB** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOVB** = size (**COVB**,1),

V — Vector of length **NY** containing the one-step-ahead prediction error. (Output)
If **Y** is not needed, then **V** and **Y** can occupy the same storage locations.

COVV — **NY** by **NY** matrix such that **COVV** * σ^2 is the variance-covariance matrix of **V**. (Output)
If **R** is not needed, then **COVV** and **R** can occupy the same storage locations.

LDCOVV — Leading dimension of **COVV** exactly as specified in the dimension statement in the calling program. (Input)

FORTRAN 90 Interface

Generic: CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET [, ...])
Specific: The specific interface names are S_KALMN and D_KALMN.

FORTRAN 77 Interface

Single: CALL KALMN (NY, Y, NB, Z, LDZ, R, LDR, IT, T, LDT, IQ, Q, LDQ, TOL, B, COVB, LDCOVB, N, SS, ALNDET, V, COVV, LDCOVV)
Double: The double precision name is DKALMN.

Description

Routine **KALMN** is based on a recursive algorithm given by Kalman (1960), which has come to be known as the Kalman filter. The underlying model is known as the state-space model. The model is specified stage by stage where the stages generally correspond to time points at which the observations become available. The routine **KALMN** avoids many of the computations and storage requirements that would be necessary if one were to process all the data at the end of each stage in order to estimate the state vector. This is accomplished by using previous computations and retaining in storage only those items essential for processing of future observations.

The notation used here follows that of Sallas and Harville (1981). Let y_k (input in **Y**) be the $n_k \times 1$ vector of observations that become available at time k. The subscript k is used here rather than t, which is more customary in time series, to emphasize that the model is expressed in stages $k = 1, 2, \ldots$ and that these stages need not correspond to equally spaced time points. In fact, they need not correspond to time points of any kind. The *observation equation* for the state-space model is

$$y_k = Z_k b_k + e_k \qquad k = 1, 2, \ldots$$

Here, Z_k (input in **Z**) is an $n_k \times q$ known matrix and b_k is the $q \times 1$ state vector. The state vector b_k is allowed to change with time in accordance with the *state equation*

$$b_{k+1} = T_{k+1} b_k + w_{k+1} \quad k = 1, 2, \ldots$$

starting with $b_1 = \boldsymbol{\mu}_1 + w_1$.

The change in the state vector from time k to $k + 1$ is explained in part by the *transition matrix* T_{k+1}(input in **T**), which is assumed known. It is assumed that the q-dimensional w_k's ($k = 1, 2,$).are independently distributed multivariate normal with mean vector 0 and variance-covariance matrix $\boldsymbol{\sigma}^2 Q_k$, that the n_k-dimensional e_k's ($k = 1, 2,$).are independently distributed multivariate normal with mean vector 0 and variance-covariance matrix $\boldsymbol{\sigma}^2 R_k$, and that the w_k's and e_k's are independent of each other. Here, $\boldsymbol{\mu}_1$ is the mean of b_1 and is assumed known, $\boldsymbol{\sigma}^2$ is an unknown positive scalar. Q_{k+1} (input in **Q**) and R_k (input in **R**) are assumed known.

Denote the estimator of the realization of the state vector b_k given the observations $y_1, y_2, \ldots, y_j$ by

$$\hat{\beta}_{k|j}$$

By definition, the mean squared error matrix for

$$\hat{\beta}_{k|j}$$

is

$$\sigma^2 C_{k|j} = E(\hat{\beta}_{k|j} - b_k)(\hat{\beta}_{k|j} - b_k)^T$$

At the time of the k-th invocation, we have

$$\hat{\beta}_{k|k-1}$$

and $C_{k|k-1}$, which were computed from the (k - 1)-st invocation, input in **B** and **COVB**, respectively. During the k-th invocation, routine **KALMN** computes the filtered estimate

$$\hat{\beta}_{k|k}$$

along with $C_{k|k}$. These quantities are given by the *update equations*:

$$\hat{\beta}_{k|k} = \hat{\beta}_{k|k-1} + C_{k|k-1} Z_k^T H_k^{-1} v_k$$
$$C_{k|k} = C_{k|k-1} - C_{k|k-1} Z_k^T H_k^{-1} Z_k C_{k|k-1}$$

where

$$v_k = y_k - Z_k \hat{\beta}_{k|k-1}$$

and where

$$H_k = R_k + Z_k C_{k|k-1} Z_k^T$$

Here, v_k (stored in **V**) is the one-step-ahead prediction error, and $\sigma^2 H_k$ is the variance-covariance matrix for v_k. H_k is stored in **COVV**. The "start-up values" needed on the first invocation of **KALMN** are

$$\hat{\beta}_{1|0} = \mu_1$$

and $C_1|_0 = Q_1$ input via **B** and **COVB**, respectively. Computations for the *k*-th invocation are completed by **KALMN** computing the one-step-ahead estimate

$$\hat{\beta}_{k+1|k}$$

along with $C_{k+1|k}$ given by the *prediction equations*:

$$\hat{\beta}_{k+1|k} = T_{k+1} \hat{\beta}_{k|k}$$
$$C_{k+1|k} = T_{k+1} C_{k|k} T_{k+1}^T + Q_{k+1}$$

If both the filtered estimates and one-step-ahead estimates are needed by the user at each time point, **KALMN** can be invoked twice for each time point—first with **IT** = 1 and **IQ** = 1 to produce

$$\hat{\beta}_{k|k}$$

and $C_{k|k}$, and second with **NY** = 0 to produce

$$\hat{\beta}_{k+1|k}$$

and $C_{k+1|k}$ (With **IT** = 1 and **IQ** = 1, the prediction equations are skipped. With **NY** = 0, the update equations are skipped.)

Often, one desires the estimate of the state vector more than one-step-ahead, i.e., an estimate of

$$\hat{\beta}_{k|j}$$

is needed where $k > j + 1$. At time j, **KALMN** is invoked to compute

$$\hat{\beta}_{j+1|j}$$

Subsequent invocations of **KALMN** with **NY** = 0 can compute

$$\hat{\beta}_{j+2|j}, \hat{\beta}_{j+3|j}, \ \ldots, \hat{\beta}_{k|j}$$

Computations for

$$\hat{\beta}_{k|j}$$

and $C_{k|j}$ assume the variance-covariance matrices of the errors in the observation equation and state equation are known up to an unknown positive scalar multiplier, σ^2. The maximum likelihood estimate of σ^2 based on the observations $y_1, y_2, \ldots, y_m$, is given by

$$\hat{\sigma}^2 = SS / N$$

where

$$N = \sum_{k=1}^{m} n_k \text{ and } SS = \sum_{k=1}^{m} v_k^T H_k^{-1} v_k$$

If σ^2 is known, the R_k's and Q_k's can be input as the variance-covariance matrices exactly. The earlier discussion is then simplified by letting $\sigma^2 = 1$.

In practice, the matrices T_k, Q_k, and R_k are generally not completely known. They may be known functions of an unknown parameter vector θ. In this case, **KALMN** can be used in conjunction with an optimization program (see routine **UMINF**, (IMSL MATH/LIBRARY)) to obtain a maximum likelihood estimate of θ. The natural logarithm of the likelihood function for $y_1, y_2, \ldots, y_m$ differs by no more than an additive constant from

$$L(\theta,\sigma^2;y_1,y_2,\ \ldots\ ,y_m) = -\frac{1}{2}N\ln\sigma^2 - \frac{1}{2}\sum_{k=1}^{m}\ln[\det(H_k)] - \frac{1}{2}\sigma^{-2}\sum_{k=1}^{m}v_k^T H_k^{-1}v_k$$

(Harvey 1981, page 14, equation 2.21). Here,

$$\sum_{k=1}^{m}\ln[\det(H_k)]$$

(stored in **ALNDET**) is the natural logarithm of the determinant of V where $\sigma^2 V$ is the variance-covariance matrix of the observations.

Minimization of $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ over all θ and σ^2 produces maximum likelihood estimates. Equivalently, minimization of $-2L_c(\theta; y_1, y_2, \ldots, y_m)$ where

$$L_c(\theta;y_1,\ y_2,\ \ldots\ ,y_m) = -\frac{1}{2}N\ln\left(\frac{SS}{N}\right) - \frac{1}{2}\sum_{k=1}^{m}\ln[\det(H_k)]$$

produces maximum likelihood estimates

$$\hat{\theta} \quad \text{and} \quad \hat{\sigma}^2 = SS/N$$

The minimization of $-2L_c(\theta; y_1, y_2, \ldots, y_m)$ instead of $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$, reduces the dimension of the minimization problem by one. The two optimization problems are equivalent since

$$\hat{\sigma}^2(\theta) = SS(\theta)/N$$

minimizes $-2L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ for all θ, consequently,

$$\hat{\sigma}^2(\theta)$$

can be substituted for σ^2 in $L(\theta, \sigma^2; y_1, y_2, \ldots, y_m)$ to give a function that differs by no more than an additive constant from $L_c(\theta; y_1, y_2, \ldots, y_m)$.

The earlier discussion assumed H_k to be nonsingular. If H_k is singular, a modification for singular distributions described by Rao (1973, pages 527–528) is used. The necessary changes in the preceding discussion are as follows:

1. Replace

   $$H_k^{-1}$$

   by a generalized inverse.

2. Replace det(H_k) by the product of the nonzero eigenvalues of H_k.

3. Replace N by

$$\sum_{k=1}^{m} \text{rank}\left(H_k\right)$$

Maximum likelihood estimation of parameters in the Kalman filter is discussed by Sallas and Harville (1988) and Harvey (1981, pages 111–113).

Comments

1. Workspace may be explicitly provided, if desired, by use of **K2LMN/DK2LMN**. The reference is:

```
CALL K2LMN (NY, Y, NB, Z, LDZ, R, LDR, IT, T, LDT, IQ, Q, LDQ, TOL, B, COVB, LDCOVB, N, SS,
    ALNDET, V, COVV, LDCOVV, COVVCH, WK1, WK2)
```

   The additional arguments are as follows.

   COVVCH — Work vector of length **NY * NY** containing the Cholesky factor of the **COVV** matrix. If **R** and **COVV** are not needed, **COVVCH**, **R**, and **COVV** can occupy the same storage locations and **LDR** must equal **LDCOVV**.

   WK1 — Work vector of length **NB * NB**.

   WK2 — Work vector of length **NB * NY** + max(**NB**, **NY**).

2. Informational errors

Type	Code	Description
4	1	R + Z * COVB * Z^T is not nonnegative definite within the tolerance defined by TOL. Either TOL is too small, or R or COVB is not nonnegative definite.
4	2	The system of equations $COVVCH^T * x = V$ is inconsistent. The variance-covariance matrix of the observations is inconsistent with the observations input in Y.
4	3	The system of equations $COVVCH^T * x = Z * COVB$ is inconsistent. The Cholesky factorization to compute COVVCH may be based on too large a value for TOL. The input of a smaller value for TOL may be appropriate.

3. If **R**, **Q**, and **T** are known functions of unknown parameters, **KALMN** can be used in conjunction with routine **UMINF** (IMSL MATH/LIBRARY) to perform maximum likelihood estimation of these unknown parameters. **UMINF** should be used to minimize the function

$$\text{N * ALOG(SS/N) + ALNDET}$$

4. In order to maintain acceptable numerical accuracy, the double precision version of **KALMN** is usually required.

Examples

Example 1

Routine **KALMN** is used to compute the filtered estimates and one-step-ahead estimates for a scalar problem discussed by Harvey (1981, pages 116–117). The observation equation and state equation are given by

$$y_k = b_k + e_k$$
$$b_{k+1} = b_k + w_{k+1} \quad k = 1,2,3,4$$

where the e_k's are identically and independently distributed normal with mean 0 and variance σ^2, the w_k's are identically and independently distributed normal with mean 0 and variance $4\sigma^2$, and b_1 is distributed normal with mean 4 and variance $16\sigma^2$. Two invocations of **KALMN** are needed for each time point in order to compute the filtered estimate and the one-step-ahead estimate. The first invocation uses Default: **IQ** = 1 and **IT** = 1 so that the prediction equations are skipped in the computations. The second invocation uses **NY** = 0 so that the update equations are skipped in the computations.

This example also computes the one-step-ahead prediction errors. Harvey (1981, page 117) contains a misprint for the value v_4 that he gives as 1.197. The correct value of v_4 = 1.003 is computed by **KALMN**.

```
      USE UMACH_INT
      USE KALMN_INT

      IMPLICIT   NONE
      INTEGER    LDCOVB, LDCOVV, LDQ, LDR, LDT, LDZ, NB, NOBS, NY
      PARAMETER  (NB=1, NOBS=4, NY=1, LDCOVB=NB, LDCOVV=NY, LDQ=NB, &
                 LDR=NY, LDT=NB, LDZ=NY)
!
      INTEGER    I, IQ, IT, N, NOUT
      REAL       ALNDET, B(NB), COVB(LDCOVB,NB), &
                 COVV(LDCOVV,NY), Q(LDQ,NB), R(LDR,NY), SS, T(LDT,NB), &
                 V(NY), Y(NY), YDATA(NOBS), Z(LDZ,NB)
!
      DATA YDATA/4.4, 4.0, 3.5, 4.6/, Z/1.0/, R/1.0/, Q/4.0/, T/1.0/
!
      CALL UMACH (2, NOUT)
!                                Initial estimates for state vector
!                                and variance-covariance matrix.
!                                Initialize SS and ALNDET.
      B(1)      = 4.0
      COVB(1,1) = 16.0
      N         = 0
      SS        = 0.0
      ALNDET    = 0.0
      WRITE (NOUT,99998)
!
      DO 10  I=1, NOBS
!                                Update
         Y(1) = YDATA(I)
         CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, T=T, Q=Q, V=V, &
```

```
                        COVV=COVV)
            WRITE (NOUT,99999) I, I, B(1), COVB(1,1), N, SS, ALNDET, &
                               V(1), COVV(1,1)
!                                    Prediction
            IQ = 0
            IT = 0
            CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, NY=0, IT=IT, T=T, &
                        IQ=IQ, Q=Q, V=V, COVV=COVV)
            WRITE (NOUT,99999) I + 1, I, B(1), COVB(1,1), N, SS, ALNDET, &
                               V(1), COVV(1,1)
      10 CONTINUE
99998 FORMAT (' k/j', '   B    ', '  COVB  ', ' N', '   SS   ', &
              ' ALNDET ', '   V    ', '  COVV  ')
99999 FORMAT (I2, '/', I1, 2F8.3, I2, 4F8.3)
      END
```

Output

```
 k/j    B      COVB   N   SS    ALNDET    V      COVV
 1/1   4.376   0.941 1   0.009   2.833   0.400  17.000
 2/1   4.376   4.941 1   0.009   2.833   0.400  17.000
 2/2   4.063   0.832 2   0.033   4.615  -0.376   5.941
 3/2   4.063   4.832 2   0.033   4.615  -0.376   5.941
 3/3   3.597   0.829 3   0.088   6.378  -0.563   5.832
 4/3   3.597   4.829 3   0.088   6.378  -0.563   5.832
 4/4   4.428   0.828 4   0.260   8.141   1.003   5.829
 5/4   4.428   4.828 4   0.260   8.141   1.003   5.829
```

Example 2

Routine **KALMN** is used with routine **UMINF** (IMSL MATH/LIBRARY) to find a maximum likelihood estimate of the parameter θ in a MA(1) time series represented by $y_k = \varepsilon_k - \theta\varepsilon_{k-1}$. Routine **RNARM** (see Chapter 18, "Random Number Generation") is used to generate 200 random observations from an MA(1) time series with $\theta = 0.5$ and $\sigma^2 = 1$.

The MA(1) time series is cast as a state-space model of the following form (see Harvey 1981, pages 103–104, 112):

$$y_k = \begin{pmatrix} 1 & 0 \end{pmatrix} b_k$$

$$b_k = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} b_{k-1} + w_k$$

where the two-dimensional w_k's are independently distributed bivariate normal with mean 0 and variance $\sigma^2 Q_k$ and

$$Q_1 = \begin{pmatrix} 1+\theta^2 & -\theta \\ -\theta & \theta^2 \end{pmatrix}$$

$$Q_k = \begin{pmatrix} 1 & -\theta \\ -\theta & \theta^2 \end{pmatrix} \qquad k = 2,\ 3,\ \ldots,\ 200$$

The warning error that is printed as part of the output is not serious and indicates that **UMINF** is generally used for multi-parameter minimization.

```
      USE RNSET_INT
      USE RNARM_INT
      USE UMINF_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS, NTHETA
      PARAMETER  (NOBS=200, NTHETA=1)
!
      INTEGER    IADIST, IPARAM(7), ISEED, LAGAR(1), LAGMA(1), NOUT, &
                 NPAR, NPMA
      REAL       A(NOBS+1), AVAR, CNST, FSCALE, FVALUE, PAR(1), &
                 PMA(1), RPARAM(7), THETA(NTHETA), WI(1), XGUESS(1), &
                 XSCALE(1), YDATA(NOBS)
      COMMON     /MA1/ YDATA
      EXTERNAL   FCN
!
      ISEED = 123457
      CALL RNSET (ISEED)
      PMA(1)   = 0.5
      LAGMA(1) = 1
      CNST    = 0.0
      NPAR     = 0
      NPMA     = 1
      IADIST   = 0
      AVAR     = 1.0
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
                  IADIST, AVAR, A, WI, YDATA, NPAR=NPAR)
!                                 Use UMINF to find maximum likelihood
!                                 estimate of the MA parameter THETA.
      CALL UMINF (FCN, THETA)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) '* * * Final Estimate for THETA * * *'
      WRITE (NOUT,*) 'Maximum likelihood estimate, THETA = ', THETA(1)
      END
!                                Use KALMN to evaluate the likelihood.
      SUBROUTINE FCN (NTHETA, THETA, FUNC)
      USE KALMN_INT
      INTEGER    NTHETA
      REAL       THETA(NTHETA), FUNC
!
      INTEGER    LDCOVB, LDCOVV, LDQ, LDR, LDT, LDZ, NB, NOBS, NY
      PARAMETER  (NB=2, NOBS=200, NY=1, LDCOVB=NB, LDCOVV=NY, LDQ=NB, &
                 LDR=NY, LDT=NB, LDZ=NY)
!
```

```
      INTEGER    I, IQ, IT, N
      REAL       ABS, ALNDET, ALOG, B(NB), COVB(LDCOVB,NB), &
                 COVV(LDCOVV,NY), Q(LDQ,NB), R(LDR,NY), SS, T(LDT,NB), &
                 TOL, V(NY), Y(NY), YDATA(NOBS), Z(LDZ,NB)
      COMMON     /MA1/ YDATA
      INTRINSIC  ABS, ALOG
!
      DATA T/0.0, 0.0, 1.0, 0.0/, Z/1.0, 0.0/
!
      IF (ABS(THETA(1)) .GT. 1.0) THEN
!                                 Estimate out of parameter space.
!                                 Set function to a large number.
         FUNC = 1.E10
         RETURN
      END IF
      IQ     = 0
      Q(1,1) = 1.0
      Q(1,2) = -THETA(1)
      Q(2,1) = -THETA(1)
      Q(2,2) = THETA(1)**2
      IT     = 0
!                                 No error in the
!                                 observation equation.
      R(1,1) = 0.0
!                                 Initial estimates for state vector
!                                 and variance-covariance matrix.
!                                 Initialize SS and ALNDET.
      B(1)      = 0.0
      B(2)      = 0.0
      COVB(1,1) = 1.0 + THETA(1)**2
      COVB(1,2) = -THETA(1)
      COVB(2,1) = -THETA(1)
      COVB(2,2) = THETA(1)**2
      N         = 0
      SS        = 0.0
      ALNDET    = 0.0
!
      DO 10  I=1, NOBS
         Y(1) = YDATA(I)
         CALL KALMN (Y, Z, R, B, COVB, N, SS, ALNDET, IT=IT, T=T, &
                     IQ=IQ, Q=Q)
   10 CONTINUE
      FUNC = N*ALOG(SS/N) + ALNDET
      RETURN
      END
```

Output

```
*** WARNING  ERROR 1 from U5INF.  This routine may be inefficient for a
***          problem of size N = 1.
  Here is a traceback of subprogram calls in reverse order:
  Routine name                  Error type  Error code
  ------------                  ----------  ----------
   U5INF                             6           1    (Called internally)
   U3INF                             0           0    (Called internally)
   U2INF                             0           0    (Called internally)
   UMINF                             0           0
   USER                              0           0
* * * Final Estimate for THETA * * *
```

```
    Maximum likelihood estimate, THETA =    0.452944
```

AUTO_UNI_AR

more...

Automatic selection and fitting of a univariate autoregressive time series model. The lag for the model is automatically selected using Akaike's Information Criterion (AIC). Estimates of the autoregressive parameters for the model with minimum AIC are calculated using method of moments or maximum likelihood.

Required Arguments

W — Vector containing the stationary time series. (Input)

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NPAR — Number of autoregressive parameters. (Output)

PAR — Vector of length MAXLAG, which contains the estimates for the autoregressive parameters in the model with the minimum AIC. The estimates are in the first NPAR values of this vector. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

IMETH — Estimation method option. (Input)

0	Method of moments
1	Maximum likelihood
2	Method of least-squares

Default: IMETH = 0.

MAXIT — Maximum number of estimation iterations. (Input)
Default: MAXIT = 500.

AVAR — Innovation variance. (Output)

AIC — Akaike's Information Criterion. (Output)

FORTRAN 90 Interface

Generic: CALL AUTO_UNI_AR (W, MAXLAG, NPAR, PAR [, ...])
Specific: The specific interface names are S_AUTO_UNI_AR and D_AUTO_UNI_AR.

Description

The routine AUTO_UNI_AR automatically selects the order of the AR model that best fits the data and then computes the AR coefficients. The algorithm used in AUTO_UNI_AR is derived from the work of Akaike, H., et. al (1979) and Kitagawa & Akaike (1978). This code was adapted from the UNIMAR procedure published as part of the TIMSAC-78 Library.

The best-fit AR model is determined by successively fitting AR models with 1, 2, ..., MAXLAG autoregressive coefficients. For each model, Akaike's Information Criterion (AIC) is calculated based on the formula:

$$AIC = -2\ln(likelihood) + 2(NPAR)$$

AUTO_UNI_AR uses the approximation to this formula developed by Ozaki and Oda (1979).

$$AIC = (NOBS - MAXLAG)\ln(\hat{\sigma}^2) + 2(NPAR) + (NOBS - MAXLAG)(\ln(2\pi) + 1)$$

The best fit model is the model with minimum AIC. If the number of parameters in this model is equal to the highest order autoregressive model fitted, i.e., NPAR=MAXLAG, then a model with smaller AIC might exist for larger values of MAXLAG. In this case, increasing MAXLAG to explore AR models with additional autoregressive parameters might be warranted.

If IMETH = 0, estimates of the autoregressive coefficients for the model with the minimum AIC are calculated using method of moments. If IMETH = 1, the model with the minimum AIC is identified and coefficients are then estimated using maximum likelihood. Otherwise, if IMETH = 2, the coefficients for the model with minimum AIC are computed using the method of least-squares.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed for each year from 1770 through 1869. In this example, **AUTO_UNI_AR** found the minimum AIC fit is an autoregressive model with 10 lags:

$$w_t = \phi_0 + \phi_1 w_{t-1} + \ \ldots \ + \phi_{10} w_{t-10} + a_t$$

using the formula

$$\phi_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right)$$

the lag 10 AR model for this series can be represented as:

$$\begin{aligned} w_t = {} & 0.85\ M + 1.24\ w_{t-1} - 0.50\, w_{t-2} - 0.16\ w_{t-3} \\ & + 0.23\, w_{t-4} \ - 0.20 w_{t-5} + 0.11 w_{t-6} \\ & - 0.08 w_{t-7} + 0.09 w_{t-8} + 0.01 w_{t-9} + 0.10 w_{t-10} + a_t \end{aligned}$$

```
      use auto_uni_ar_int
      use wrrrn_int
      use gdata_int
      implicit none
!                                  SPECIFICATIONS FOR PARAMETERS
      integer, parameter   :: maxlag=20
      integer              :: npar
      real(kind(1e0))      :: aic, avar
      real(kind(1e0))      :: par(maxlag)
      real(kind(1e0))      :: x(176,2)
      integer              :: ncol, nrow
!                                  SPECIFICATIONS FOR LOCAL VARIABLES
      integer              :: nout
!
      call umach (2, nout)
      write(nout,*) 'AIC Automatic Order selection '
      write(nout,*) 'AR coefficients estimated using Maximum Likelihood'
      write(nout,*)
!                                  Get Wolfer Sunspot Data
      call gdata(2,x,nrow,ncol)
!                                  Example #1
      call auto_uni_ar(x(22:,2), maxlag, npar, par, aic=aic, imeth=1,&
                       avar=avar)
      write(nout,*) 'Order Selected: ', npar
      write(nout,*) 'AIC = ', aic,'      Variance = ', avar
      call wrrrn('Final AR Coefficients estimated by MAXIMUM LIKELIHOOD', &
                  par, nra=npar, nca=1, lda=npar)
      end
```

Output

```
     AIC Automatic Order selection
     AR coefficients estimated using Maximum Likelihood

     Order Selected:  10
     AIC =  1092.0347  Variance =  211.70743

     Final AR Coefficients estimated by MAXIMUM LIKELIHOOD
                           1   1.243
                           2  -0.503
                           3  -0.158
                           4   0.230
                           5  -0.200
                           6   0.114
                           7  -0.079
                           8   0.094
                           9   0.010
                          10   0.096
```

TS_OUTLIER_IDENTIFICATION

more...

Detects and determines outliers and simultaneously estimates the model parameters in a time series whose underlying outlier free series follows a general seasonal or nonseasonal ARMA model.

Required Arguments

MODEL — Array of length 4 containing the order (p, 0, q) x (0, d, 0)$_s$ of the ARIMA model the outlier free series is following. Specifically, `MODEL(1)` = p, `MODEL`(2) = q, `MODEL`(3) = s, `MODEL`(4) = d. (Input)

W — Array of length `NOBS` containing the original time series. (Input)

X — Array of length `NOBS` containing the outlier free series. (Output)

Optional Arguments

NOBS — Number of observations in time series `W`. (Input)
Default: `NOBS` = size (`W`).

DELTA — The dynamic dampening effect parameter used in the detection of a Temporary Change Outlier (TC), 0.0 < `DELTA` < 1.0. (Input)
Default: `DELTA` = 0.7 .

CRITICAL — Critical value used as a threshold for outlier detection, `CRITICAL` > 0. (Input)
Default: `CRITICAL` = 3.0.

EPSILON — Positive tolerance value controlling the accuracy of parameter estimates during outlier detection. (Input)
Default: `EPSILON` = 0.001.

RELERR — Stopping criterion for use in the nonlinear equation solver used by `NSPE`, see routine `NSPE` for more details. (Input)
Default: `RELERR` = 1.0e-10.

TOLSS — Tolerance level used to determine convergence of the nonlinear least-squares algorithm used by NSLSE, see routine NSLSE for more details. (Input)
TOLSS must be greater than zero and less than one.
Default: TOLSS = 0.9 × AMACH(4).

RESIDUAL — Array of length NOBS containing the residuals for the outlier free series. (Output)

RESSIGMA — Residual standard error of the outlier free series. (Output)

NOUTLIERS — The number of outliers detected. (Output)

IOUTLIERSTATS — Pointer to an array of size NOUTLIERS by 2 containing outlier statistics. The first column contains the time at which the outlier was observed ($t = t_1, t_1 + 1, t_1 + NOBS$) and the second column contains an identifier indicating the type of outlier observed. Outlier types fall into one of five categories:

IOUTLIERSTATS	Category
0	Innovational Outliers (IO)
1	Additive Outliers (AO)
2	Level Shift Outliers(LS)
3	Temporary Change Outliers(TC)
4	Unable to Identify(UI)

If no outliers are detected, an array of size 0 is returned. (Output)

TAUSTAT — Pointer to an array of length NOUTLIERS containing the *t* value for each detected outlier. (Output)

OMEGA — Pointer to an array of length NOUTLIERS containing the computed ωweights for the detected outliers. (Output)

PARAMS — Array of length 1+*p*+*q* containing the estimated constant, AR and MA parameters, respectively. (Output)

AIC — Akaike's Information Criterion (AIC) for the fitted model. (Output)

AICC— Akaike's Corrected Information Criterion (AICC) for the fitted model. (Output)

BIC — Bayesian Information Criterion (BIC) for the fitted model. (Output)

FORTRAN 90 Interface

Generic: CALL TS_OUTLIER_IDENTIFICATION (MODEL, W, X [, ...])

Specific: The specific interface names are **S_TS_OUTLIER_IDENTIFICATION** and **D_TS_OUTLIER_IDENTIFICATION**.

Description

Consider a univariate time series $\{Y_t\}$ that can be described by the following multiplicative seasonal ARIMA model of order $(p, 0, q) \times (0, d, 0)_s$:

$$Y_t - \mu = \frac{\theta(B)}{\nabla_s^d \phi(B)} a_t,\ t = 1, \ldots, n.$$

Here, $\nabla_s^d = \left(1 - B^s\right)^d$, $\theta\left(B\right) = 1 - \theta_1 B - \ \ldots\ - \theta_q B^q$, $\phi(B) = 1 - \phi_1 B - \ \ldots\ - \phi_p B^p$. B is the lag operator, $B^k Y_t = Y_{t-k}$, $\left\{a_t\right\}$ is a white noise process, and **μ** denotes the mean of the series $\{Y_t\}$.

In general, $\{Y_t\}$ is not directly observable due to the influence of outliers. Chen and Liu (1993) distinguish between four types of outliers: innovational outliers (IO), additive outliers (AO), temporary changes (TC) and level shifts (LS). If an outlier occurs as the last observation of the series, then Chen and Liu's algorithm is unable to determine the outlier's classification. In **TS_OUTLIER_IDENTIFICATION**, such an outlier is called a UI (unable to identify) and is treated as an innovational outlier.

In order to take the effects of multiple outliers occurring at time points $t_1, t_2, \ldots, t_m$ into account, Chen and Liu consider the following model:

$$Y_t^* - \mu = \sum_{j=1}^{m} \omega_j L_j(B) I_t(t_j) + \frac{\theta(B)}{\nabla_s^d \phi(B)} a_t.$$

Here, $\{Y_t^*\}$ is the observed outlier contaminated series, and ω_j and $L_j(B)$ denote the magnitude and dynamic pattern of outlier j, respectively. $I_t(t_j)$ is an indicator function that determines the temporal course of the outlier effect, $I_{tj}(t_j) = 1$, $I_t(t_j) = 0$ otherwise. **Note** that $L_j(B)$ operates on I_t via $B^k I_t = I_{t-k}$, k = 0, 1, ...

The last formula shows that the outlier free series $\{Y_t\}$ can be obtained from the original series $\{Y_t^*\}$ by removing all occurring outlier effects:

$$Y_t = Y_t^* - \sum_{j=1}^{m} \omega_j L_j\left(B\right) I_t\left(t_j\right)$$

The different types of outliers are characterized by different values for $L_j\left(B\right)$:

1. $L_j(B) = \frac{\theta(B)}{\nabla_s^d \phi(B)}$ for an innovational outlier,

2. $L_j(B) = 1$ for an additive outlier,

3. $L_j(B) = (1 - B)^{-1}$ for a level shift outlier *and*

4. $L_j(B) = (1 - \delta B)^{-1},\ 0 < \delta < 1,$ for a temporary change outlier.

TS_OUTLIER_IDENTIFICATION is an implementation of Chen and Liu's algorithm. It determines the coefficients in $\phi(B)$, $\theta(B)$, and the outlier effects in the model for the observed series jointly in three stages. The magnitude of the outlier effects is determined by least squares estimates. Outlier detection itself is realized by examination of the maximum value of the standardized statistics of the outlier effects. For a detailed description, see Chen and Liu's original paper (1993).

Intermediate and final estimates for the coefficients in $\phi(B)$ and $\theta(B)$ are computed by routines NSPE and NSLSE. If the roots of $\phi(B)$ or $\theta(B)$ lie on or within the unit circle, then the algorithm stops with an appropriate error message. In this case, different values for *p* and *q* should be tried.

Examples

Example 1

This example is based on estimates of the Canadian lynx population. TS_OUTLIER_IDENTIFICATION is used to fit an ARIMA(2,2,0) model of the form $(1 - B)^2(1 - \phi_1 B - \phi_2 B^2)Y_t = a_t$, *t* = 1, 2, ..., 144, $\{a_t\}$ Gaussian White noise, to the given series. TS_OUTLIER_IDENTIFICATION computes parameters ϕ_1 = 0.123609 and ϕ_2 = -0.178963, and identifies a LS outlier at time point *t* = 16.

```
          use umach_int
          use ts_outlier_identification_int

          implicit none

          integer :: i, nout
          integer :: noutliers
          integer, dimension(4) :: model
          integer, dimension(:,:), pointer :: outlierstat
          real(kind(1e0)) :: ressigma, aic
          real(kind(1e0)), dimension(3) :: parameters
          real(kind(1e0)), dimension(114) :: w, x

          w = (/ 0.24300E01,0.25060E01,0.27670E01,0.29400E01,0.31690E01,&
                 0.34500E01,0.35940E01,0.37740E01,0.36950E01,0.34110E01,&
                 0.27180E01,0.19910E01,0.22650E01,0.24460E01,0.26120E01,&
                 0.33590E01,0.34290E01,0.35330E01,0.32610E01,0.26120E01,&
                 0.21790E01,0.16530E01,0.18320E01,0.23280E01,0.27370E01,&
                 0.30140E01,0.33280E01,0.34040E01,0.29810E01,0.25570E01,&
                 0.25760E01,0.23520E01,0.25560E01,0.28640E01,0.32140E01,&
                 0.34350E01,0.34580E01,0.33260E01,0.28350E01,0.24760E01,&
                 0.23730E01,0.23890E01,0.27420E01,0.32100E01,0.35200E01,&
```

```
                0.38280E01,0.36280E01,0.28370E01,0.24060E01,0.26750E01,&
                0.25540E01,0.28940E01,0.32020E01,0.32240E01,0.33520E01,&
                0.31540E01,0.28780E01,0.24760E01,0.23030E01,0.23600E01,&
                0.26710E01,0.28670E01,0.33100E01,0.34490E01,0.36460E01,&
                0.34000E01,0.25900E01,0.18630E01,0.15810E01,0.16900E01,&
                0.17710E01,0.22740E01,0.25760E01,0.31110E01,0.36050E01,&
                0.35430E01,0.27690E01,0.20210E01,0.21850E01,0.25880E01,&
                0.28800E01,0.31150E01,0.35400E01,0.38450E01,0.38000E01,&
                0.35790E01,0.32640E01,0.25380E01,0.25820E01,0.29070E01,&
                0.31420E01,0.34330E01,0.35800E01,0.34900E01,0.34750E01,&
                0.35790E01,0.28290E01,0.19090E01,0.19030E01,0.20330E01,&
                0.23600E01,0.26010E01,0.30540E01,0.33860E01,0.35530E01,&
                0.34680E01,0.31870E01,0.27230E01,0.26860E01,0.28210E01,&
                0.30000E01,0.32010E01,0.34240E01,0.35310E01 /)

        model = (/ 2,0,1,2 /)

        call ts_outlier_identification(model, w, x, CRITICAL=3.5, &
                   RESSIGMA=ressigma, NOUTLIERS=noutliers,&
                   IOUTLIERSTATS=outlierstat, PARAMS=parameters,&
                   AIC=aic)

        call umach(2, nout)
        write(nout,FMT="(T2,A,I2)") 'Number of outliers:', noutliers
        write(nout,FMT="(T2,A)") 'Outlier statistics:'
        write(nout,FMT="(T4,A,TR10,A)") 'Time point','Outlier type'
        write(nout,FMT="(I8,TR20,I2)") (outlierstat(i,:),i=1,noutliers)
        write(nout,FMT="(/,T2,A)") 'ARMA parameters:'
        write(nout,FMT="(T3,I2,TR5,f10.6)") (i,parameters(i),i=1,3)
        write(nout,FMT="(/,T2,A,f10.6)") 'RSE: ', ressigma
        write(nout,FMT="(T2,A,f10.6)") 'AIC: ', aic
        write(nout,FMT="(/,T2,A)") 'Extract from the series:'
        write(nout,FMT="(T2,A,TR6,A,TR6,A)") 'time point',&
                              'original series', 'outlier free series'
        do i=1,36
          write(nout, FMT="(T5,I2,T15,F15.6,T35,F19.6)") i, w(i), x(i)
        end do
        end
```

Output

```
 Number of outliers: 1
 Outlier statistics:
   Time point          Outlier type
      16                     2

 ARMA parameters:
   1       0.000000
   2       0.124390
   3      -0.179959

 RSE:   0.319650
 AIC: 282.995575

 Extract from the series:
 time point      original series      outlier free series
     1              2.430000               2.430000
     2              2.506000               2.506000
     3              2.767000               2.767000
```

```
           4           2.940000           2.940000
           5           3.169000           3.169000
           6           3.450000           3.450000
           7           3.594000           3.594000
           8           3.774000           3.774000
           9           3.695000           3.695000
          10           3.411000           3.411000
          11           2.718000           2.718000
          12           1.991000           1.991000
          13           2.265000           2.265000
          14           2.446000           2.446000
          15           2.612000           2.612000
          16           3.359000           2.701984
          17           3.429000           2.771984
          18           3.533000           2.875984
          19           3.261000           2.603984
          20           2.612000           1.954984
          21           2.179000           1.521984
          22           1.653000           0.995984
          23           1.832000           1.174984
          24           2.328000           1.670984
          25           2.737000           2.079984
          26           3.014000           2.356984
          27           3.328000           2.670984
          28           3.404000           2.746984
          29           2.981000           2.323984
          30           2.557000           1.899984
          31           2.576000           1.918984
          32           2.352000           1.694984
          33           2.556000           1.898984
          34           2.864000           2.206984
          35           3.214000           2.556984
          36           3.435000           2.777984
```

Example 2

This example is an artificial realization of an ARMA(1,1) process via formula

$Y_t - 0.8Y_{t-1} = 10.0 + a_t + 0.5a_{t-1},\ t = 1, \ldots, 300,\ \{a_t\}$ Gaussian white noise, $E\left[Y_t \right] = 50.0$.

An additive outlier with $\omega_1 = 4.5$ was added at time point $t = 150$, a temporary change outlier with $\omega_2 = 3.0$ was added at time point $t = 200$.

```
        use umach_int
        use ts_outlier_identification_int

        implicit none

        integer :: i, nout
        integer :: noutliers
        integer, dimension(4) :: model
        integer, dimension(:,:), pointer :: outlierstat
        real(kind(1e0)) :: ressigma, aic
        real(kind(1e0)), dimension(3) :: parameters
        real(kind(1e0)), dimension(300) :: w, x
        real(kind(1e0)), dimension(:), pointer :: omega
```

```
w = (/ 50.0000000,50.2728081,50.6242599,51.0373917,51.9317627,&
       50.3494759,51.6597252,52.7004929,53.5499802,53.1673279,&
       50.2373505,49.3373871,49.5516472,48.6692696,47.6606636,&
       46.8774185,45.7315445,45.6469727,45.9882355,45.5216560,&
       46.0479660,48.1958656,48.6387749,49.9055367,49.8077278,&
       47.7858467,47.9386749,49.7691956,48.5425873,49.1239853,&
       49.8518791,50.3320694,50.9146347,51.8772049,51.8745689,&
       52.3394470,52.7273712,51.4310036,50.6727448,50.8370399,&
       51.2843437,51.8162918,51.6933670,49.7038231,49.0189247,&
       49.455703,50.2718010,49.9605980,51.3775749,50.2285385,&
       48.2692299,47.6495590,49.2938499,49.1924858,49.6449242,&
       50.0446815,51.9972496,54.2576981,52.9835434,50.4193535,&
       50.3617897,51.8276901,53.1239929,54.0682144,54.9238319,&
       55.6877632,54.8896332,54.0701065,52.2754097,52.2522354,&
       53.1248703,51.1287193,50.5003815,49.6504173,47.2453079,&
       45.4555626,45.8449707,45.9765129,45.7682228,45.2343674,&
       46.6496811,47.0894432,49.3368340,50.8058052,49.9132500,&
       49.5893288,48.2470627,46.9779968,45.6760864,45.7070389,&
       46.6158409,47.5303612,47.5630417,47.0389214,46.0352287,&
       45.8161545,45.7974396,46.0015373,45.3796463,45.3461685,&
       47.6444016,49.3327446,49.3810692,50.2027817,51.4567032,&
       52.3986320,52.5819206,52.7721825,52.6919098,53.3274345,&
       55.1345940,56.8962631,55.7791634,55.0616989,52.3551178,&
       51.3264084,51.0968323,51.1980476,52.8001442,52.0545082,&
       50.8742943,51.5150337,51.2242050,50.5033989,48.7760124,&
       47.4179192,49.7319527,51.3320541,52.3918304,52.4140434,&
       51.0845947,49.6485748,50.6893463,52.9840813,53.3246994,&
       52.4568024,51.9196091,53.6683121,53.4555359,51.7755814,&
       49.2915611,49.8755112,49.4546776,48.6171913,49.9643021,&
       49.3766441,49.2551308,50.1021881,51.0769119,55.8328133,&
       52.0212708,53.4930801,53.2147255,52.2356453,51.9648819,&
       52.1816330,51.9898071,52.5623627,51.0717278,52.2431946,&
       53.6943054,54.3752098,54.1492615,53.8523254,52.1093712,&
       52.3982697,51.2405128,50.3018112,51.3819618,49.5479546,&
       47.5024452,47.4447708,47.8939056,48.4070015,48.2440681,&
       48.7389755,49.7309227,49.1998024,49.5798340,51.1196213,&
       50.6288414,50.3971405,51.6084099,52.4564743,51.6443901,&
       52.4080658,52.4643364,52.6257210,53.1604691,51.9309731,&
       51.4137230,52.1233368,52.9867249,53.3180733,51.9647636,&
       50.7947655,52.3815842,50.8353729,49.4136009,52.8355217,&
       52.2234840,51.1392517,48.5245132,46.8700218,46.1607285,&
       45.2324257,47.4157829,48.9989090,49.6230736,50.4352913,&
       51.1652985,50.2588654,50.7820129,51.0448799,51.2880516,&
       49.6898804,49.0288200,49.9338837,48.2214432,46.2103348,&
       46.9550171,47.5595894,47.7176018,48.4502945,50.9816895,&
       51.6950073,51.6973495,52.1941261,51.8988075,52.5617599,&
       52.0218391,49.5236053,47.9684906,48.2445183,48.8275146,&
       49.7176971,51.5649338,52.5627213,52.0182419,50.9688835,&
       51.5846901,50.9486771,48.8685837,48.5600624,48.4760094,&
       48.5348396,50.4187813,51.2542381,50.1872864,50.4407692,&
       50.6222687,50.4972000,51.0036087,51.3367500,51.7368202,&
       53.0463791,53.6261253,52.0728683,48.9740753,49.3280830,&
       49.2733917,49.8519020,50.8562126,49.5594254,49.6109200,&
       48.3785629,48.0026474,49.4874268,50.1596375,51.8059540,&
       53.0288620,51.3321075,49.3114815,48.7999306,47.7201881,&
       46.3433914,46.5303612,47.6294632,48.6012459,47.8567657,&
       48.0604057,47.1352806,49.5724792,50.5566483,49.4182968,&
       50.5578079,50.6883736,50.6333389,51.9766159,51.0595245,&
       49.3751640,46.9667702,47.1658173,47.4411278,47.5360374,&
```

```
                48.9914742,50.4747620,50.2728043,51.9117165,53.7627792 /)


        model = (/ 1,1,1,0 /)

        call ts_outlier_identification(model, w, x, CRITICAL=3.5,&
                         RESSIGMA=ressigma, NOUTLIERS=noutliers,&
                         IOUTLIERSTATS=outlierstat,OMEGA=omega,&
                         PARAMS=parameters, AIC=aic, RELERR=1.0e-05)

        call umach(2, nout)
        write(nout,FMT="(/,T2,A)") 'ARMA parameters:'
        write(nout,FMT="(T3,I2,TR5,f10.6)") (i,parameters(i),i=1,3)
        write(nout,FMT="(/,T2,A,I2)") 'Number of outliers:', noutliers
        write(nout,FMT="(/,T2,A)") 'Outlier statistics:'
        write(nout,FMT="(T4,A,TR10,A)") 'Time point','Outlier type'
        write(nout,FMT="(I8,TR20,I2)") (outlierstat(i,:),i=1,noutliers)
        write(nout,FMT="(/,T2,A)") 'Omega statistics:'
        write(nout,FMT="(T4,A,TR10,A)") 'Time point','Omega'
        write(nout,FMT="(I9,TR10,f10.6)") (outlierstat(i,1),omega(i),&
                                           i=1,noutliers)
        write(nout,FMT="(/,T2,A,f10.6)") 'RSE: ', ressigma
        write(nout,FMT="(T2,A,f12.6)") 'AIC: ', aic
        end
```

Output

```
 ARMA parameters:
    1      10.700689
    2       0.787765
    3      -0.498039

  Number of outliers: 2

  Outlier statistics:
    Time point          Outlier type
      150                      1
      200                      3

  Omega statistics:
    Time point          Omega
       150            4.478198
       200            3.381984

  RSE:   1.007209
  AIC:  1417.036377
```

TS_OUTLIER_FORECAST

Computes forecasts, associated probability limits and Ψ weights for an outlier contaminated time series.

Required Arguments

W — Array of length NOBS containing the outlier free time series. (Input)

RESIDUAL — Array of length NOBS containing the residuals of the outlier free time series determined from routine TS_OUTLIER_IDENTIFICATION. (Input)

NOUTLIERS — The number of outliers in W, determined from routine TS_OUTLIER_IDENTIFICATION. (Input)

IOUTLIERSTATS — Array of size NOUTLIERS by 2 containing outlier statistics from routine TS_OUTLIER_IDENTIFICATION. (Input).
The first column contains the time at which the outlier was observed ($t = t_1, t_1 + 1, \ldots, t_1 + NOBS$) and the second column contains an identifier indicating the type of outlier observed. Outlier types fall into one of five categories:

IOUTLIERSTATS	Category
0	Innovational Outliers (IO)
1	Additive Outliers (AO)
2	Level Shift Outliers (LS)
3	Temporary Change Outliers (TC)
4	Unable to Identify (UI)

If NOUTLIERS = 0 this array is ignored.

OMEGA — Array of length NOUTLIERS containing the omega weights for the outliers determined through routine TS_OUTLIER_IDENTIFICATION. (Input)

DELTA — The dynamic dampening effect parameter used in the outlier detection, 0.0 < DELTA < 1.0. (Input)

MODEL — Array of length four containing estimates for *p*, *q*, *s* and *d* in MODEL(1), MODEL(2), MODEL(3) and MODEL(4), respectively. (Input)

PARAMS — Array of length 1+*p*+*q* containing the estimated constant, AR and MA parameters as output from routine TS_OUTLIER_IDENTIFICATION. (Input)

MXLEAD — Maximum lead time for forecasts. (Input)
The forecasts are taken at origin $t = t_{NOBS}$, the time point of the last observed value in the series, for lead times 1, 2,..., MXLEAD.
MXLEAD must be greater than zero.

FCST — An array of size MXLEAD by 3 containing the forecasted values for the outlier contaminated series in the first column. The second column contains the deviations from each forecast that give the 100(1 - ALPHA)% probability limits, and the third column contains the ψ weights of the infinite order moving average form of the model. (Output)

Optional Arguments

NOBS — Number of observations in the time series W. (Input)
Default: NOBS = size (W).

ALPHA — Value in the exclusive interval (0,1) used to specify the 100(1-ALPHA)% probability limits of the forecast. (Input)
Default: ALPHA = 0.05

OUTFREEFCST — An MXLEAD by 3 array containing the forecasted values for the original outlier free series in the first column. The second column contains standard errors for these forecasts, and the third column contains the ψ weights of the infinite order moving average form of the model. (Output)

FORTRAN 90 Interface

Generic: CALL TS_OUTLIER_FORECAST (W, RESIDUAL, NOUTLIERS, IOUTLIERSTATS, OMEGA, DELTA, MODEL, PARAMS, MXLEAD, FCST [, ...])

Specific: The specific interface names are S_TS_OUTLIER_FORECAST and D_TS_OUTLIER_FORECAST.

Description

Consider the following model for a given outlier contaminated univariate time series $\left\{ Y_t^* \right\}_{t=1,\ldots,n}$:

$$Y_t^* = Y_t + \sum_{j=1}^{m} \omega_j L_j(B) I_t(t_j).$$

For an explanation of the notation, see the "Description" section for TS_OUTLIER_IDENTIFICATION. It follows from the formula above that the Box-Jenkins forecast at origin *t* for lead time *l*, $\hat{Y}_t^*(l)$, can be computed as:

$$\hat{Y}_t^*(l) = \hat{Y}_t(l) + \sum_{j=1}^{m} \omega_j L_j(B) I_{t+l}(t_j), \quad l = 1, \ldots, \texttt{MXLEAD}$$

Therefore, computation of the forecasts for $\left\{ Y_t^* \right\}$ is done in two steps:

1. Computation of the forecasts for the outlier free series $\left\{ Y_t \right\}$.
2. Computation of the forecasts for the original series $\left\{ Y_t^* \right\}$ by adding the multiple outlier effects to the forecasts for $\left\{ Y_t \right\}$.

Step 1 above:

Since

$$\varphi(B)(Y_t - \mu) = \theta(B)\, a_t,$$

where

$$\varphi(B) := \nabla_s^d \phi(B) = 1 - \varphi_1 B - \ \ldots \ - \varphi_{p+sd} B^{p+sd},$$

the Box-Jenkins forecast at origin t for lead time l, $\hat{Y}_t(l)$, can be computed recursively as:

$$\hat{Y}_t(l) = (1 - \sum_{j=1}^{p+sd} \varphi_j)\mu + \sum_{j=1}^{p+sd} \varphi_j \hat{Y}_t(l-j) - \sum_{j=l}^{q} \theta_j a_{t+l-j}.$$

Here,

$$\hat{Y}_t(l-j) = \begin{cases} Y_{t+l-j} & \text{for} \quad l - j \le 0 \\ \hat{Y}_t(l-j) & \text{for} \quad l - j > 0 \end{cases},$$

and

$$a_k = \begin{cases} 0 & \text{for } k \le \max\{1,\ p + sd\} \\ Y_k - \hat{Y}_{k-1}(1) & \text{for } k = \max\{1, p + sd\} + 1, \ \ldots, n \end{cases}.$$

Step 2 above:

The formulas for $L_j(B)$ for the different types of outliers are as follows:

Innovational outliers (IO)	$L_j(B) = \frac{\theta(B)}{\nabla_s^d \phi(B)} := \psi(B) = \sum_{k=0}^{\infty} \psi_k B^k, \ \psi_0 = 1$
Additive outliers (AO)	$L_j(B) = 1$

Level shifts (LS) $L_j(B) = \frac{1}{1-B} = \sum_{k=0}^{\infty} B^k$

Temporary changes (TC) $L_j(B) = \frac{1}{1-\delta B} = \sum_{k=0}^{\infty} \delta^k B^k$

Assuming the outlier occurs at time point t_j, the outlier impact is therefore:

Innovational outliers (IO) $$\omega_j L_j(B) I_t(t_j) = \begin{cases} 0 & \text{for} \quad t < t_j, \\ \omega_j \psi_k & \text{for} \quad t = t_j + k,\ k \geq 0 \end{cases}$$

Additive outliers (AO) $$\omega_j L_j(B) I_t(t_j) = \begin{cases} 0 & \text{for} \quad t \neq t_j \\ \omega_j & \text{for} \quad t = t_j \end{cases}$$

Level shifts (LS) $$\omega_j L_j(B) I_t(t_j) = \begin{cases} 0 & \text{for} \quad t < t_j \\ \omega_j & \text{for} \quad t = t_j + k,\ k \geq 0 \end{cases}$$

Temporary changes (TC) $$\omega_j L_j(B) I_t(t_j) = \begin{cases} 0 & \text{for} \quad t < t_j, \\ \omega_j \delta^k & \text{for} \quad t = t_j + k,\ k \geq 0 \end{cases}$$

From these formulas, the forecasts $\hat{Y}_t^*(l)$ can be computed easily.

The $100(1-\alpha)$ percent probability limits for Y_{t+l}^* and Y_{t+l} are given by

$$\hat{Y}_t^*(l) \text{ (or } \hat{Y}_t(l), \textit{resp.}) \pm u_{\alpha/2}(1 + \sum_{j=1}^{l-1} \psi_j^2)^{1/2} s_a,$$

where $u_{\alpha/2}$ is the $100(1-\alpha/2)$ percentile of the standard normal distribution, s_a^2 is an estimate of the variance σ_a^2 of the random shocks (returned from routine `TS_OUTLIER_IDENTIFICATION`), and the ψ weights $\{\psi_j\}$ are the coefficients in

$$\psi(B) := \sum_{k=0}^{\infty} \psi_k B^k := \frac{\theta(B)}{\nabla_s^d \phi(B)},\ \psi_0 = 1\,.$$

For a detailed explanation of these concepts, see Chapter 5: "Forecasting", Box, Jenkins and Reinsel (1994).

Example

This example is a realization of an ARMA(2,1) process described by the model $Y_t - Y_{t-1} + 0.24Y_{t-2} = 10.0 + a_t + 0.5a_{t-1},\ \{a_t\}$, a Gaussian white noise process.

Outliers were artificially added to the outlier free series $\{Y_t\}_{t=1,\ldots,280}$ at time points $t = 150$ (level shift, $\omega_1 = +2.5$) and $t = 200$ (additive outlier, $\omega_2 = +3.2$), resulting in the outlier contaminated series $\{Z_t\}_{t=1,\ldots,280}$. For both series, forecasts were determined for time points $t = 281, \ldots, 290$ and compared with the actual values of the series.

```
    USE TS_OUTLIER_IDENTIFICATION_INT
    USE TS_OUTLIER_FORECAST_INT
    USE WRRRL_INT
    USE UMACH_INT

    IMPLICIT NONE

    real(kind(1e0)), dimension(290) :: w = (/ &
      41.6699982,41.6699982,42.0752144,42.6123962,43.6161919,42.1932831, &
      43.1055450,44.3518715,45.3961258,45.0790215,41.8874397,40.2159805, &
      40.2447319,39.6208458,38.6873589,37.9272423,36.8718872,36.8310852, &
      37.4524879,37.3440933,37.9861374,40.3810501,41.3464622,42.6495285, &
      42.6096764,40.3134537,39.7971268,41.5401535,40.7160759,41.0363541, &
      41.8171883,42.4190292,43.0318832,43.9968109,44.0419617,44.3225212, &
      44.6082611,43.2199631,42.0419197,41.9679718,42.4926224,43.2091255, &
      43.2512283,41.2301674,40.1057358,40.4510574,41.5329170,41.5678177, &
      43.0090141,42.1592140,39.9234505,38.8394127,40.4319878,40.8679352, &
      41.4551926,41.9756317,43.9878922,46.5736389,45.5939293,42.4487762, &
      41.5325394,42.8830910,44.5771217,45.8541985,46.8249474,47.5686378, &
      46.6700745,45.4120026,43.2305107,42.7635345,43.7112923,42.0768661, &
      41.1835632,40.3352280,37.9761467,35.9550056,36.3212509,36.9925880, &
      37.2625008,37.0040665,38.5232544,39.4119797,41.8316803,43.7091446, &
      42.9381447,42.1066780,40.3771248,38.6518707,37.0550499,36.9447708, &
      38.1017685,39.4727097,39.8670387,39.3820763,38.2180786,37.7543488, &
      37.7265244,38.0290642,37.5531158,37.4685936,39.8233147,42.0480766, &
      42.4053535,43.0117416,44.1289330,45.0393829,45.1114540,45.0086479, &
      44.6560631,45.0278931,46.7830849,48.7649765,47.7991905,46.5339661, &
      43.3679199,41.6420822,41.2694893,41.5959740,43.5330009,43.3643608, &
      42.1471291,42.5552788,42.4521446,41.7629128,39.9476891,38.3217010, &
      40.5318718,42.8811569,44.4796944,44.6887932,43.1670265,41.2226143, &
      41.8330154,44.3721924,45.2697029,44.4174194,43.5068550,44.9793015, &
      45.0585403,43.2746620,40.3317070,40.3880501,40.2627106,39.6230278, &
      41.0305252,40.9262009,40.8326912,41.7084885,42.9038048,45.8650513, &
      46.5231590,47.9916115,47.8463135,46.5921936,45.8854408,45.9130440, &
      45.7450371,46.2964249,44.9394569,45.8141251,47.5284042,48.5527802, &
      48.3950577,47.8753052,45.8880005,45.7086983,44.6174774,43.5567932, &
      44.5891113,43.1778679,40.9405632,40.6206894,41.3330421,42.2759552, &
      42.4744949,43.0719833,44.2178459,43.8956337,44.1033440,45.6241455, &
      45.3724861,44.9167595,45.9180603,46.9077835,46.1666603,46.6013489, &
      46.6592331,46.7291603,47.1908340,45.9784355,45.1215782,45.6791115, &
      46.7379875,47.3036957,45.9968834,44.4669495,45.7734680,44.6315041, &
      42.9911766,46.3842583,43.7214432,43.5276833,41.3946495,39.7013168, &
      39.1033401,38.5292892,41.0096245,43.4535828,44.6525154,45.5725899, &
      46.2815285,45.2766647,45.3481712,45.5039482,45.6745682,44.0144806, &
      42.9305000,43.6785469,42.2500534,40.0007210,40.4477005,41.4432716, &
      42.0058670,42.9357758,45.6758842,46.8809929,46.8601494,47.0449791, &
      46.5420647,46.8939934,46.2963371,43.5479164,41.3864059,41.4046364, &
      42.3037987,43.6223717,45.8602371,47.3016396,46.8632469,45.4651413, &
      45.6275482,44.9968376,42.7558670,42.0218239,41.9883728,42.2571678, &
      44.3708687,45.7483635,44.8832512,44.7945862,44.8922577,44.7409401, &
      45.1726494,45.5686874,45.9946709,47.3151054,48.0654068,46.4817467, &
```

```
      42.8618279,42.4550323,42.5791168,43.4230957,44.7787971,43.8317108, &
      43.6481781,42.4183960,41.8426285,43.3475227,44.4749908,46.3498306, &
      47.8599319,46.2449913,43.6044006,42.4563484,41.2715340,39.8492508, &
      39.9997292,41.4410820,42.9388237,42.5687332,42.6384087,41.7088661, &
      43.9399033,45.4284401,44.4558411,45.1761856,45.3489113,45.1892662, &
      46.3754730,45.6082802 /)

    integer :: i, nout
    integer, dimension(4) :: model
    integer :: noutliers, nobs=280, mxlead=10
    integer, dimension(:,:), pointer :: outlierstat
    real(kind(1e0)), dimension(:), pointer :: omega
    real(kind(1e0)) :: delta = 0.7, res_sigma, aic
    real(kind(1e0)), dimension(280) :: x, residual
    real(kind(1e0)), dimension(10,4) :: forecast_table
    real(kind(1e0)), dimension(10,3) :: fcst, outfreefcst
    real(kind(1e0)), dimension(4) :: params
    character (len = 10) :: fmt
    character (len = 20), dimension(5) :: clabel
    character (len = 4), dimension(10) :: rlabel

    fmt = '(F11.4)'
    clabel(1) = '   '
    clabel(2) = 'Orig. series'
    clabel(3) = 'forecast'
    clabel(4) = 'prob. limits'
    clabel(5) = 'psi weights'

    rlabel = (/ ' 1',' 2',' 3',' 4',' 5',' 6',' 7',' 8',' 9','10' /)

    model = (/ 2,1,1,0 /)

    CALL TS_OUTLIER_IDENTIFICATION (model, w, x, NOBS=nobs, DELTA=delta,  &
               RELERR=1.0e-5, RESIDUAL=residual,  &
               RESSIGMA=res_sigma, NOUTLIERS=noutliers, &
               IOUTLIERSTATS=outlierstat, OMEGA=omega, &
               PARAMS=params, AIC=aic)

    CALL UMACH(2, nout)
    WRITE (nout,*) 'ARMA parameters:'
    DO i=1, 1+model(1)+model(2)
      WRITE (nout,*) i,'    ', params(i)
    END DO

    WRITE (nout,*)
    WRITE (nout,*) 'Number of outliers: ', noutliers
    WRITE (nout,*)
    WRITE (nout,*) 'Outlier statistics: '
    WRITE (nout,FMT="(T2, A, T22, A)") 'Time point','Outlier type'
    DO i=1,noutliers
      WRITE (nout,FMT="(T2, I10, T22, I12)") outlierstat(i,1), outlierstat(i,2)
    END DO

    WRITE (nout,*)
    WRITE (nout,*) 'RSE: ', res_sigma
    WRITE (nout,*) 'AIC: ', aic
    WRITE (nout,*)

    CALL TS_OUTLIER_FORECAST (x, residual, noutliers, outlierstat, &
                  omega, delta, model, params, mxlead, fcst,  &
```

```
                NOBS=nobs, OUTFREEFCST=outfreefcst)

forecast_table(1:mxlead,1) = w(281:290)
forecast_table(1:mxlead,2:4) = fcst(1:mxlead,1:3)

CALL WRRRL ('* * * Forecast Table for outlier contaminated series * * *',&
            forecast_table, rlabel, clabel, FMT=fmt)

forecast_table(1:mxlead,1) = w(281:290) - 2.5
forecast_table(1:mxlead,2:4) = outfreefcst(1:mxlead,1:3)

WRITE (nout,*)

clabel(2) = 'Outlier free series'
CALL WRRRL ('* * * Forecast Table for outlier free series * * *', &
            forecast_table, RLABEL, CLABEL, FMT=fmt)
END
```

Output

```
ARMA parameters:
  1      8.837544
  2      0.9461826
  3      -0.1512835
  4      -0.5606939

 Number of outliers:  2

 Outlier statistics:
 Time point          Outlier type
        150                     2
        200                     1

 RSE:  1.0042976
 AIC:  1323.6127

             * * * Forecast Table for outlier contaminated series * * *
                  Orig. series     forecast  prob. limits  psi weights
               1       42.6384      42.3113        1.9684       1.5069
               2       41.7089      42.7868        3.5598       1.2745
               3       43.9399      43.2756        4.3550       0.9779
               4       45.4284      43.6662        4.7615       0.7325
               5       44.4558      43.9618        4.9750       0.5451
               6       45.1762      44.1825        5.0894       0.4050
               7       45.3489      44.3465        5.1514       0.3007
               8       45.1893      44.4683        5.1853       0.2233
               9       46.3755      44.5588        5.2039       0.1658
              10       45.6083      44.6259        5.2141       0.1231

             * * * Forecast Table for outlier free series * * *
                 Outlier free     forecast  prob. limits  psi weights
                       series
               1       40.1384      40.5805        1.9684       1.5069
               2       39.2089      41.0560        3.5598       1.2745
               3       41.4399      41.5449        4.3550       0.9779
               4       42.9284      41.9355        4.7615       0.7325
               5       41.9558      42.2311        4.9750       0.5451
               6       42.6762      42.4517        5.0894       0.4050
```

7	42.8489	42.6158	5.1514	0.3007
8	42.6893	42.7376	5.1853	0.2233
9	43.8755	42.8281	5.2039	0.1658
10	43.1083	42.8952	5.2141	0.1231

AUTO_ARIMA

more...

Automatically identifies time series outliers, determines parameters of a multiplicative seasonal ARIMA $(p,0,q) \times (0,d,0)_S$ model and produces forecasts that incorporate the effects of outliers whose effects persist beyond the end of the series.

Required Arguments

ITIME_POINTS — Array of length NOBS containing the time points $t_1, t_2, \ldots, t_{\text{NOBS}}$ at which the time series was observed. Time points must be integer and in strictly ascending order. It is assumed that the time points of the time series after estimation of the missing values are equidistant with distance 1 between two consecutive time points. (Input)

W — Array of length NOBS containing the observed time series values $Y_1^*, Y_2^*, \ldots, Y_{\text{NOBS}}^*$. This series can contain outliers and missing observations. Outliers are identified by this routine and missing values are identified by the time values in array ITIME_POINTS. If the time interval between two consecutive time points is greater than one, i.e. $t_{i+1} - t_i = m > 1$, then $m - 1$ missing values are assumed to exist between t_i and t_{i+1} at times $t_i + 1, t_i + 2, \ldots, t_{i+1} - 1$. Therefore, the gap free series is assumed to be defined for equidistant time points . Missing values are automatically estimated prior to identifying outliers and producing forecasts. Forecasts are generated for both missing and observed values. (Input)

PARAMS — Allocatable array of length 1+*p*+*q* containing the estimated constant, AR and MA parameters of the adjusted optimum seasonal ARIMA $(p,0,q) \times (0,d,0)_S$ model. If *d* = 0, then an ARMA(*p, q*) model is fitted to the outlier-free version of the observed series Y_t^*. If *d* > 0, these parameters are computed for an ARMA(***p***,***q***) representation of the seasonally adjusted series $Z_t^* = \nabla_s^d \cdot Y_t^* = (1 - B_s)^d \cdot Y_t^*$, where $B_s Y_t^* = Y_{t-s}^*$ and s ≥ 1…. (Output)

Optional Arguments

NOBS — Number of observations in the time series. Assuming that the series is defined at time points $t_1, \ldots, t_{NOBS}$, the actual length of the series, including missing values, is $N = t_{NOBS} - t_1 + 1$. (Input)
Default: NOBS = size (W).

IMETH — Method to be used in model selection. (Input)
1 –Automatic $\text{ARIMA}(p,0,0) \times (0,d,0)_s$ selection

2 – Grid search (requires arguments IAR and IMA)

3 – Specified $\text{ARIMA}(p,0,q) \times (0,d,0)_s$ model (Requires argument MODEL)

Default: IMETH = 1.

MAXLAG — Maximum number of autoregressive parameters allowed in fitting an AR model to the series. (Input)
Default: MAXLAG = 10.

INFOCRIT — The information criterion used for optimum model selection. (Input)

INFOCRIT	selected information criterion
0	Akaike's Information Criterion (AIC)
1	Akaike's Corrected Information Criterion (AICC)
2	Bayesian Information Criterion (BIC)

Default: INFOCRIT = 0.

DELTA — Dynamic dampening effect parameter used in the detection of a Temporary Change Outlier (TC). (Input)
It is required that 0.0 < DELTA < 1.0.
Default: DELTA = 0.7.

CRITICAL — Critical value used as a threshold for outlier detection. (Input)
CRITICAL must be greater than zero.
Default: CRITICAL = 3.0.

EPSILON — Positive tolerance value controlling the accuracy of parameter estimates during outlier detection. (Input)
Default: EPSILON = 0.001.

TOLSS — Tolerance level used to determine convergence of the nonlinear least-squares algorithm used by NSLSE, see routine NSLSE for more details. (Input)
TOLSS must be greater than zero and less than one.
Default: TOLSS = 0.9 × AMACH(4).

IAR — Array containing the candidate values for the AR order *p* from which the optimum is being selected. All candidate values in IAR must be non-negative and IAR must contain at least one value. If IMETH = 2 then IAR must be defined. Otherwise, IAR is ignored. (Input)

IMA — Array containing the candidate values for the MA order *q* from which the optimum is being selected. All candidate values in IMA must be non-negative and IMA must contain at least one value. If IMETH = 2 then IMA must be defined. Otherwise, IMA is ignored. (Input)

IPER — Array containing the candidate values for *s* from which the optimum is being selected. All candidate values in IPER must be positive and IPER must contain at least one value. (Input)
Default: IPER(:) = 1

IORD — Array containing the candidate values for *d* from which the optimum is being selected. All candidate values in IORD must be non-negative and IORD must contain at least one value. (Input)
Default: IORD (:) = 0

ALPHA — Value in the exclusive interval (0,1) used to specify the 100(1-ALPHA)% probability limits of the forecast. (Input)
Default: ALPHA = 0.05.

MXLEAD — Maximum lead time for forecasts. (Input)
Default: MXLEAD = 0.

MODEL — Array of length 4 containing values for *p, q, s*, and *d*. (Input/Output)
For IMETH = 1 or IMETH = 2 , MODEL is ignored on input. If IMETH = 3 then *p* and *q* must be defined on input. If IPER and IORD are not defined then *s* and *d* must also be defined on input. On output, MODEL contains optimum values for *p, q, s*, and *d* in MODEL(1), MODEL(2), MODEL(3) and MODEL(4), respectively.

RESIDUAL — Array of length $N = t_{NOBS} - t_1 + 1$ containing estimates for the white noise in the gap-free and outlier-free original series. (Output)

RSE — Residual standard error (RSE) of the outlier-free and gap-free original series. (Output)

NOUTLIERS — Number of outliers detected. (Output)

IOUTLIERSTATS — Pointer to an array of size NOUTLIERS by 2 containing the outlier statistics. The first column contains the time point at which the outlier was observed (time points ranging from t_1 to t_{NOBS}) and the second column contains an identifier indicating the type of outlier observed. Outlier types fall into one of five categories:

IOUTLIERSTATS	Category
0	Innovational Outliers (IO)
1	Additive Outliers (AO)
2	Level Shift Outliers (LS)
3	Temporary Change Outliers (TC)
4	Unable to Identify (UI)

If no outliers are detected, then an array of size 0 is returned. (Output)

AIC — Akaike's Information Criterion (AIC) for the fitted optimum model. Uses an approximation of the maximum log-likelihood based on an estimate of the innovation variance of the series. (Output)

AICC — Akaike's Corrected Information Criterion (AICC) for the fitted optimum model. Uses an approximation of the maximum log-likelihood based on an estimate of the innovation variance of the series. (Output)

BIC — Bayesian Information Criterion (BIC) for the fitted optimum model. Uses an approximation of the maximum log-likelihood based on an estimate of the innovation variance of the series. (Output)

OUTFREESERIES — Array of size N by 2 containing the adjusted time series. The first column contains the NOBS observations from the original series, $\{Y_t^*\}$, plus estimated values for any time gaps. The second column contains the same values as the first column adjusted by removing any outlier effects. In effect, the second column contains estimates of the underlying outlier-free series, $\{Y_t\}$. If no outliers are detected then both columns will contain identical values. (Output)

OUTFREEFCST — Array of size MXLEAD by 3 containing the forecasted values for the original outlier and gap free series at origin t_{NOBS} for lead times $1, \ldots,$MXLEAD in the first column. The second column contains standard errors for these forecasts, and the third column contains the Ψ weights of the infinite order moving average form of the model. (Output)

OUTLIERFCST — Array of size MXLEAD by 3 containing the forecasted values for the original and gap free series for $t = t_{\text{NOBS}} + 1, \ldots, t_{\text{NOBS}} +$ MXLEAD in the first column. The second column contains standard errors for these forecasts, and the third column contains the Ψ weights of the infinite order moving average form of the model. (Output)

FORTRAN 90 Interface

Generic: CALL AUTO_ARIMA (ITIME_POINTS, W, PARAMS [, ...])

Specific: The specific interface names are S_AUTO_ARIMA and D_AUTO_ARIMA.

Description

Routine **AUTO_ARIMA** determines the parameters of a multiplicative seasonal ARIMA $(p,0,q) \times (0,d,0)_S$ model, and then uses the fitted model to identify outliers and prepare forecasts. The order of this model can be specified or automatically determined.

The ARIMA $(p,0,q) \times (0,d,0)_S$ model handled by **AUTO_ARIMA** has the following form:

$$\phi(B)\nabla_s^d(Y_t - \mu) = \theta(B)a_t, \quad t = 1, 2, \ \ldots\ , n,$$

where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ \cdots\ - \phi_p B^p,\ \theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ \cdots\ - \theta_q B^q,\ \nabla_s^d = (1 - B^s)^d$$

and

$$B^k Y_t = Y_{t-k}.$$

It is assumed that all roots of $\phi(B)$ and $\theta(B)$ lie outside the unit circle. Clearly, if $s = 1$ this reduces to the traditional ARIMA(*p, d, q*) model.

Y_t is the unobserved, gap-free and outlier-free time series with mean μ, and white noise a_t. This model is referred to as the underlying, outlier-free model. Routine **AUTO_ARIMA** does not assume that this series is observable. It assumes that the observed values might be contaminated by one or more outliers, whose effects are added to the underlying outlier-free series:

$$Y_t^* = Y_t + outlier\_effect_t.$$

Outlier identification uses the algorithm developed by Chen and Liu (1993). Outliers are classified into 1 of 5 types:

1. innovational
2. additive
3. level shift
4. temporary change *and*
5. unable to identify

Once outliers are identified, **AUTO_ARIMA** estimates Y_t, the outlier-free series representation of the data, by removing the estimated outlier effects.

Using the information about the adjusted ARIMA $(p,0,q) \times (0,d,0)_S$ model and the removed outliers, forecasts are then prepared for the outlier-free series. Outlier effects are added to these forecasts to produce a forecast for the observed series, Y_t^*. If there are no outliers, then the forecasts for the outlier-free series and the observed series will be identical.

Model Selection

Users have an option of either specifying specific values for *p, q*, *s* and *d* or have **AUTO_ARIMA** automatically select best fit values. Model selection can be conducted in one of three methods listed below depending upon the value of optional argument **IMETH**.

Method 1: Automatic ARIMA (p,0,0) x $(0,d,0)_S$ Selection

This method initially searches for the AR(p) representation with minimum AIC for the noisy data, where *p* = 0,..., **MAXLAG**.

If **IORD** is defined then the values in **IPER** and **IORD** are included in the search to find an optimum ARIMA $(p,0,0) \times (0,d,0)_S$ representation of the series. Here, every possible combination of values for *p, s* in **IPER** and *d* in **IORD** is examined. The best found ARIMA $(p,0,0) \times (0,d,0)_S$ representation is then used as input for the outlier detection routine.

The optimum values for *p, q, s* and *d* are returned in **MODEL(1)**, **MODEL(2)**, **MODEL(3)** and **MODEL(4)**, respectively.

Method 2: Grid Search

The second automatic method conducts a grid search for *p* and *q* using all possible combinations of candidate values in **IAR** and **IMA**. Therefore, for this method the definition of **IAR** and **IMA** is required.

If **IORD** is defined, the grid search is extended to include the candidate values for *s* and *d* given in **IPER** and **IORD**, respectively.

If **IORD** is not defined, no seasonal adjustment is attempted, and the grid search is restricted to searching for optimum values of *p* and *q* only.

The optimum values of *p, q, s* and *d* are returned in **MODEL(1)**, **MODEL(2)**, **MODEL(3)** and **MODEL(4)**, respectively.

Method 3: Specified ARIMA $(p,0,q) \times (0,d,0)_s$ Model

In the third method, specific values for p, q, s and d are given. The values for p and q must be defined in `MODEL(1)` and `MODEL(2)`, respectively. If `IPER` and `IORD` are not defined, then values $s > 0$ and $d \geq 0$ must be specified in `MODEL(3)` and `MODEL(4)`. `If IPER` and `IORD` are defined, then a grid search for the optimum values of s and d is conducted using all possible combinations of input values in `IPER` and `IORD`. The optimum values of s and d can be found in `MODEL(3)` and `MODEL(4)`, respectively.

Outliers

The algorithm of Chen and Liu (1993) is used to identify outliers. The number of outliers identified is returned in `NOUTLIERS`. Both the time and classification for these outliers are returned in `IOUTLIERSTATS`. Outliers are classified into one of five categories based upon the standardized statistic for each outlier type. The time at which the outlier occurred is given in the first column of `IOUTLIERSTATS`. The outlier identifier returned in the second column is according to the descriptions in the following table:

Outlier Identifier	Name	General Description
0	(IO) Innovational Outlier	Innovational outliers persist. That is, there is an initial impact at the time the outlier occurs. This effect continues in a lagged fashion with all future observations. The lag coefficients are determined by the coefficient of the underlying ARIMA $(p,0,q) \times (0,d,0)_s$ model.
1	(AO) Additive Outlier	Additive outliers do not persist. As the name implies, an additive outlier affects only the observation at the time the outlier occurs. Hence additive outliers have no effect on future forecasts.
2	(LS) Level Shift	Level shift outliers persist. They have the effect of either raising or lowering the mean of the series starting at the time the outlier occurs. This shift in the mean is abrupt and permanent.
3	(TC) Temporary Change	Temporary change outliers persist and are similar to level shift outliers with one major exception. Like level shift outliers, there is an abrupt change in the mean of the series at the time this outlier occurs. However, unlike level shift outliers, this shift is not permanent. The TC outlier gradually decays, eventually bringing the mean of the series back to its original value. The rate of this decay is modeled using the parameter `DELTA`. The default of `DELTA`= 0.7 is the value recommended for general use by Chen and Liu (1993).
4	(UI) Unable to Identify	If an outlier is identified as the last observation, then the algorithm is unable to determine the outlier's classification. For forecasting, a UI outlier is treated as an IO outlier. That is, its effect is lagged into the forecasts.

Except for additive outliers (AO), the effect of an outlier persists to observations following that outlier. Forecasts produced by **AUTO_ARIMA** take this into account.

Examples

Example 1

This example uses time series LNU03327709 from the US Department of Labor, Bureau of Labor Statistics. It contains the unadjusted special unemployment rate, taken monthly from January 1994 through September 2005. The values 01/2004 – 03/2005 are used by **AUTO_ARIMA** for outlier detection and parameter estimation. In this example , Method 1 without seasonal adjustment is chosen to find an appropriate AR(p) model. A forecast is done for the following six months and compared with the actual values 04/2005 – 09/2005.

```
      use auto_arima_int
      use wrrrl_int
      use umach_int

      implicit none
!                                 Specifications for parameters
      integer :: nobs, mxlead, i, noutliers, nout
      integer, dimension(4) :: model
      integer, dimension(:,:), pointer :: outlierstat
      integer, dimension(141) :: times
      real(kind(1e0)) :: aic, rse
      real(kind(1e0)), dimension(141) :: x
      real(kind(1e0)), dimension(6,3) :: outlierfcst
      real(kind(1e0)), dimension(6,4) :: forecast_table
      real(kind(1e0)), dimension(:), allocatable :: parameters
      character (len=10), dimension(1) :: rlabel
      character (len = 14), dimension(5) :: clabel
      character (len = 10) :: fmt
!                                 Time series data
      x = (/ 12.8,12.2,11.9,10.9,10.6,11.3,11.1,10.4,10.0,9.7,9.7,&
               9.7,11.1,10.5,10.3,9.8,9.8,10.4,10.4,10.0,9.7,9.3,&
                9.6,9.7,10.8,10.7,10.3,9.7,9.5,10.0,10.0,9.3,9.0,&
                   8.8,8.9,9.2,10.4,10.0,9.6,9.0,8.5,9.2,9.0,8.6,&
                     8.3,7.9,8.0,8.2,9.3,8.9,8.9,7.7,7.6,8.4,8.5,&
                     7.8,7.6,7.3,7.2,7.3,8.5,8.2,7.9,7.4,7.1,7.9,&
                     7.7,7.2,7.0,6.7,6.8,6.9,7.8,7.6,7.4,6.6,6.8,&
                     7.2,7.2,7.0,6.6,6.3,6.8,6.7,8.1,7.9,7.6,7.1,&
                   7.2,8.2,8.1,8.1,8.2,8.7,9.0,9.3,10.5,10.1,9.9,&
                   9.4,9.2,9.8,9.9,9.5,9.0,9.0,9.4,9.6,11.0,10.8,&
                10.4,9.8,9.7,10.6,10.5,10.0,9.8,9.5,9.7,9.6,10.9,&
                   10.3,10.4,9.3,9.3,9.8,9.8,9.3,8.9,9.1,9.1,9.1,&
                           10.2,9.9,9.4,8.7,8.6,9.3,9.1,8.8,8.5 /)

      times = (/                      1,2,3,4,5,6,7,8,9,10,11,12,&
                            13,14,15,16,17,18,19,20,21,22,23,24,&
                            25,26,27,28,29,30,31,32,33,34,35,36,&
                            37,38,39,40,41,42,43,44,45,46,47,48,&
                            49,50,51,52,53,54,55,56,57,58,59,60,&
```

```
                                61,62,63,64,65,66,67,68,69,70,71,72,&
                                73,74,75,76,77,78,79,80,81,82,83,84,&
                                85,86,87,88,89,90,91,92,93,94,95,96,&
                           97,98,99,100,101,102,103,104,105,106,107,108,&
                       109,110,111,112,113,114,115,116,117,118,119,120,&
                       121,122,123,124,125,126,127,128,129,130,131,132,&
                                133,134,135,136,137,138,139,140,141 /)

        rlabel(1) = 'NUMBER'
        clabel(1) = '   '
        clabel(2) = 'Series'
        clabel(3) = 'Forecast'
        clabel(4) = 'Prob. Limits'
        clabel(5) = 'Psi Weights'
        mxlead = 6
        nobs = 135

        call auto_arima (times, x, parameters, NOBS=nobs, MAXLAG=5,&
                MODEL=model, AIC=aic, CRITICAL=4.0,&
                NOUTLIERS=noutliers, IOUTLIERSTATS=outlierstat,&
                RSE=rse, MXLEAD=mxlead, OUTLIERFCST=outlierfcst)

        call umach (2,nout)
        write (nout,*) 'Method 1: Automatic ARIMA model selection,'//&
                       ' no differencing'
        write (nout,FMT="(T2,4(A,I2))") 'Model chosen: p =', model(1),&
               ', q =', model(2), ', s =', model(3), ', d =', model(4)
        write (nout,*)
        write (nout,FMT="(T2,A,I2)") 'Number of outliers: ', noutliers
        write (nout,*) 'Outlier statistics:'
        write (nout,*) 'Time point    Outlier type'
        do i=1,noutliers
          write (nout,FMT="(I11,T15,I13)") outlierstat(i,1),&
                 outlierstat(i,2)
        end do
        write (nout,*)
        write (nout,*) 'AIC = ', aic
        write (nout,*) 'RSE = ', rse
        write (nout,*)
        write(nout,FMT="(/,T2,A)") 'ARMA parameters:'
        do i=1, 1+model(1)+model(2)
          write (nout,FMT="(T3,I2,TR5,f10.6)") i, parameters(i)
        end do
        forecast_table(1:mxlead,1) = x(nobs+1:nobs+mxlead)
        forecast_table(1:mxlead, 2:4) = outlierfcst(1:mxlead, 1:3)
        write (nout,*)
        fmt = '(F11.4)'
        call wrrrl('* * * Forecast Table * * *',  forecast_table,&
                 rlabel, clabel, FMT = fmt)
        end
```

Output

```
 Method 1: Automatic ARIMA model selection, no differencing
  Model chosen: p = 5, q = 0, s = 1, d = 0

  Number of outliers:  4
  Outlier statistics:
  Time point    Outlier type
           8               2
```

```
           13              0
           97              0
          109              0

 AIC =    397.5339
 RSE =   0.3966153


 ARMA parameters:
   1       0.481449
   2       0.813321
   3      -0.043181
   4      -0.220261
   5       0.199172
   6       0.199179

                * * * Forecast Table * * *
          Series     Forecast  Prob. Limits  Psi Weights
1         8.7000       9.0273        0.7774       0.8133
2         8.6000       9.0309        1.0020       0.6183
3         9.3000       9.3195        1.1113       0.2475
4         9.1000       9.4767        1.1278       0.1946
5         8.8000       9.4176        1.1379       0.3726
6         8.5000       9.2256        1.1742       0.5253
```

Example 2

This is the same as Example 1, except now **AUTO_ARIMA** uses Method 2 with a possible seasonal adjustment. As a result, the unadjusted model with $p = 3$, $q = 2$, $s = 1$, $d = 0$ is chosen as optimum.

```
      use auto_arima_int
      use wrrrl_int
      use umach_int

      implicit none
!                                 Specifications for parameters
      integer :: nobs, mxlead, i, noutliers, nout
      integer, dime nsion(4) :: model
      integer, dimension(2) :: iper = (/ 1,2 /)
      integer, dimension(3) :: iord = (/ 0,1,2 /)
      integer, dimension(4) :: ima = (/ 0,1,2,3 /)
      integer, dimension(4) :: iar = (/ 0,1,2,3 /)
      integer, dimension(141) :: times
      integer, dimension(:,:), pointer ::  outlierstat
      real(kind(1e0)) :: aic, rse
      real(kind(1e0)), dimension(:), allocatable :: parameters
      real(kind(1e0)), dimension(6,3) :: outlierfcst
      real(kind(1e0)), dimension(6,4) :: forecast_table
      real(kind(1e0)), dimension(141) :: x
      character (len = 10), dimension(1) :: rlabel
      character (len = 14), dimension(5) :: clabel
      character (len = 10) :: fmt

!                                 Time series data
      x = (/ 12.8,12.2,11.9,10.9,10.6,11.3,11.1,10.4,10.0,9.7,9.7,&
               9.7,11.1,10.5,10.3,9.8,9.8,10.4,10.4,10.0,9.7,9.3,&
                9.6,9.7,10.8,10.7,10.3,9.7,9.5,10.0,10.0,9.3,9.0,&
```

```
                  8.8,8.9,9.2,10.4,10.0,9.6,9.0,8.5,9.2,9.0,8.6,&
                    8.3,7.9,8.0,8.2,9.3,8.9,8.9,7.7,7.6,8.4,8.5,&
                    7.8,7.6,7.3,7.2,7.3,8.5,8.2,7.9,7.4,7.1,7.9,&
                    7.7,7.2,7.0,6.7,6.8,6.9,7.8,7.6,7.4,6.6,6.8,&
                    7.2,7.2,7.0,6.6,6.3,6.8,6.7,8.1,7.9,7.6,7.1,&
                  7.2,8.2,8.1,8.1,8.2,8.7,9.0,9.3,10.5,10.1,9.9,&
                  9.4,9.2,9.8,9.9,9.5,9.0,9.0,9.4,9.6,11.0,10.8,&
               10.4,9.8,9.7,10.6,10.5,10.0,9.8,9.5,9.7,9.6,10.9,&
                  10.3,10.4,9.3,9.3,9.8,9.8,9.3,8.9,9.1,9.1,9.1,&
                          10.2,9.9,9.4,8.7,8.6,9.3,9.1,8.8,8.5 /)

      times = (/                      1,2,3,4,5,6,7,8,9,10,11,12,&
                            13,14,15,16,17,18,19,20,21,22,23,24,&
                            25,26,27,28,29,30,31,32,33,34,35,36,&
                            37,38,39,40,41,42,43,44,45,46,47,48,&
                            49,50,51,52,53,54,55,56,57,58,59,60,&
                            61,62,63,64,65,66,67,68,69,70,71,72,&
                            73,74,75,76,77,78,79,80,81,82,83,84,&
                            85,86,87,88,89,90,91,92,93,94,95,96,&
                     97,98,99,100,101,102,103,104,105,106,107,108,&
                  109,110,111,112,113,114,115,116,117,118,119,120,&
                  121,122,123,124,125,126,127,128,129,130,131,132,&
                           133,134,135,136,137,138,139,140,141 /)

      rlabel(1) = 'NUMBER'
      clabel(1) = '   '
      clabel(2) = 'Series'
      clabel(3) = 'Forecast'
      clabel(4) = 'Prob. Limits'
      clabel(5) = 'Psi Weights'

      mxlead = 6
      nobs = 135
      call auto_arima (times, x, parameters, NOBS=nobs, MAXLAG=5,&
                       MODEL=model, AIC=aic,CRITICAL=4.0,&
                       NOUTLIERS=noutliers, IMETH=2,&
                       IAR=iar, IMA=ima, IPER=iper, IORD=iord,&
                       IOUTLIERSTATS=outlierstat, RSE=rse,&
                       MXLEAD=mxlead, OUTLIERFCST=outlierfcst)

      call umach (2, nout)
      write (nout,*) 'Method 2: Grid search, differencing allowed'
      write (nout,FMT="(T2,4(A,I2))") 'Model chosen: p =', model(1),&
            ', q =', model(2), ', s =', model(3), ', d =', model(4)
      write (nout,*)
      write (nout,FMT="(T2,A,I2)") 'Number of outliers: ', noutliers
      write (nout,*) 'Outlier statistics:'
      write (nout,*) 'Time point    Outlier type'
      do i=1,noutliers
        write (nout, FMT="(I11, T15, I13)") outlierstat(i,1),&
              outlierstat(i,2)
      end do
      write (nout,*)
      write (nout,*) 'AIC = ', aic
      write (nout,*) 'RSE = ', rse
      write (nout,*)
      write (nout,*) 'ARMA parameters:'
      do i=1, 1+model(1)+model(2)
        write (nout,FMT="(T3,I2,TR5,f10.6)") i, parameters(i)
```

```
      end do
      write (nout,*)
      forecast_table(1:mxlead,1) = x(nobs+1:nobs+mxlead)
      forecast_table(1:mxlead,2:4) = outlierfcst(1:mxlead, 1:3)
      fmt = '(F11.4)'
      call wrrrl('* * * Forecast Table * * *', forecast_table,&
                 rlabel, clabel, FMT = fmt)
      end
```

Output

```
 Method 2: Grid search, differencing allowed
 Model chosen: p = 3, q = 2, s = 1, d = 0

 Number of outliers:  1
 Outlier statistics:
 Time point    Outlier type
        109               0

 AIC =    408.0768
 RSE =   0.4124085

 ARMA parameters:
   1       0.509427
   2       1.944686
   3      -1.901132
   4       0.901670
   5       1.113016
   6      -0.915008

                * * * Forecast Table * * *
          Series     Forecast  Prob. Limits  Psi Weights
1         8.7000       9.1109        0.8083       0.8317
2         8.6000       9.1811        1.0513       0.6312
3         9.3000       9.5185        1.1686       0.5481
4         9.1000       9.7804        1.2497       0.6157
5         8.8000       9.7117        1.3452       0.7245
6         8.5000       9.3842        1.4671       0.7326
```

Example 3

This example is the same as Example 1 but now Method 3 with the optimum model parameters $p = 3$, $q = 2$, $s = 1$, $d = 0$ from Example 2 is chosen for outlier detection and forecasting.

```
      use auto_arima_int
      use wrrrl_int
      use umach_int

      implicit none
!                                  Parameter specifications
      integer :: nobs, mxlead, i, noutliers, nout
      integer, dimension(4) :: model
      integer, dimension(141) :: times
      integer, dimension(:,:), pointer :: outlierstat
      real(kind(1e0)) :: aic, rse
      real(kind(1e0)), dimension(141) :: x
      real(kind(1e0)), dimension(6,3) :: outlierfcst
```

```
      real(kind(1e0)), dimension(6,4) :: forecast_table
      real(kind(1e0)), dimension(:), allocatable :: parameters
      character (len = 10), dimension(1) :: rlabel
      character (len = 14), dimension(5) :: clabel
      character (len = 10) :: fmt
!                                Time series data
      x = (/ 12.8,12.2,11.9,10.9,10.6,11.3,11.1,10.4,10.0,9.7,9.7,&
               9.7,11.1,10.5,10.3,9.8,9.8,10.4,10.4,10.0,9.7,9.3,&
                9.6,9.7,10.8,10.7,10.3,9.7,9.5,10.0,10.0,9.3,9.0,&
                   8.8,8.9,9.2,10.4,10.0,9.6,9.0,8.5,9.2,9.0,8.6,&
                     8.3,7.9,8.0,8.2,9.3,8.9,8.9,7.7,7.6,8.4,8.5,&
                     7.8,7.6,7.3,7.2,7.3,8.5,8.2,7.9,7.4,7.1,7.9,&
                     7.7,7.2,7.0,6.7,6.8,6.9,7.8,7.6,7.4,6.6,6.8,&
                     7.2,7.2,7.0,6.6,6.3,6.8,6.7,8.1,7.9,7.6,7.1,&
                   7.2,8.2,8.1,8.1,8.2,8.7,9.0,9.3,10.5,10.1,9.9,&
                   9.4,9.2,9.8,9.9,9.5,9.0,9.0,9.4,9.6,11.0,10.8,&
                10.4,9.8,9.7,10.6,10.5,10.0,9.8,9.5,9.7,9.6,10.9,&
                   10.3,10.4,9.3,9.3,9.8,9.8,9.3,8.9,9.1,9.1,9.1,&
                            10.2,9.9,9.4,8.7,8.6,9.3,9.1,8.8,8.5/)

      times = (/                       1,2,3,4,5,6,7,8,9,10,11,12,&
                              13,14,15,16,17,18,19,20,21,22,23,24,&
                              25,26,27,28,29,30,31,32,33,34,35,36,&
                              37,38,39,40,41,42,43,44,45,46,47,48,&
                              49,50,51,52,53,54,55,56,57,58,59,60,&
                              61,62,63,64,65,66,67,68,69,70,71,72,&
                              73,74,75,76,77,78,79,80,81,82,83,84,&
                              85,86,87,88,89,90,91,92,93,94,95,96,&
                    97,98,99,100,101,102,103,104,105,106,107,108,&
                109,110,111,112,113,114,115,116,117,118,119,120,&
                121,122,123,124,125,126,127,128,129,130,131,132,&
                              133,134,135,136,137,138,139,140,141/)

      mxlead = 6
      nobs = 135
      model = (/ 3, 2, 1, 0 /)

      rlabel(1) = 'NUMBER'
      clabel(1) = '   '
      clabel(2) = 'Series'
      clabel(3) = 'Forecast'
      clabel(4) = 'Prob. Limits'
      clabel(5) = 'Psi Weights'

      call auto_arima (times, x, parameters, NOBS=nobs, MODEL=model,&
               AIC=aic, CRITICAL=4.0, NOUTLIERS=noutliers, IMETH=3,&
               IOUTLIERSTATS=outlierstat, RSE=rse, MXLEAD=mxlead,&
               OUTLIERFCST=outlierfcst)

      call umach (2, nout)
      write (nout,*) 'Method 3: Specified ARIMA model'
      write (nout,FMT="(T2,4(A,I2))") 'Model: p =',model(1),', q =',&
                     model(2), ', s =', model(3), ', d =', model(4)
      write (nout,*)
      write (nout,FMT="(T2,A,I2)") 'Number of outliers: ', noutliers
      write (nout,*) 'Outlier statistics:'
      write (nout,*) 'Time point   Outlier type'
      do i=1,noutliers
        write (nout,FMT="(I11,T15,I12)") outlierstat(i,1),&
                     outlierstat(i,2)
```

```
        end do
        write (nout,*)
        write (nout,*) 'AIC=', aic
        write (nout,*) 'RSE=', rse
        write (nout,*)
        write (nout,*) 'ARMA parameters:'
        do i=1, 1+model(1)+model(2)
          write (nout,FMT="(T3,I2,TR5,f10.6)") i, parameters(i)
        end do
        forecast_table(1:mxlead,1) = x(nobs+1:mxlead)
        forecast_table(1:mxlead,2:4) = outlierfcst(1:mxlead,1:3)
        write (nout,*)
        fmt = '(F11.4)'
        call wrrrl ('* * * Forecast Table * * *', forecast_table,&
                  rlabel, clabel, FMT = fmt)
        end
```

Output

```
 Method 3: Specified ARIMA model
 Model: p = 3, q = 2, s = 1, d = 0

 Number of outliers:  1
 Outlier statistics:
 Time point   Outlier type
       109              0

 AIC=   408.0768
 RSE=  0.4124085

 ARMA parameters:
   1       0.509427
   2       1.944686
   3      -1.901132
   4       0.901670
   5       1.113016
   6      -0.915008

                * * * Forecast Table * * *
          Series     Forecast  Prob. Limits  Psi Weights
 1       8.7000       9.1109        0.8083       0.8317
 2       8.6000       9.1811        1.0513       0.6312
 3       9.3000       9.5185        1.1686       0.5481
 4       9.1000       9.7804        1.2497       0.6157
 5       8.8000       9.7117        1.3452       0.7245
 6       8.5000       9.3842        1.4671       0.7326
```

AUTO_FPE_UNI_AR

Automatic selection and fitting of a univariate autoregressive time series model using Akaike's Final Prediction Error (FPE) criteria. Estimates of the autoregressive parameters for the model with minimum FPE are calculated using the methodology described in Akaike, H., et. al (1979).

Required Arguments

MAXLAG — Maximum lag of the sample autocovariances for the stationary time series, *W*. (Input)

ACV — Vector of length MAXLAG + 1 containing the sample autocovariances of *W*. The first element, ACV(0) must be the sample variance of the series and the remaining elements, ACV(1), ..., ACV(MAXLAG), contain the autocovariances of the series for lags 1 through MAXLAG. (Input)

NOBS — Number of observations in the time series. (Input)

NPAR — Number of autoregressive parameters in the the selected model. (Output)

PAR — Vector of length MAXLAG containing estimates for the autoregressive parameters in the model with the minimum Final Prediction Error. The estimates are in the first NPAR values of this vector. The remaining values are set to 0. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

FPE — Final Prediction Error for fitted model. (Output)

CHISQ — Chi-square statistic, with 1 degree of freedom, for the selected model. CHISQ is used to examine the significance of the fitted model. (Output)

AVAR — Estimate of noise variance. (Output)

FORTRAN 90 Interface

Generic: CALL AUTO_FPE_UNI_AR (MAXLAG, ACV, NOBS, NPAR, PAR [, ...])

Specific: The specific interface names are S_AUTO_FPE_UNI_AR and D_AUTO_FPE_UNI_AR.

Description

This routine is based upon the FPEAUT program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972).

The Final Prediction Error for an autoregressive model with lag *k* is defined as:

$$FPE_{\kappa} = \left[1 + \frac{k+1}{N} \right] \cdot r_k$$

where N = NOBS and r_k is the minimum of

$$r_k = E\left[W_t - \sum_{m=1}^{k} \phi_m^{(k)} W_{t-m} - \phi_0^{(k)} \right]^2$$

with respect to the autoregressive coefficients

$$\left\{ \phi_m^{(k)};\ m = 0,\ 1,\ \cdots,\ k \right\}$$

$FPE_{\kappa} = \left[1 + \frac{k+1}{N} \right] \cdot r_k$ is estimated using the formula:

$$FPE_k = \frac{(N + (k+1))}{(N - (k+1))} \cdot \sigma_k^2$$

where σ_k^2 is calculated from the recursive relationship:

$$\sigma_k^2 = \sigma_{k-1}^2 (1 - (\hat{\phi}_k \cdot k)^2), \text{ where } \hat{\phi}_0^2 = 0 \text{ and } \sigma_0^2 = \frac{1}{N} \sum_{t=1}^{N} (W_t - \overline{W})^2$$

The model selected and parameter estimates vary depending upon the value of MAXLAG. Akaike and Nakagawa (1972) recommend that MAXLAG start with values between $2\sqrt{N}$ and $3\sqrt{N}$.

In every case, however, MAXLAG must be strictly less than $N/2$.

As MAXLAG is increased numerical accuracy decreases. It is even possible numerically for the estimated FPE to become negative. If this happens, use double precision.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. In this example, **AUTO_FPE_UNI_AR**, found the minimum FPE solution is an autoregressive model with 10 lags. This is slightly different from the optimum solution found by **AUTO_UNI_AR**, using minimum AIC instead of FPE.

The solution reported by **AUTO_UNI_AR** is an AR model with 2 lags.

$$w_t = \phi_0 + \phi_1 w_{t-1} + \phi_2 w_{t-2} + a_t$$

Using the formula

$$\phi_0 = \mu\left(1 - \sum_{i=1}^{NPAR} \phi_i\right)$$

we obtain the following representation for this series.

$$w_t = 0.32\mu + 1.318 w_{t-1} - 0.635 w_{t-2} + a_t$$

```
      use auto_fpe_uni_ar_int
      use wrrrn_int
      use gdata_int
      use acf_int
      implicit none
!                                  SPECIFICATIONS FOR PARAMETERS
      integer, parameter   :: maxlag=20, nobs=100
      integer              :: npar
      real(kind(1e0))      :: fpe, chisq
      real(kind(1e0))      :: ac(maxlag+1), avar
      real(kind(1e0))      :: acv(maxlag+1), par(maxlag)
      real(kind(1e0))      :: x(176,2)
      integer              :: ncol, nrow
!
!                                  Get Wolfer Sunspot Data
      call gdata(2,x,nrow,ncol)
!
      npar = 20
      write(*,*) 'Univariate FPE Automatic Order selection '
      write(*,*) ' '
!                                  Compute the autocovariances
      call acf (x(22:,2), maxlag, ac, acv=acv, nobs=nobs)
!                                  Example #1
      call auto_fpe_uni_ar(maxlag, acv, nobs, npar, par,  &
                           fpe=fpe, chisq=chisq, avar=avar)
!
      write(*,*) 'Minimum FPE = ', fpe
      write(*,*) 'Chi-squared = ', chisq
      write(*,*) 'Avar = ', avar
      call wrrrn('AR Coefficients', par, nra=npar, nca=1, lda=npar)
!
```

```
end
```

Output

```

Univariate FPE Automatic Order selection

Minimum FPE =  306.91787
Chi-squared =  39.056877
Avar = 289.03915

AR Coefficients
   1   1.318
   2  -0.635
```

AUTO_PARM

Estimates structural breaks in non-stationary univariate time series.

Required Arguments

Y — Array containing the time series. (Input)

NPCS — Number of requested/estimated pieces or segments of the time series. NPCS is considered input when IFITONLY = 1. (Input/Output)

ARP — A pointer to an array of size NPCS × 2. ARP is considered input when IFITONLY = 1. (Input/Output)

Column Index	Description
1	Requested/estimated break points
2	AR order for each segment

ARFIT — A pointer to an array of size NPCS × MAXARORDER containing the AR coefficient estimates for each segment. ARFIT (i,j) is the *j*-th coefficient for segment *i* where *i*=1, NPCS and *j*=l, ARP(i,2). (Output)
Note that the intercept is not reported.

ARSTAT — A pointer to an array of size NPCS × 2. (Output)

Column Index	Description
1	Likelihood values for each of the fitted AR models
2	Residual variances for each of the fitted AR models

SCVAL — Final value of the selection criterion. (Output)

Optional

MAXARORDER—Maximum order to consider for each AR model. (Input)
Default: MAXARORDER = 20.

IMTH — Method of estimation. (Input)

Value	Method
0	Yule-Walker
1	Least Squares
2	Burg

Default: `IMTH` = 0.

ISELCRI — Selection criterion. (Input)

	Method
0	Minimum Description Length (MDL)
1	Akaike's Information Criterion (AIC)

Default: `ISELCRI` = 0.

ILIKE — Likelihood computation method. (Input)

	Method
0	Exact
1	Approximate

Default: `ILIKE` = 0.

IFITONLY — Option to only fit the specified model. (Input)

	Action
0	No, do all the computations
1	Yes, only fit the specified model

Default: `IFITONLY` = 0.

IPRINT — Printing option. (Input)

	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: `IPRINT` = 0.

ISEED — Seed of the random number generator. (Input)

For the same data and parameter settings, AUTO_PARM will return the same results each time if ISEED = 1, 2 ... 2147483646. If ISEED = 0, the system clock will be used to generate a seed. The result will be nondeterministic.

Default: ISEED = 0.

Note: The following arguments are for setting up and running the embedded Genetic Algorithm. In most situations, the default values should be used for these arguments. Users may wish to change some or all for testing or research purposes.

PDISTN — Array of length MAXARORDER + 1 giving the probability distribution over the AR order variable *p* = 0,..., MAXARORDER. (Input)

j = 1,..., MAXARORDER + 1 is used to randomly assign an AR order to breakpoint position *j* for a given chromosome. PDISTN(*j*) > = 0 and if SUM(PDISTN) is not equal to 1, the values will be normalized, i.e., PDISTN(*j*) = PDISTN(*j*)/SUM(PDISTN).

Default**:** PDISTN (*j*) = 1/(MAXARORDER + 1) for all *j*.

MSPAN — Array of length MAXARORDER + 1 containing minimum number of observations required for valid estimates of AR model with order *p* = 0, ..., MAXARORDER. (Input)

Default: MSPAN (*p*+ 1) = 2 * (number of parameters) + 2 = 2 * (*p* + 2) + 2.

GAPARM — Array of length 4 containing parameters that control the behavior of the genetic algorithm. These values should be strictly greater than zero and less than one to avoid unexpected results. (Input)

***GAPARM* (1)** — Probability used to set initial break points in a chromosome.

Default: MIN (MSPAN) / size(Y).

***GAPARM* (2)** — Probability of crossover used to decide between a crossover and a mutation.

Default**:** 1 – MIN (MSPAN) / size(Y).

***GAPARM* (3)** — In the mutation operation, probability an AR(*p*) model is enforced at the current position.

Default: 0.4.

***GAPARM* (4)** — In the mutation operation, probability a break point is disallowed at the current position.

Default**:** 0.3.

Note: GAPARM(3) and GAPARM(4) must be valid probabilities and their sum must be between 0 and 1. 1 – GAPARM(3)– GAPARM(4) is the probability that the chromosome *j* inherits the parent's chromosome *j*.

ISLAND — Array of length 5 containing the migration policy parameters. (Input)

***ISLAND* (1)** — Number of islands.
Default**:** 40.

***ISLAND* (2)** — Number of generations that pass before migration occurs. Note that the convergence of the algorithm is revised whether migrations take place or not (see argument `ISLAND(5)`).
Default: 5.

***ISLAND* (3)** — Number of subjects that migrate at each migration event.
Default: 2.

***ISLAND* (4)** — Population size (number of chromosomes) per island.
Default: 40.

***ISLAND* (5) =** Migration flag. If 1, migration is performed.
Default: 1.

MAXMIG — Maximum number of times that migrations may take place before the routine is stopped if convergence has not occurred. (Input)
Default: `MAXMIG` = 20.

STOPITERS — Number of iterations. The routine will declare convergence and stop the iterations if the criterion value (MDL/AIC) has not changed after `STOPITERS` consecutive migrations. Otherwise, the algorithm will declare non-convergence and stop after `MAXMIG` migrations have taken place. See also `MAXMIG` and `ISLAND(2)`. Note that logically, `STOPITERS` < `MAXMIG`. (Input)
Default: `STOPITERS` = 10.

Interface

Generic: `CALL AUTO_PARM` (`Y`, `NPCS`, `ARP`, `ARFIT`, `ARSTAT`, `SCVAL` `[`, ...`]`)
Specific: The specific interface names are `S_AUTO_PARM` and `D_AUTO_PARM`.

Description

Routine `AUTO_PARM` estimates the structural breaks of a non-stationary time series using, with permission from the Authors, the method developed by Davis et al (2006). `AUTO_PARM` estimates a partition of the time index and models the time series in each segment as a separate auto-regressive (AR(*p*)) process. The routine returns the estimated breakpoints, the estimated AR(*p*) models, and supporting statistics.

For the observed time series , Y_t, $t = 1,...n$ the problem is to find *m*(*m*+1 = `NPCS`), the number of breaks, their locations, $1 < \tau_1 < \tau_2 < \cdots < \tau_m < n$ (`ARP(:,1)`), and P_j(`ARP(:,2)`), *j* = 1, ..., *m* + 1, the order of the AR process in which the *j*-th segment is modeled. That is, $Y_t = X_{t,j}$ for $\tau_{j-1} \le t < \tau_j$ (for convenience, $\tau_0 := 1$ and $\tau_{m+1} := n + 1$) where $\{X_{t,j}\}$ is an AR process of order P_j.

$$X_{t,j} = \phi_{1,j}X_{t-1,j} + \cdots + \phi_{p_j,j}X_{t-p_j,j} + \sigma_t\varepsilon_t$$

and ε_t, the noise sequence, is i.i.d. with mean 0 and variance 1. Note that a series with m breaks will have $m + 1$ segments.

The vector $\{m,\tau_1,\ldots\tau_m,p_1,\ldots p_{m+1}\}$ completely specifies a piecewise AR model. To estimate this vector **AUTO_PARM** optimizes, with respect to this vector, one of two selection critieria: the first is a Minimum Description Length (MDL) criterion, and the second is the Akaike's Information Criterion (AIC). The MDL is defined as

$$\begin{aligned} &MDL(m,\tau_1,\ldots\tau_m,p_1,\ldots p_{m+1}) \\ &\quad = \ln\, m + (m-1)\ln\, n + \sum_{j=1}^{m+1}\frac{2+p_j}{2}\ln\, n_j - \ln L \end{aligned}$$

while the AIC criterion is given by

$$\begin{aligned} AIC(m,\tau_1,\ldots\tau_m,p_1,.p_{m+1}) &= 2(\text{number of parameters}) - 2\ln\, L \\ &= 2(1 + m + \sum_{j=1}^{m+1}(2 + p_j)) - 2\ln\, L \end{aligned}$$

where, given a candidate value of the vector $\{m,\tau_1,\ldots\tau_m,p_1,\ldots p_{m+1}\}$, L is the likelihood of the fitted piecewise AR model evaluated at the parameter estimates,

$$\{\hat{\gamma}_1,\hat{\phi}_{1,1},\ldots\hat{\phi}_{1,p_1},\hat{\sigma}_1^2,\ldots\hat{\gamma}_{m+1},\hat{\phi}_{1,m+1},\ldots\hat{\phi}_{1,p_{m+1}},\hat{\sigma}_{m+1}^2\}.$$

The parameters $\{\gamma_j,\phi_{1,1},\ldots\phi_{1,p_j},\sigma_j^2\}$ of the j-th AR segment are estimated by the choice of one of three estimation methods: Yule-Walker, Burg, or Least Squares.

For simplicity, assume the mean of each series $\{X_{t,j}\}$ is 0 and that the errors are Gaussian. Then, the piecewise AR model has Gaussian likelihood

$$L = \prod_{j=1}^{m+1}(2\pi)^{-n_j/2}|V_j|^{-1/2}\exp\{-Y_j^T V_j Y_j\}$$

where V_j is the variance-covariance of the j-th AR segment (of order p_j) and Y_j is the vector of observations of the j-th segment, i.e., $Y_j = (Y_{\tau_j},Y_{\tau_j+1},\ldots Y_{\tau_{j+1}-1})^T$

To find the minimizer $\{m, \tau_j, p_j\}^*$ of either MDL or AIC, **AUTO_PARM** employs a Genetic Algorithm with islands, migration, cross-over and mutations. See Davis et.al. (2006) for further details.

Comments

1. AUTO_PARM approximates locally stationary time series by independent auto-regressive processes. Experimental results suggest that AUTO_PARM gives reasonable estimates of the structural breaks of a given time series, even if the segment series are not autoregressive. Also, based on experimental results, MDL gives better results than AIC as a selection criterion.

2. Informational Error

Type	Code	Description
3	1	ISEED has been set out of range. ISEED is being reset to 123457.
3	2	MAXMIG migrations were reached in the genetic algorithm before the selection criterion value converged. Try increasing MAXMIG or using the double precision routine.

Examples

The examples below illustrate different scenarios using AUTO_PARM. The example series used in each case is the airline demand data (Box, Jenkins and Reinsel, 1994), which gives monthly total demand for the period January 1949 through December 1960. Each scenario sets the optional argument, ISEED = 123457.

```
      use gdata_int
      use auto_parm_int
      use diff_int
      use umach_int
      implicit none
!                                 Specifications for local variables
!                                 arguments for auto_parm
      integer :: npcs, maxarorder, ifitonly, iprint, island(5), iseed
      integer, pointer :: arp(:,:)
      real(kind(1e0)) :: x(144,1), scval
      real(kind(1e0)), pointer, dimension(:,:) :: arfit, arstat
!                                 Other locals
      integer :: n, nvar, idata, nlost, iper(1), iord(1), nout
      real(kind(1e0)) :: dx(144,1)

      call umach(2, nout)
!                                 Get the data
      iseed = 123457
      idata = 4
      nullify(arp)
      nullify(arfit)
      nullify(arstat)
      call gdata(idata, x, n, nvar)

!                                 Example 1: Use defaults and print
!                                 final results
      write(nout,*)'Example 1: Use defaults'
      write(nout,*)
```

```
      call auto_parm(x(:,1), npcs, arp, arfit, arstat, scval, &
                     iseed=iseed, iprint=1)

!                                  Example 2: Differenced series
!                                  set period for the difference.
!                                  iper is in years for this data set
      write(nout,*)
      write(nout,*)'Example 2: Differenced series'
      write(nout,*)
!                                  Set the order for the difference.
      iper(1) = 1
      iord(1) = 1
!                                  Get differenced series dx
      call diff(x(:,1),iper,iord,n,dx(:,1),nlost=nlost)
!                                  Compare results on the differenced
!                                  series
      deallocate(arp)
      deallocate(arfit)
      deallocate(arstat)
      call auto_parm(dx((1+nlost):n,1), npcs, arp, arfit, arstat, &
                     scval, iseed=iseed, iprint=1)

!                                 Example 3: Original series
!                                 lower maximum AR order
      write(nout,*)
      write(nout,*)'Example 3: Original series, lower order allowed'
      write(nout,*)
      maxarorder = 5
      deallocate(arp)
      deallocate(arfit)
      deallocate(arstat)
      call auto_parm(x(:,1), npcs, arp, arfit, arstat, scval, &
                     maxarorder=maxarorder, iseed=iseed, iprint=1)

!                                  Example 4: differenced series, lower
!                                  maximum AR order
      write(nout,*)
      write(nout,*)'Example 4: Differenced series, lower maximum'// &
                   ' AR order'
      write(nout,*)
      deallocate(arp)
      deallocate(arfit)
      deallocate(arstat)
      call auto_parm(dx((1+nlost):n,1), npcs, arp, arfit, arstat, &
                     scval, maxarorder=maxarorder, iseed=iseed, &
                     iprint=1)

!                                  Example 5: Original series, force
!                                             fit the segments
!                                  Fit the specified model only at the
!                                  break points
      write(nout,*)
      write(nout,*)'Example 5: Original series'
      write(nout,*)'Force fit the segments at the break points'
      write(nout,*)
      npcs = 2
      deallocate(arp)
      allocate(arp(npcs,2))
      arp(1,2) =  2
      arp(2,2) =  1
```

```
      arp(1,1) =  1
      arp(2,1) = 60
      deallocate(arfit)
      deallocate(arstat)
      call auto_parm(x(:,1), npcs, arp, arfit, arstat, scval, &
                     ifitonly=1, iseed=iseed, iprint=1)
      end
```

Output

```
 Example 1: Use defaults

    ============== final results ===============
 number of pieces:            2

 selection criteria value:   684.242

 total time:  0.7198125     conv:             1

 ==================== final model estimates =====================
 break point   order     est. coeff.    likelihood     resid. var
  ARP( 1,1)  ARP( 1,2)   ARFIT( 1,:)    ARSTAT( 1,1)   ARSTAT( 1,2)
    1           1        0.77542
                                        186.945        355.025
  ARP( 2,1)  ARP( 2,2)   ARFIT( 2,:)    ARSTAT( 2,1)   ARSTAT( 2,2)
   44          13        1.03701
                        -0.07802
                        -0.03890
                        -0.03452
                         0.11961
                        -0.12852
                         0.01990
                        -0.04886
                         0.08090
                        -0.13118
                         0.22122
                         0.53863
                        -0.61515
                                        486.665        691.471

 Example 2: Differenced series

    ============== final results ===============
 number of pieces:            1

 selection criteria value:   624.284

 total time:  0.7204375     conv:             1

 ==================== final model estimates =====================
 break point   order     est. coeff.    likelihood     resid. var
  ARP( 1,1)  ARP( 1,2)   ARFIT( 1,:)    ARSTAT( 1,1)   ARSTAT( 1,2)
    1          12        -0.02842
                        -0.22436
                        -0.16846
                        -0.24267
                        -0.10573
                        -0.22429
```

```
                        -0.12126
                        -0.26446
                        -0.07087
                        -0.24327
                        -0.07136
                         0.57129
                                          619.321        297.351

 Example 3: Original series, lower order allowed

    ============== final results ===============
 number of pieces:            2

selection criteria value:   705.297

 total time:  0.3136875     conv:             1

 ==================== final model estimates =====================
 break point   order     est. coeff.    likelihood     resid. var
  ARP( 1,1)  ARP( 1,2)   ARFIT( 1,:)    ARSTAT( 1,1)   ARSTAT( 1,2)
    1           1        0.89533
                                          270.393        333.564
  ARP( 2,1)  ARP( 2,2)   ARFIT( 2,:)    ARSTAT( 2,1)   ARSTAT( 2,2)
   63           2        1.19788
                        -0.35922
                                          424.270       1632.337

 Example 4: Differenced series, lower maximum AR order

    ============== final results ===============
 number of pieces:            2

selection criteria value:   698.359

 total time:  0.2981875     conv:             1

 ==================== final model estimates =====================
 break point   order     est. coeff.    likelihood     resid. var
  ARP( 1,1)  ARP( 1,2)   ARFIT( 1,:)    ARSTAT( 1,1)   ARSTAT( 1,2)
    1           0        -------
                                          335.565        357.388
  ARP( 2,1)  ARP( 2,2)   ARFIT( 2,:)    ARSTAT( 2,1)   ARSTAT( 2,2)
   77           1        0.33310
                                          352.175       1786.345

 Example 5: Original series
 Force fit the segments at the break points

    ============== final results ===============
 number of pieces:            2

selection criteria value:   712.521

 ==================== final model estimates =====================
 break point   order     est. coeff.    likelihood     resid. var
  ARP( 1,1)  ARP( 1,2)   ARFIT( 1,:)    ARSTAT( 1,1)   ARSTAT( 1,2)
    1           2        1.12156
                        -0.24876
```

```
                                                258.192        313.889
      ARP( 2,1)  ARP( 2,2)   ARFIT( 2,:)    ARSTAT( 2,1)   ARSTAT( 2,2)
       60            1        0.88605
                                                443.696       1937.635
```

AUTO_MUL_AR

Automatic selection and fitting of a multivariate autoregressive time series model. This is the multivariate version of AUTO_UNI_AR. The lag for the model is automatically selected using Akaike's Information Criterion (AIC).

Required Arguments

X — NOBS by NCHANX matrix containing the stationary multivariate time series. (Input)
Each row of X corresponds to an observation of a multivariate time series, and each column of X corresponds to a univariate time series.

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NMATRIX — Number of autoregressive parameter matrices in the minimum AIC model. (Output)

PAR — Three dimensional NMATRIX by NCHANX by NCHANX array containing estimates for the autoregressive parameters in the model. (Output)

Optional Arguments

NOBS — Number of observations in each time series.
Default: NOBS = size(X,1) (Input)

NCHANX — Number of variables, channels, in the multivariate time series. (Input)
NCHANX is the number of columns in X that should be processed. Default: NCHANX = size(X,2)

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0

AVAR — NCHANX by NCHANX array of estimates of the noise variance. (Output)

NPAR — Vector of length NCHANX containing the number of autoregressive parameters fitted for each time series. (Output)

AIC — Akaike's Information Criterion . (Output)

FORTRAN 90 Interface

Generic: **CALL AUTO_MUL_AR** (X, MAXLAG, NMATRIX, PAR [, ...])

Specific: The specific interface names are **S_AUTO_MUL_AR** and **D_AUTO_MUL_AR**.

Description

The routine **AUTO_MUL_AR** automatically selects the order of the multivariate AR model that best fits the data and then displays the AR coefficients. This procedure is an adaptation of the MULMAR procedure published in the TIMSAC-78 Library by Akaike, H., et. al (1979) and Kitagawa & Akaike (1978).

A multivariate AR model can be expressed as:

$$X_t = \sum_{i=1}^{NMATRIX} A_i \cdot X_{t-i} + U_t$$

where

$$X_t = (x_{1,t}, x_{2,t}, \ldots, x_{NCHANX,t})'$$

is a column vector containing the values for the **NCHANX** univariate time series at time = *t*.

$$A_1, A_2, \ldots, A_{NPAR}$$

are the **NCHANX** by **NCHANX** matrices containing the autoregressive parameter estimates for lags 1, 2, ..., **NMATRIX**. And

$$U_t = (\varepsilon_{1,t}, \varepsilon_{2,t}, \ldots, \varepsilon_{NCHANX,t})'$$

is a column vector containing the values for the **NCHANX** white noise values for each time series at time = *t*.

The best fit AR model is determined by successfully fitting multivariate AR models with 1 to **NMATRIX** autoregressive coefficients. For each model, Akaike's Information Criterion (AIC) is calculated using the formula:

$$AIC = (NOBS - MAXLAG) \cdot \ln(\det(AVAR)) + 2 \cdot K + \left(NOBS - MAXLAG\right)\left(\ln\left(2\pi\right) + 1\right)$$

where

K = number of non-zero autoregressive coefficients in the parameter matrices.

The best fit model is the model with the minimum AIC. If the number of parameters in this model is equal to the highest order autoregressive model fitted, i.e., NMATRIX=MAXLAG, then a model with smaller AIC might exist for larger values of MAXLAG. In this case, increasing MAXLAG to explore AR models with additional autoregressive parameters might be warranted.

Example

Consider the data in northern light activity and earthquakes in Robinson 1967, page 204, for the years 1770 through 1869.

```
      USE GDATA_INT
      USE AUTO_MUL_AR_INT
      USE WRRRN_INT

      IMPLICIT NONE
      INTEGER, PARAMETER :: NOBS=100, MAXLAG=10
      INTEGER NCOL, NROW, NMATRIX, I
      REAL(KIND(1E0)) RDATA(NOBS,4), X(NOBS,3), PAR(MAXLAG,3,3)

      CALL GDATA, RDATA, NROW, NCOL)
      X(:,1) = Rdata(:,3)
      X(:,2) = Rdata(:,4)
      X(:,3) = Rdata(:,2)

      CALL AUTO_MUL_AR(X,MAXLAG,NMATRIX,PAR)
        WRITE(*,*) "PAR = "
DO I=1,NMATRIX
        CALL WRRRN("",PAR(I,:,:))
   ENDDO
      END
```

Output

```
 PAR =

         1       2       3
1   0.822   0.000   0.000
2   0.000   0.000  -0.290
3   0.098   0.000   1.214

          1       2       3
1  -0.1136   0.0000   0.0000
2   0.2489   0.0000   0.0000
3   0.0040   0.0000  -0.8682

          1       2       3
1   0.0000   0.0000   0.0000
2   0.0000   0.0000   0.0000
3   0.0725   0.0000   0.3374


          1       2       3
1   0.0000   0.0000   0.0000
```

```
      2   0.0000   0.0000   0.0000
      3  -0.0693   0.0400  -0.2217

              1        2        3
      1   0.0000   0.0000   0.0000
      2   0.0000   0.0000   0.0000
      3   0.0745  -0.0025   0.1253

              1        2        3
      1   0.0000   0.0000   0.0000
      2   0.0000   0.0000   0.0000
      3  -0.0299   0.0510  -0.1570

              1         2         3
      1   0.00000   0.00000   0.00000
      2   0.00000   0.00000   0.00000
      3   0.00806   0.05025   0.04558

              1         2         3
      1   0.00000   0.00000   0.00000
      2   0.00000   0.00000   0.00000
      3   0.03858   0.01929  -0.09825

              1        2        3
      1   0.0000   0.0000   0.0000
      2   0.0000   0.0000   0.0000
      3   0.0904  -0.0154   0.1376

              1         2         3
      1   0.00000   0.00000   0.00000
      2   0.00000   0.00000   0.00000
      3   0.00000  -0.06879   0.00000
```

AUTO_FPE_MUL_AR

Automatic selection and fitting of a multivariate autoregressive time series model using Akaike's Multivariate Final Prediction Error (MFPE) criteria.

Required Arguments

X — NOBS by NCHANX matrix containing the stationary time series. Each row corresponds to an observation in the time series, and each column corresponds to a univariate time series for one of the channels. (Input)

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

NMATRIX — Number of autoregressive parameter matrices in minimum FPE model. (Output)

PAR — MAXLAG by NCHANX by NCHANX array containing estimates for the autoregressive parameters in the selected model. (Output)

Optional Arguments

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0

NOBS — Number of observations in each time series. (Input)
Default = size(X,1).

NCHANX — Number of variables, channels, in the multivariate time series. (Input)
Default = size(X,2).

CCV — NCHANX by NCHANX by MAXLAG+1 matrix containing the sample crosscovariances of the NCHANX time series variables. For the *i*-th time series variable, the first element, CCV(i,i, 1) is the sample variance for the *i*-th series and the remaining elements, CCV(i,i, 2) ... CCV(i,i, MAXLAG+1), contain the autocovariances of the *i*-th series for lags 1 thru MAXLAG. Elements CCV(i,j, 2), ..., CCV(i,j, MAXLAG+1,), contain the autocovariances between the *i*-th and *j*-th series for lags 1 thru MAXLAG (Output)

AVAR — NCHANX by NCHANX matrix containing estimates of the noise variance for each of the NCHANX time series. (Output)

FPE — Multivariate Final Prediction Error for fitted model. (Output)

FORTRAN 90 Interface

Generic: CALL AUTO_FPE_MUL_AR (X, MAXLAG, NMATRIX, PAR [, ...])

Specific: The specific interface names are S_AUTO_FPE_MUL_AR and D_AUTO_FPE_MUL_AR.

Description

The AUTO_FPE_MUL_AR routine is based upon the FPEC program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972). Estimates of the autoregressive parameters for the model with minimum FPE are calculated using the methodology described in Akaike, H., et. al (1979).

The multivariate Final Prediction Error for a multivariate autoregressive model with lag *p* is defined as:

$$MFPE_p = \frac{\left(1 + \frac{pm+1}{N}\right)^m}{\left(1 - \frac{pm+1}{N}\right)^m} \left\| \hat{\Sigma}_p \right\|$$

where $\left\| \hat{\Sigma}_p \right\|$ is the determinant of the estimated NCHANX by NCHANX matrix of covariances of the white noise in the NCHANX series, *m* = NCHANX and *N* = NOBS.

The model selected and parameter estimates vary depending upon the value of MAXLAG. Akaike and Nakagawa (1972) recommend that MAXLAG start with values between $2\sqrt{N} / m$ and $3\sqrt{N} / m$, and that MAXLAG does not exceed $N / (5 \cdot m)$.

In every case, however, MAXLAG must never exceed $N / (2 \cdot m)$.

The numerical accuracy decreases as MAXLAG increases. In this case, it is possible for the estimated FPE to become negative. If this happens , using double percision may help.

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) along with data on northern light and earthquake activity (Robinson 1967, page 204) for each year from 1770 through 1869 to be a 3-channel time series. The following program automatically fits this data to an AR model with a MAXLAG = 10. In this example, AUTO_FPE_MUL_AR selects a multivariate autoregressive model with 2 lags.

```
      use auto_fpe_mul_ar_int
      use wrrrl_int
      use gdata_int
      implicit none
!                                  SPECIFICATIONS FOR PARAMETERS

      integer,parameter     :: nobs=100, nvar=3, maxlag=10, npar=2
      integer               :: nrow, ncol, nmatrix
      real(kind(1e0))       :: ccv(maxlag+1,nvar,nvar)
      real(kind(1e0))       :: x(nobs,nvar+1)
      real(kind(1e0))       :: par(maxlag, nvar, nvar)
      real(kind(1e0))       :: avar(nvar, nvar), aic, fpe

!                                  SPECIFICATIONS FOR LOCAL VARIABLES
      character             :: label(1)*4
      integer               :: i
!                                  EXAMPLE #1
      label(1) = 'NONE'
!                                  GET ROBINSON MULTICHANNEL DATA
      call gdata( 8, x, nrow, ncol)
!
      call auto_fpe_mul_ar(x(1:,2:), maxlag, nmatrix, par,  aic=aic, &
           fpe=fpe, avar=avar)
!
      write(*,*) 'Order Selected: ', nmatrix
      write(*,*) 'FPE = ', fpe
      do i = 1, npar
         call wrrrl('PAR ', par(i, 1:nvar, 1:nvar), label, &
                label, fmt='(F15.10)')
      enddo
      write(*,*) ' '
      call wrrrl('AVAR (White Noise)', avar, label, label,  &
                 fmt='(F15.10)')
      end
```

Output

```
 Order Selected:  2
  FPE =  825727500.0
                          PAR
     1.2989480495     0.0272710621     0.0238259397
     0.0364407972     0.7911528349     0.0851913393
    -0.2183818072    -0.0601412356    -0.0717520714
                          PAR
    -0.6405253410     0.0186619535    -0.0319643840
     0.0082284175    -0.1621686369     0.0587148108
    -0.1242173612     0.3418768048    -0.0425752103

                  AVAR (White Noise)
```

```
      281.7733154297   231.8975372314    57.6690979004
      231.8975982666  1296.4528808594    70.7170333862
       57.6690635681    70.7170333862  1752.6368408203
```

BAY_SEA

Bayesian seasonal adjustment modeling. The model allows for a decomposition of a time series into trend, seasonal, and an error component.

Required Arguments

W — Vector containing the stationary time series. (Input)

Optional Arguments

IORDER — Order of trend differencing. (Input)
Default: `IORDER` = 2.

ISORDER — Order of seasonal differencing. (Input)
Default: `ISORDER` = 1.

NFOCAST — Number of forecasted values. (Input)
Default: `NFOCAST` = 0.

NPERIOD — Number of seasonals within a period. (Input)
Default: `NPERIOD` = 12.

RIGID — Controls rigidity of the seasonal pattern. (Input)
Default: `RIGID` = 1.0.

LOGT — Model option. (Input)

`LOGT`	Model
0	Non-additive model
1	Log additive model

Default: `LOGT` = 0.

IPRINT — Printing option. (Input)

IPRINT	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

ABIC — The Akaike Bayesian Information Criterion for the estimated model. (Output)

TREND — Vector of length size (W) + NFOCAST containing the estimated trend component for each data value followed by the trend estimates for the NFOCAST forecasted values. (Output)

SEASONAL — Vector of length size (W) + NFOCAST containing the estimated seasonal components for each data value followed by the estimates for the NFOCAST forecasted seasonal values. (Output)

COMP — Vector of length size (W) containing the estimated irregular components. (Output)

SMOOTHED — Vector of length size (W) + NFOCAST containing the estimated smoothed estimates for each of the time series values followed by the NFOCAST forecast values. (Output)

FORTRAN 90 Interface

Generic: CALL BAY_SEA (W [, …])

Specific: The specific interface names are S_BAY_SEA and D_BAY_SEA.

Description

Routine BAY_SEA is based upon the algorithm published by Akaike (1980). This algorithm uses a Bayesian approach to the problem of fitting the following autoregressive model for a time series W_t decomposed into a trend and a seasonal component.

Adopting the notation described earlier in the Usage Notes section of this chapter, if

$$t \in ZZ = \{ \cdots, -2, -1, 0, 1, 2, \cdots \}$$

then a seasonal autoregressive model can be represented by the following relationship:

$$W_t = T_t + S_t + A_t$$

where W_t is the stationary time series with mean μ, T_t denotes an underlying trend, S_t seasonal component and A_t a noise or irregular component.

A non-Bayesian approach to this problem would be to estimate the trend and seasonal components by minimizing

$$\sum_{t=1}^{N}\left[\left\{W_t - T_t - S_t\right\}^2 + d^2\left\{\left(T_t - 2T_{t-2} + T_{t-2}\right)^2 + r^2\left(S_t - S_{t-p}\right)^2 + z^2\left(S_t + S_{t-1} + \cdots + S_{t-p}\right)^2\right\}\right]$$

where *p* is the period of the seasonal component, and *d*, *r*, and *z* are properly chosen constants.

In **BAY_SEA**, the approach is to select the parameter *d*, which controls the smoothness of the trend and seasonality estimates, using Bayesian methods. The prior distribution controls the smoothness of the trend and seasonal components by assuming low order Gaussian autoregressive models for some differences of these components. The choice of the variance of the Gaussian distribution is realized by maximizing the log likelihood of the Bayesian model.

The other smoothing parameters, *r* and *z*, are determined by the value of **RIGID**. The default value for **RIGID** is 1. Larger values of **RIGID** produce a more rigid seasonal pattern. Normally, a series is first fit using the default value for **RIGID**. The smoothness of the trend and seasonality estimates are examined and then **RIGID** is either increased or decreased depending upon whether more or less seasonal smoothing is needed.

Additionally, **BAY_SEA** selects the optimum autoregressive model as the model that minimizes **ABIC**.

$$ABIC = -2 \bullet ln(likelihood)\ ,$$

where the *likelihood* in this case is the mixed Bayesian maximum likelihood. Smaller values of *ABIC* represent a better fit.

Example

This example uses unadjusted unemployment for women over 20 years of age in the U.S. for 1991-2001, as reported by the U.S. Bureau of Labor Statistics (**www.bls.gov**). Figure 18 displays this data together with the smoothed preditions from **BAY_SEA**. Figure 19 displays the same data with trend predictions from **BAY_SEA**.

```
          use bay_sea_int
          use wrrrl_int

          IMPLICIT     NONE
    !                                SPECIFICATIONS FOR PARAMETERS
          integer, parameter :: focast=12, nobs=132, nyears=11
          real(kind(1e0))           :: b(nobs)
    !                                U.S. Labor Statistics
    !                                unemployment for women
    !                                over 20 years of age
          data b/ 2968D0,3009D0,2962D0,2774D0,3040D0,3165D0,& !1991
                  3104D0,3313D0,3178D0,3142D0,3129D0,3107D0,&
                  3397D0,3447D0,3328D0,3229D0,3286D0,3577D0,& !1992
                  3799D0,3867D0,3655D0,3360D0,3310D0,3369D0,&
                  3643D0,3419D0,3108D0,3118D0,3146D0,3385D0,& !1993
```

```
                  3458D0,3468D0,3330D0,3244D0,3135D0,3005D0,&
                  3462D0,3272D0,3275D0,2938D0,2894D0,3106D0,& !1994
                  3150D0,3289D0,3136D0,2829D0,2776D0,2467D0,&
                  2944D0,2787D0,2749D0,2762D0,2578D0,2900D0,& !1995
                  3100D0,3102D0,2934D0,2864D0,2652D0,2456D0,&
                  3088D0,2774D0,2701D0,2555D0,2677D0,2741D0,& !1996
                  3052D0,2966D0,2772D0,2723D0,2705D0,2640D0,&
                  2898D0,2788D0,2718D0,2406D0,2520D0,2645D0,& !1997
                  2708D0,2811D0,2666D0,2380D0,2292D0,2187D0,&
                  2750D0,2595D0,2554D0,2213D0,2218D0,2449D0,& !1998
                  2532D0,2639D0,2449D0,2326D0,2302D0,2065D0,&
                  2447D0,2398D0,2381D0,2250D0,2086D0,2397D0,& !1999
                  2573D0,2475D0,2299D0,2054D0,2127D0,1935D0,&
                  2425D0,2245D0,2298D0,2005D0,2208D0,2379D0,& !2000
                  2459D0,2539D0,2182D0,1959D0,2012D0,1834D0,&
                  2404D0,2329D0,2285D0,2175D0,2245D0,2492D0,& !2001
                  2636D0,2892D0,2784D0,2771D0,2878D0,2856D0/

        integer  :: i
        real(kind(1e0)) :: abicm
        real(kind(1e0)) :: trend(nobs+focast), season(nobs+focast)
        real(kind(1e0)) :: irreg(nobs), smooth(nobs+focast)
        character       :: months(focast)*3, years(nyears+2)*4
        data months/'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', &
                    'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'/
        data years/'    ', '1991', '1992', '1993', '1994', '1995', &
                   '1996', '1997','1998','1999','2000','2001','2002'/

        call bay_sea(b, trend=trend, seasonal=season, &
                     comp=irreg, abic=abicm, iorder=2, isorder=1,&
                     nfocast=focast, smoothed=smooth)

        write(*,*) 'ABIC = ', abicm
        call wrrrl('TREND with last 12 values forecasted',trend,months, &
                    years, nra=12,nca=nyears+1,lda=12)

          call wrrrl('SEASONAL with last 12 values forecasted',season,&
                      months,years, nra=12,nca=nyears+1, lda=12)
          call wrrrl('IRREGULAR=Original data-TREND-SEASONAL',irreg, &
                      months,years, nra=12,nca=nyears,lda=12)

        end
```

Output

```
 ABIC =  1297.6403

                        TREND with last 12 values forecasted
          1991     1992     1993     1994     1995     1996     1997     1998
 Jan   2879.8   3318.9   3422.6   3228.7   2827.3   2795.7   2743.1   2481.3
 Feb   2918.2   3359.9   3387.8   3206.0   2815.8   2785.6   2720.6   2469.7
 Mar   2955.1   3399.1   3355.5   3177.0   2812.4   2777.9   2694.2   2455.4
 Apr   2990.0   3436.1   3329.5   3142.1   2814.7   2773.2   2665.2   2439.0
 May   3022.7   3469.0   3309.6   3103.9   2819.3   2771.3   2636.0   2422.8
 Jun   3052.8   3496.1   3294.8   3064.8   2825.5   2771.0   2607.4   2408.5
 Jul   3082.8   3514.5   3283.8   3025.7   2830.9   2772.4   2580.8   2396.8
 Aug   3116.2   3521.9   3276.0   2987.3   2833.2   2773.6   2557.2   2387.7
 Sep   3153.4   3517.8   3270.3   2949.0   2831.7   2774.7   2536.3   2380.3
```

Oct	3193.8	3503.7	3264.8	2911.3	2826.1	2774.5	2518.0	2373.6
Nov	3235.6	3482.4	3256.8	2876.6	2816.6	2770.4	2503.0	2366.6
Dec	3277.6	3455.2	3245.3	2847.6	2805.9	2760.2	2491.5	2359.1

	1999	2000	2001	2002
Jan	2352.1	2235.9	2206.0	3166.1
Feb	2345.6	2237.6	2239.0	3275.1
Mar	2338.1	2239.3	2281.4	3384.1
Apr	2328.3	2238.9	2333.1	3493.0
May	2315.5	2235.2	2393.9	3602.0
Jun	2301.4	2226.3	2464.7	3710.9
Jul	2286.0	2212.8	2545.7	3819.9
Aug	2269.8	2196.6	2636.7	3928.9
Sep	2255.5	2181.3	2735.7	4037.8
Oct	2244.6	2171.9	2840.4	4146.8
Nov	2238.3	2172.1	2948.2	4255.8
Dec	2235.6	2183.3	3057.2	4364.7

SEASONAL with last 12 values forecasted

	1991	1992	1993	1994	1995	1996	1997	1998	1999
Jan	162.9	165.6	169.3	172.0	173.8	176.3	176.4	177.3	176.4
Feb	51.4	51.5	50.5	49.5	48.6	48.8	50.0	51.1	51.0
Mar	-24.0	-23.9	-23.4	-18.8	-16.3	-13.0	-8.7	-4.9	-3.0
Apr	-191.0	-190.1	-189.1	-188.0	-186.6	-187.8	-188.5	-187.9	-186.6
May	-140.6	-143.3	-145.4	-147.4	-148.1	-147.1	-147.1	-147.9	-147.7
Jun	67.2	66.6	65.8	64.4	63.3	62.2	62.7	63.6	64.9
Jul	176.9	180.1	181.6	183.0	185.5	186.6	185.8	186.1	187.2
Aug	251.9	253.0	252.6	253.3	253.1	252.8	253.7	254.4	255.2
Sep	76.4	77.2	77.1	77.3	75.5	73.3	72.6	70.7	68.9
Oct	-79.5	-80.5	-80.1	-80.8	-81.5	-84.3	-87.6	-90.1	-93.4
Nov	-119.8	-120.7	-120.5	-120.2	-120.4	-119.7	-119.7	-118.1	-117.6
Dec	-235.7	-237.9	-243.0	-247.9	-250.5	-251.2	-254.2	-256.2	-257.5

	2000	2001	2002
Jan	177.2	177.6	177.5
Feb	50.8	51.6	51.5
Mar	-1.9	-1.7	-1.8
Apr	-187.2	-186.5	-186.6
May	-145.7	-145.6	-145.7
Jun	65.6	65.1	65.0
Jul	186.4	184.7	184.7
Aug	256.8	256.9	256.9
Sep	67.3	67.0	67.0
Oct	-95.1	-94.6	-94.6
Nov	-117.3	-116.6	-116.6
Dec	-258.1	-257.1	-257.1

IRREGULAR=Original data-TREND-SEASONAL

	1991	1992	1993	1994	1995	1996	1997	1998	1999
Jan	-74.7	-87.5	51.1	61.2	-57.1	116.0	-21.5	91.4	-81.5
Feb	39.4	35.6	-19.3	16.5	-77.4	-60.4	17.3	74.2	1.5
Mar	30.9	-47.2	-224.1	116.8	-47.1	-63.9	32.5	103.5	45.9
Apr	-25.0	-17.0	-22.5	-16.1	133.9	-30.5	-70.6	-38.0	108.3
May	157.8	-39.7	-18.2	-62.6	-93.1	52.8	31.2	-56.9	-81.9
Jun	45.0	14.3	24.4	-23.1	11.2	-92.1	-25.1	-23.2	30.7
Jul	-155.6	104.4	-7.4	-58.6	83.6	93.0	-58.6	-50.9	99.9
Aug	-55.0	92.1	-60.6	48.4	15.7	-60.4	0.1	-3.1	-49.9
Sep	-51.8	60.0	-17.4	109.6	26.8	-76.0	57.1	-2.1	-25.3
Oct	27.7	-63.3	59.3	-1.5	119.4	32.8	-50.3	42.5	-97.1
Nov	13.2	-51.7	-1.3	19.5	-44.3	54.3	-91.3	53.5	6.3

```
      Dec    65.1   151.7     2.7  -132.8   -99.4   131.0   -50.3   -37.9   -43.1

             2000    2001
      Jan    11.9    20.4
      Feb   -43.5    38.4
      Mar    60.5     5.3
      Apr   -46.7    28.5
      May   118.5    -3.3
      Jun    87.1   -37.7
      Jul    59.9   -94.4
      Aug    85.6    -1.6
      Sep   -66.6   -18.7
      Oct  -117.8    25.3
      Nov   -42.7    46.4
      Dec   -91.2    56.0
```

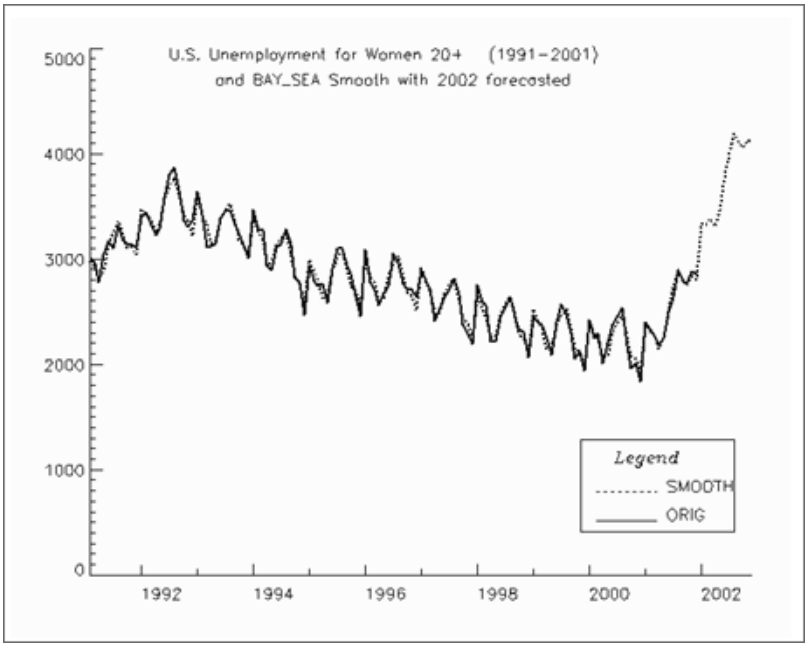


Figure 18, Sample Smoothed Predictions from BAY_SEA

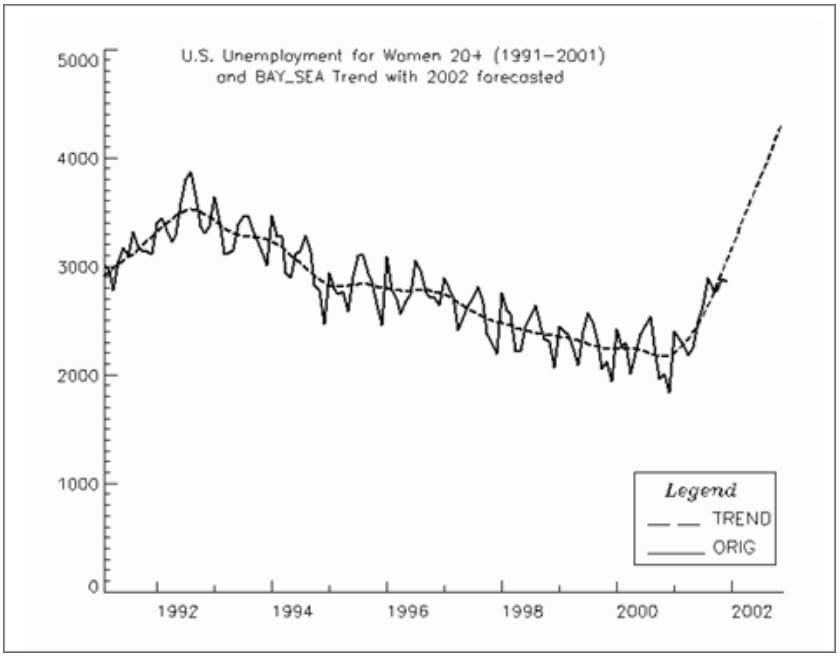


Figure 19, Sample Trend Predictions from BAY_SEA

OPT_DES

more...

Optimal controller design based upon the methodology of Akaike and Nakagaw (1972). The routine allows for multiple channels for both the controlled and manipulated variables. The gain matrix is computed after automatically selecting the best autoregressive model using minimum multivariate final prediction error (MFPE).

Required Arguments

MAXLAG — Maximum number of autoregressive parameters requested. (Input)

X — NOBS by NCHANX matrix containing the multi-channel time series for the manipulated variables, also called control system input variables. Each row corresponds to a different observation in the series, and each column of X corresponds to a univariate time series for one of the controlled channels. (Input)

Y — NOBS by NCHANY matrix containing the multi-channel time series for the controlled variables, also called control system output variables. Each row corresponds to a different observations in the series, and each column of Y corresponds to a univariate time series for one of the manipulated channels. (Input)

NPAR — Number of autoregressive parameter matrices in minimum FPE model. (Output)

GAIN — NCHANX by NPAR gain matrix. (Output)

Optional Arguments

Q — NCHANY by NCHANY non-negative definite symmetric weighting matrix for the quadratic optimization criterion. (Input)
Default: If AVAR is non-singular, Q = Inverse of AVAR. If AVAR is singular, then Q = diag(1/$AVAR_{1,1}$, 1/$AVAR_{2,2}$, ..., 1/$AVAR_{NCHANY,NCHANY}$).

R — A NCHANX by NCHANX positive definite symmetric weighting matrix for the quadratic optimization criterion. (Input)
Default: R = diag(1/σ^2_{X1}, 1/σ^2_{X2}, ..., 1/$\sigma^2_{XNCHANX}$)

IPRINT — Printing option. (Input)

0	No printing
1	Prints final results only
2	Prints intermediate and final results

Default: IPRINT = 0.

NCHANX — Number of time series for manipulated variables. (Input)
Default: NCHANX = size(X,2)

NCHANY — Number of time series for controlled variables. (Input)
Default: NCHANY = size(Y,2)

NOBS — Number of observations in each time series. (Input)
Default: NOBS = size(X,1)

NSTAGE — Number of stages used to compute the gain matrix, GAIN. (Input)
Default: NSTAGE = 20

TRANSY —NPAR by NCHANY by NCHANY transition matrix. (Output)

GAMMAX — NPAR by NCHANY by NCHANX gamma matrix. (Output)

CCV — NCHAN by NCHAN by (MAXLAG+1) matrix containing the sample autocovariances of the NCHAN time series variables, where NCHAN = NCHANX+NCHANY. For the i-th time series variable, the first element, CCV(*i*,*i*,1) is the sample variance for the *i*-th series and the remaining elements, CCV(*i*,*i*,2) … CCV(*i*,*i*,MAXLAG+1), contain the autocovariances of the *i*-th series for lags 1 thru MAXLAG. Elements CCV(*i*,*j*,2) … CCV(*i*,*j*,MAXLAG +1) contain the autocovariances between the *i*-th and *j*-th series for lags 1 through MAXLAG. (Output)

AVAR — NCHANY by NCHANY matrix containing estimates of the noise variances, autocovariances and cross-covariances for the NCHANY time series. (Output)

FPE — Final Prediction Error for fitted model. (Output)

AIC — Akaike's Information Criterion for fitted model. (Output)

FORTRAN 90 Interface

Generic: CALL OPT_DES (MAXLAG, X, Y, NPAR, GAIN [, …])

Specific: The specific interface names are **S_OPT_DES** and **D_OPT_DES**.

Description

The routine **OPT_DES** is based upon the FPEC and OPTDES program published in the TIMSAC –71 Library described by Akaike, H. and Nakagawa, T (1972). Estimates of the autoregressive parameters for the model with minimum multivariate final prediction error (MFPE) are calculated using the methodology described in Akaike, H., et. al (1979). Their methodology produces an estimate of the gain matrix, *G*, in a feedback control system with the following relationship:

$$X_t = GZ_t$$

where

X_t is a vector containing the value of the **NCHANX** control variables at time = *t*,

Z_t = the state vector at time = *t*, and

G is the **NCHANX** by **NPAR** gain matrix (**GAIN**).

The gain matrix is estimated by minimizing the quadratic criterion:

$$J_I = E\left\{\sum_{t=1}^{I}\left[Z_t' Q Z_t + X_{t-1}' R X_{t-1}\right]\right\}$$

where *E* is the expectation operator, and *I* = **NSTAGE**.

Akaike and Nikagawa (1972) describe a process for obtaining the optimum gain matrix by adjusting the quadratic criterion matrices, *Q* and *R*. Initially they recommend setting *Q* and *R* to the default values described above for *Q* and *R*. They recommend that the behavior of the controller using the gain matrix obtained from these defaults be examined by simulating the controller.

From these simulations, the variances of the control input variables (*X*) should be examined to identify channels whose variances are too large or too small. If they are too large or small, their corresponding diagonal elements in *R* and *Q* should be decreased or increased.

The multivariate final prediction error for a multivariate autoregressive model with lag *p* is defined as:

$$MFPE_p = \frac{\left(1 + \frac{k \cdot m + 1}{N}\right)^m}{\left(1 - \frac{k \cdot m + 1}{N}\right)^m} \cdot \left\|\hat{\sum}_p\right\|$$

where $\left\|\hat{\Sigma}_p\right\|$ is the determinant of the estimated **NCHANY** × **NCHANY** matrix of covariances of the white noise in the multivariate series, *N* =**NOBS** and *m* =**NCHANY**.

The model selected and parameter estimates vary depending upon the value of **MAXLAG**. Akaike and Nakagawa (1972) provide a rule of thumb for **MAXLAG**: start with values between $2\sqrt{N}/m$ and $3\sqrt{N}/m$, and keep **MAXLAG** below $N/(5 \cdot m)$.

Similar to the univariate case, **MAXLAG** must never exceed $N/(2 \cdot m)$.

The numerical accuracy decreases as **MAXLAG** increases. In this case, it is possible for the estimated MFPE to become negative. If this happens try using double precision.

NSTAGE, the number of stages used to compute the gain matrix, should be selected to provide an accurate estimate of the gain matrix. Essentially, the gain matrix is the matrix for control of the system at time = **NSTAGE**. As **NSTAGE** is increased, the gain matrix converges to a constant solution. That is, as **NSTAGE** is increased,

$$G_{NSTAGE} = G_{NSTAGE+1}$$

It is tempting to set **NSTAGE** to a large value. However, this will increase execution time and subject the final estimates to rounding errors.

Example

The following example uses the Gas Furnace Data (Box and Jenkins 1976, pages 532-533). In this example, the controller has one variable, percent CO_2 , that is controlled by a single manipulated variable, input gas rate in cubic feet/minute. These multi-channel series consist of **NOBS**=296 observations.

```
          USE GDATA_INT
          USE WRRRN_INT
          USE OPT_DES_INT

          IMPLICIT NONE
          INTEGER, PARAMETER :: MAXLAG=10, NOBS=296
          INTEGER NCOL, NROW, NPAR
          REAL(KIND(1E0)) RDATA(NOBS,2), X(NOBS,1), Y(NOBS,1), GAIN(1,MAXLAG), &
          Q(1,1), R(1,1)
          EQUIVALENCE (X, RDATA(1,1)), (Y, RDATA(1,2))

          Q(1,1) = 0.16E0
          R(1,1) = 0.001E0
          GAIN=0.0E0
          CALL GDATA (7, RDATA, NROW, NCOL)
          CALL OPT_DES(MAXLAG, X, Y, NPAR, GAIN, Q=Q, R=R)
          CALL WRRRN("GAIN", GAIN(:,:NPAR))
          END
```

Output:

```

                 GAIN
          1      2       3       4
```

```
      1.418   1.851   1.894   1.502
```

LOFCF

Performs lack-of-fit test for a univariate time series or transfer function given the appropriate correlation function.

Required Arguments

NOBS — Number of observations in the stationary time series. (Input)
NOBS must be greater than or equal to two.

LAGMIN — Minimum lag of the correlation function. (Input)
LAGMIN corresponds to the lower bound of summation in the lack of fit test statistic. Generally, LAGMIN is set to one if CF is an autocorrelation function and is set to zero if CF is a cross correlation function.

LAGMAX — Maximum lag of the correlation function. (Input)
LAGMAX corresponds to the upper bound of summation in the lack of fit test statistic. LAGMAX must be greater than or equal to LAGMIN and less than NOBS.

CF — Vector of length LAGMAX + 1 containing the correlation function. (Input)
The correlation coefficient for lag *k* is given by CF(*k* + 1), *k* = LAGMIN, LAGMIN + 1, ..., LAGMAX.

NPFREE — Number of free parameters in the formulation of the time series model. (Input)
NPFREE must be greater than or equal to zero and less than LAGMAX.

Q — Lack of fit test statistic. (Output)

PVALUE — *p*-value of the test statistic Q. (Output)
Under the null hypothesis, Q has an approximate chi-squared distribution with LAGMAX - LAGMIN + 1 - NPFREE degrees of freedom.

FORTRAN 90 Interface

Generic: CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)
Specific: The specific interface names are S_LOFCF and D_LOFCF.

FORTRAN 77 Interface

Single: CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)
Double: The double precision name is DLOFCF.

Description

Routine **LOFCF** performs a portmanteau lack of fit test for a time series or transfer function containing *n* observations given the appropriate sample correlation function

$$\hat{\rho}(k)$$

for $k = L, L + 1, \ldots, K$ where L = **LAGMIN** and K = **LAGMAX**.

The basic form of the test statistic Q is

$$Q = n(n+2)\sum_{k=L}^{K}(n-k)^{-1}\hat{\rho}(k)$$

with $L = 1$ if

$$\hat{\rho}(k)$$

is an autocorrelation function and $L = 0$ if

$$\hat{\rho}(k)$$

is a cross-correlation function. Given that the model is adequate, Q has a chi-squared distribution with $K - L + 1 - m$ degrees of freedom where m = **NPFREE** is the number of parameters estimated in the model. If the mean of the time series is estimated, Woodfield (1990) recommends not including this in the count of the parameters estimated in the model. Thus, for an ARMA(p, q) model set **NPFREE** = $p + q$ regardless of whether the mean is estimated or not. The original derivation for time series models is due to Box and Pierce (1970) with the above modified version discussed by Ljung and Box (1978). The extension of the test to transfer function models is discussed by Box and Jenkins (1976, pages 394–395).

Comments

Routine **LOFCF** may be used to diagnose lack of fit in both ARMA and transfer function models. Typical arguments for these situations are

Model	LAGMIN	LAGMAX	NPFREE
ARMA(p, q)	1	$\sqrt{NOBS}$	$p + q$
Transfer function	0	$\sqrt{NOBS}$	$r + s$

See the "Description" section for further information.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. An ARMA(2,1) with nonzero mean is fitted using routine NSLSE. The autocorrelations of the residuals are estimated using routine ACF. A portmanteau lack of fit test is computed using 10 lags with LOFCF.

The warning message from NSLSE in the output can be ignored. (See the Example for routine NSLSE for a full explanation of the warning message.)

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    IARDEG, IMEAN, IPRINT, ISEOPT, LAGMAX, LAGMIN, LDCOV,&
                 LDX, MAXBC, MDX, NOBS, NP, NPAR, NPFREE, NPMA
      PARAMETER  (IARDEG=2, IMEAN=1, IPRINT=0, ISEOPT=0, LAGMAX=10, &
                 LAGMIN=1, LDX=176, MAXBC=10, MDX=2, NOBS=100, NPAR=2, &
                 NPFREE=4, NPMA=1, NP=NPAR+NPMA+IMEAN, LDCOV=NP)
!
      INTEGER    LAGAR(NPAR), LAGMA(NPMA), MAXIT, NA, NCOL, NOUT, NROW
      REAL       A(NOBS-IARDEG+MAXBC), ACV(LAGMAX+1), AVAR, &
                 CF(LAGMAX+1), CNST, COV(LDCOV,NP), PAR(NPAR), &
                 PMA(NPMA), PVALUE, Q, RELERR, SEAC(LAGMAX), TOLBC, &
                 TOLSS, W(NOBS), WMEAN, X(LDX,MDX)
!
      EQUIVALENCE (W(1), X(22,2))
!
      DATA LAGAR/1, 2/, LAGMA/1/
!
      CALL UMACH (2, NOUT)
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, X, NROW, NCOL)
!                                USE Default Convergence parameters
!                                Compute preliminary parameter
!                                estimates for ARMA(2,1) model
      CALL NSPE (W, CNST, PAR, PMA, AVAR, WMEAN=WMEAN)
!                                Compute least squares estimates
!                                for model
      TOLSS = 0.125
!
      CALL NSLSE (W, PAR, LAGAR, PMA, LAGMA, MAXBC, CNST, COV, &
                 AVAR, IMEAN=IMEAN, WMEAN=WMEAN, TOLSS=TOLSS, A=A)
!                                Compute autocorrelations of the
!                                residuals
      CALL ACF (A, LAGMAX, CF)
!
      CALL LOFCF (NOBS, LAGMIN, LAGMAX, CF, NPFREE, Q, PVALUE)
!
      WRITE (NOUT,99998) Q
      WRITE (NOUT,99999) LAGMAX - LAGMIN + 1 - NPFREE, PVALUE
!
99998 FORMAT (/4X, 'Lack of Fit statistic (Q) = ', F12.3)
```

```
99999 FORMAT (/4X, 'Degrees of freedom (LAGMAX-LAGMIN+1-NPFREE) = ', &
            I8/4X, 'P-value (PVALUE) = ', F12.4)
      END
```

Output

```
***WARNING  ERROR 1 from NSLSE.  Least squares estimation of the parameters
***         has failed to converge.  Increase MAXBC and/or TOLBC and/or
***         TOLSS.  The estimates of the parameters at the last iteration
***         may be used as new starting values.

Lack of Fit statistic (Q) =       14.572

Degrees of freedom (LAGMAX-LAGMIN+1-NPFREE) =        6
P-value (PVALUE) =       0.0239
```

DIRIC

This function computes the Dirichlet kernel.

Function Return Value

DIRIC — Function value. (Output)

Required Arguments

M — Spectral window parameter. (Input)

RANGLE — Angle in radians. (Input)

EPS — Positive bound on |RANGLE| that determines when an approximation to the Dirichlet kernel is appropriate. (Input)
EPS must be between 0 and π inclusive. The approximation is used when |RANGLE| is less than EPS.

FORTRAN 90 Interface

Generic: DIRIC (M, RANGLE, EPS)
Specific: The specific interface names are S_DIRIC and D_DIRIC.

FORTRAN 77 Interface

Single: DIRIC (M, RANGLE, EPS)
Double: The double precision name is DDIRIC.

Description

Routine DIRIC evaluates the Dirichlet kernel, $D_M(\theta)$, for a given parameter M, angle θ = RANGLE, and tolerance ε = EPS. The computational form of the function is given by

$$D_M(\theta) = \begin{cases} \frac{(2M+1)}{2\pi}\left(\frac{\sin\left[\left(M+\frac{1}{2}\right)\theta\right]}{\left(M+\frac{1}{2}\right)\theta}\right) & |\theta| < \varepsilon \\ \frac{1}{2\pi}\left(\frac{\sin\left[\left(M+\frac{1}{2}\right)\theta\right]}{\sin(\theta/2)}\right) & \varepsilon \le |\theta| \le \pi \\ 0 & |\theta| > \pi \end{cases}$$

The first case is an approximation to $D_M(\theta)$ for small θ, and the second case is the usual theoretical definition.

In spectral analysis, the Dirichlet kernel corresponds to the truncated periodogram spectral window, and *M* is called the spectral window parameter. Since the Dirichlet kernel may be negative for certain values of θ, the truncated periodogram estimate of the spectral density may also be negative. This is an undesirable property since the true spectral density is a nonnegative function. See Priestley (1981, pages 437–438) and Anderson (1971, pages 508–511) for further discussion.

Comments

1. The Dirichlet kernel is equivalent to the truncated periodogram spectral window. The spectral window parameter denotes the truncation point in the weighted sum of sample autocovariances used to estimate the spectral density.
2. The Dirichlet kernel produces negative values for certain values of **RANGLE**. Thus, spectral windows that use the Dirichlet kernel may also take on negative values.
3. The Dirichlet kernel is defined between $-\pi$ and π, inclusive, and is zero otherwise.

Example

In this example, **DIRIC** is used to compute the Dirichlet kernel at $\theta = \pm k\pi/(2M + 1)$ for $k = 0, 1, \ldots, (2M + 1)$ where $M = 5$ and $\varepsilon = 0.01$.

```
      USE UMACH_INT
      USE DIRIC_INT

      IMPLICIT   NONE

      INTEGER    K, M, NOUT
      REAL       EPS, PI, REAL, THETA, WT
      INTRINSIC  REAL
!
      M   = 5
      EPS = .01
      PI  = 3.14159
!
```

```
      CALL UMACH (2, NOUT)
!
      WRITE (NOUT,99998)
99998 FORMAT ('  K     THETA     WEIGHT ')
      DO 10  K=0, 2*M + 1
         THETA = (PI*REAL(K))/REAL(2*M+1)
         WT    = DIRIC(M,THETA,EPS)
         WRITE (NOUT,99999) K, THETA, WT
99999    FORMAT (1X, I2, 2(3X,F8.5))
   10 CONTINUE
!
      END
```

Output

```
 K     THETA     WEIGHT
 0    0.00000    1.75070
 1    0.28560    1.11833
 2    0.57120    0.00000
 3    0.85680   -0.38312
 4    1.14240    0.00000
 5    1.42800    0.24304
 6    1.71359    0.00000
 7    1.99919   -0.18919
 8    2.28479    0.00000
 9    2.57039    0.16587
10    2.85599    0.00000
11    3.14159   -0.15915
```

FEJER

This function computes the Fejér kernel.

Function Return Value

FEJER — Function value. (Output)

Required Arguments

M — Spectral window parameter. (Input)

RANGLE — Angle in radians. (Input)

EPS — Positive bound on |RANGLE| that determines when an approximation to the Fejér kernel is appropriate. (Input)
EPS must be between 0 and π inclusive. The approximation is used when |RANGLE| is less than EPS.

FORTRAN 90 Interface

Generic: FEJER(M, RANGLE, EPS)
Specific: The specific interface names are S_FEJER and D_FEJER.

FORTRAN 77 Interface

Single: FEJER(M, RANGLE, EPS)
Double: The double precision name is DFEJER.

Description

Routine FEJER evaluates the Fejér kernel, $F_M(\theta)$, for a given parameter M, angle θ = RANGLE, and tolerance ε = EPS. The computational form of the function is given by

$$F_M(\theta) = \begin{cases} \frac{M}{2\pi}\left(\frac{\sin[M\theta/2]}{M\theta/2}\right)^2 & |\theta| < \varepsilon \\ \frac{1}{2\pi M}\left(\frac{\sin[M\theta/2]}{\sin(\theta/2)}\right)^2 & \varepsilon \le |\theta| \le \pi \\ 0 & |\theta| > \pi \end{cases}$$

The first case is an approximation to $F_M(\theta)$ for small θ, and the second case is the usual theoretical definition.

In spectral analysis, the Fejér kernel corresponds to the modified Bartlett spectral window, and *M* is called the spectral window parameter. Since the Fejér kernel is nonnegative for all values of θ, the modified Bartlett estimate of the spectral density is also nonnegative. This is a desirable property since the true spectral density is a nonnegative function. See Priestley (1981, pages 439–440) and Anderson (1971, pages 508–511) for further discussion.

Comments

1. The Fejér kernel is equivalent to the modified Bartlett spectral window. The spectral window parameter denotes the truncation point in the weighted sum of sample autocovariances used to estimate the spectral density.
2. The Fejér kernel produces nonnegative values for all values of **RANGLE**. Thus, spectral windows based on the Fejér kernel are always nonnegative.
3. The Fejér kernel is defined between $-\pi$ and π, inclusive, and is zero otherwise.

Example

In this example, **FEJER** is used to compute the Fejér kernel at $\theta = \pm k\pi/M$ for $k = 0, 1, \ldots, M$ where $M = 11$ and $\varepsilon = 0.01$.

```
      USE UMACH_INT
      USE FEJER_INT

      IMPLICIT   NONE

      INTEGER    K, M, NOUT
      REAL       EPS, PI, REAL, THETA, WT
      INTRINSIC  REAL
!
      M   = 11
      EPS = .01
      PI = 3.14159265
      CALL UMACH (2, NOUT)
!
      WRITE (NOUT,99998)
99998 FORMAT ('  K     THETA      WEIGHT ')
```

```
      DO 10  K=0, M
         THETA = (PI*REAL(K))/REAL(M)
         WT    = FEJER(M,THETA,EPS)
         WRITE (NOUT,99999) K, THETA, WT
99999    FORMAT (1X, I2, 2(3X,F8.5))
   10 CONTINUE
!
      END
```

Output

```
 K     THETA     WEIGHT
 0    0.00000    1.75070
 1    0.28560    0.71438
 2    0.57120    0.00000
 3    0.85680    0.08384
 4    1.14240    0.00000
 5    1.42800    0.03374
 6    1.71360    0.00000
 7    1.99920    0.02044
 8    2.28479    0.00000
 9    2.57039    0.01572
10    2.85599    0.00000
11    3.14159    0.00000
```

ARMA_SPEC

Calculates the rational power spectrum for an ARMA model. It also computes the rational power spectrum for AR and MA models by setting the number of MA or AR coefficients to zero, respectively.

Required Arguments

PAR — Vector of length NPAR. If NPAR > 0 then PAR contains coefficients for the autoregressive terms in the model. If NPAR = 0, then the contents of PAR are ignored. (Input)

PMA — Vector of length NPMA. If NPMA > 0 then PMA contains the coefficients for the moving-average terms in the ARMA model. If NPMA = 0 then the contents of PMA are ignored. (Input)

NF — Number of frequencies at which to evaluate the spectral density estimate. NF must be greater than or equal to one. (Input)

AVAR — Estimate of the random shock variance. AVAR must be greater than zero. (Input)

S — Vector of length NF+1 containing the estimated power spectrum. (Output)

Optional Arguments

NPAR — Number of autoregressive (AR) parameters. NPAR must be greater than or equal to zero. (Input)
Default: NPAR=size(PAR).

NPMA — Number of moving-average (MA) parameters. Must be greater than or equal to zero. (Input)
Default: NPMA= size(PMA).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing
1	Prints final results only
2	Prints intermediate and final results

FORTRAN 90 Interface

Generic: CALL ARMA_SPEC (PAR, PMA, NF, AVAR, S [, …])
Specific: The specific interface names are S_ARMA_SPEC and D_ARMA_SPEC.

Description

The routine **ARMA_SPEC** is derived from the rational power spectrum analysis described by Akaike and Nakagawa (1972) and the RASPEC routine published in the original TIMSAC Library.

Using the notation developed in the introduction to this chapter, the stationary time series W_t with mean **μ** can be represented by the nonseasonal autoregressive moving average model (ARMA) by the following relationship:

$$\phi(B)(W_t - \mu) = \theta(B)A_t$$

where

$$t \in ZZ = \{ \cdots, -2, -1, 0, 1, 2, \cdots \}$$

B is the backward shift operator defined by $B^k W_t = W_{t-k}$,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_{NPAR} B^{NPAR} \quad NPAR \geq 0$$

and

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_{NPMA} B^{NPMA} \quad NPMA \geq 0$$

Routine **ARMA_SPEC** uses estimates for the coefficients $\phi_1, \phi_2, \ldots, \phi_{NPAR}$, and $\Theta_1, \Theta_2, \ldots, \Theta_{NPMA}$ as input to its algorithm, **PAR** and **PMA** respectively. These estimates can be derived from **MAX_ARMA** or by using **NSLSE**.

Routine **ARMA_SPEC** also requires an initial estimate for the variance of the white noise in the series. In **MAX_ARMA** this is returned as **AVAR**. This is also returned from the autocovariance procedure ACF as ACV(0).

Example

Consider the Wolfer Sunspot Data (Box and Jenkins 1976, page 530) consisting of the number of sunspots observed each year from 1770 through 1869. These data can be modeled using the following model [ARMA(**NPAR**=2, **NPMA**=1)]

$$(1 - \phi_1 B - \phi_2 B^2)(W_t - \mu) = (1 - \theta_1 B)A_t$$

In this example, estimates of the coeffiecients in this model are obtained using **MAX_ARMA**. These are then sent to **ARMA_SPEC** to obtain the estimated power spectrum.

```
        USE GDATA_INT
        USE ARMA_SPEC_INT
        USE MAX_ARMA_INT
        USE WRRRN_INT
```

```
      IMPLICIT NONE
      INTEGER, PARAMETER :: NF=20, LDX=176, NDX=2
      INTEGER NCOL, NROW
      REAL(KIND(1E0))  PAR(2), PMA(1), RDATA(LDX,NDX), &
           W(100), S(0:NF), AVAR
      EQUIVALENCE (W(1), RDATA(22,2))
      DATA PAR/-0.5783E0, 0.18594E0/
      DATA PMA/-0.1E0/

!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
      CALL MAX_ARMA(W, PAR, PMA, AVAR=AVAR)
      CALL ARMA_SPEC(PAR, PMA, NF, AVAR, S)
      CALL WRRRN("S", S)
      END
```

Output

```
      S
 1    714.8
 2    786.7
 3   1030.2
 4   1450.4
 5   1619.6
 6   1146.4
 7    670.4
 8    407.2
 9    269.3
10    192.3
11    146.2
12    116.7
13     96.9
14     83.2
15     73.4
16     66.3
17     61.3
18     57.7
19     55.3
20     53.9
21     53.5
```

PFFT

more...

Computes the periodogram of a stationary time series using a fast Fourier transform.

Required Arguments

X — Vector of length NOBS containing the stationary time series. (Input)

PM — ($\lfloor N/2 \rfloor + 1$) by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of *PM* is defined as:

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi_k/N$ for IFSCAL = 0 and $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, and the cosine and sine transformations of the centered and padded time series.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero. The length of the centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time
1	Frequency in cycles per unit time

IPVER — Option for version of the periodogram. (Input)
Default: IPVER = 0.

IPVER	Action
0	Compute usual periodogram.
1	Compute modified periodogram.

Refer to the “Description” section for further details.

LDPM — Leading dimension of PM exactly as specified in the dimension statement of the calling program. (Input)
LDPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDPM = size (PM,1).

FORTRAN 90 Interface

Generic: CALL PFFT (X, PM [, ...])
Specific: The specific interface names are S_PFFT and D_PFFT.

FORTRAN 77 Interface

Single: CALL PFFT (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, IPVER, PM, LDPM)
Double: The double precision name is DPFFT.

Description

Routine PFFT computes the periodogram of a stationary time series given a sample of n = NOBS observations $\{X_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\left\{\widetilde{X}_t\right\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where N = NOBS + NPAD,

$$\widetilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \overline{X} = \frac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The discrete Fourier transform of

$$\left\{\widetilde{X}_t\right\}$$

for $t = 1, \ldots, N$ is defined by

$$\zeta_{\widetilde{X}}(\omega_k) = \sum_{t=1}^{N} \widetilde{X}_t e^{-i\omega_k t}$$

over the discrete set of frequencies

$$\omega_k = \frac{2\pi k}{N} \quad k = 0, \pm 1, \ldots, \pm \lfloor N/2 \rfloor$$

An alternative representation of

$$\zeta_{\widetilde{X}}\left(\omega_k\right)$$

in terms of cosine and sine transforms is

$$\zeta_{\widetilde{X}}(\omega_k) = \alpha_{\widetilde{X}}(\omega_k) - i\beta_{\widetilde{X}}(\omega_k)$$

where

$$\alpha_{\widetilde{X}}(w_k) = \sum_{t=1}^{n} \widetilde{X}_t \cos(\omega_k t)$$

and

$$\beta_{\widetilde{X}}(\omega_k) = \sum_{t=1}^{n} \widetilde{X}_t \sin(\omega_k t)$$

The fast Fourier transform algorithm is used to compute the discrete Fourier transform. The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the centered and padded sequence

$$\left\{\widetilde{X}_t\right\}$$

$t = 1, \ldots, N$ is defined by

$$I_{n,N,\widetilde{X}}(\omega_k) = K\left|\sum_{t=1}^{N} \widetilde{X}_t e^{-i\omega_k t}\right|^2 = K\left|\zeta_{\widetilde{X}}(\omega_k)\right|^2$$

where K is the scale factor

$$K = \begin{cases} \frac{2}{n} & \text{for the usual periodogram} \\ \frac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The scale factor of the usual periodogram relates the ordinates to the sum of squares of

$$X_t - \hat{\mu}_X$$

(Fuller 1976, pages 276–277). If the first ordinate (corresponding to $k = 0$) is replaced by one-half of its value, then if N is odd, the sum of the $\lfloor N/2 \rfloor + 1$ ordinates corresponding to $k = 0, 1, \ldots, \lfloor N/2 \rfloor$ is

$$\frac{N}{n}\sum_{t=1}^{n}\left(X_t - \hat{\mu}_X\right)^2$$

For *N* even, if the first ordinate (corresponding to *k* = 0) and the last ordinate (corresponding to *k* = *N*/2) are each replaced by one-half of their values, then the same relationship holds. The modified periodogram is an asymptotically unbiased estimate of the nonnormalized spectral density function at each frequency ω_k (Priestley 1981, page 417). The argument **IPVER** is used to specify the version of the periodogram.

The alternative representation of the discrete Fourier transform implies

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\alpha_{\tilde{X}}(\omega_k)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\beta_{\tilde{X}}(\omega_k)$$

represent the (scaled) cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \quad k = 0,\ 1,\ \ldots,\ \lfloor N/2 \rfloor$$

Use of the centered data

$$\left\{\tilde{X}_t\right\}$$

(without padding) instead of the original data $\{X_t\}$ for *t* = 1, ..., *n* does not affect the asymptotic sampling properties of the periodogram. In fact,

$$I_{n,n,\tilde{X}}(\omega_k) \equiv I_{n,n,X}(\omega_k) \quad \omega_k \neq 0$$

For $\omega_k = 0$, both

$$I_{n,n,\tilde{X}}(0) = 0$$

and

$$I_{n,n,X}(0) = K\left(\sum_{t=1}^{n} X_t\right)^2 = Kn^2\overline{X}^2$$

reflect the mean of the data. See Priestley (1981, page 417) for further discussion.

Comments

1. Workspace may be explicitly provided, if desired, by use of P2FT/DP2FT. The reference is:

   ```
   CALL P2FT (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, IPVER, PM, LDPM, CX, COEF,
      WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length *N*.

   COEF — Complex work vector of length *N*.

   WFFTC — Work vector of length 4*N* + 15.

   CPY — Work vector of length 2*N*.

2. The centered and padded time series is defined by

   CX(*j*) = X(*j*) - XCNTR for *j* = 1, ..., NOBS

   CX(*j*) = 0 for *j* = NOBS + 1, ..., *N*

   where *N* = NOBS + NPAD.

3. The periodogram *I*(ω) is an even function of the frequency ω. The relation *I*(-ω) = *I*(ω) for ω > 0.0 recovers the periodogram for negative frequencies.

4. Since cos(ω) is an even function of ω and sin(ω) is an odd function of ω, the cosine and sine transformations, respectively, satisfy *A*(-ω) = *A*(ω) and *B*(-ω) = -*B*(ω) for ω > 0.0. Similarly, the complex Fourier coefficients, stored in COEF, satisfy COEF(-ω) = conj(COEF(ω)).

5. Computation of the 2 * NOBS - 1 autocovariances of X using the inverse Fourier transform of the periodogram requires NPAD = NOBS - 1.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine PFFT to these data produces the following results.

```
USE GDATA_INT
USE PFFT_INT
USE WRRRL_INT
```

```
      IMPLICIT   NONE
      INTEGER    LDPM, NOBS
      PARAMETER  (LDPM=100, NOBS=100)
!
      INTEGER    IPVER, NCOL, NPAD, NROW
      REAL       PM(LDPM,5), RDATA(176,2), REAL, X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6
      INTRINSIC  REAL
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
           'I(w(k))', 'A(w(k))', 'B(w(k))'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
!                                Pad standard amount
!                                Frequency in radians per unit time
!                                Modified periodogram version
      IPVER = 1
!                                Compute the periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!                                Print results
      FMT = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (' ', PM, RLABEL, CLABEL, 20, 5, FMT=FMT)
!
      END
```

Output

```
 Frequency  Period     I(w(k))     A(w(k))     B(w(k))
    0.0000     NaN        0.00        0.00        0.00
    0.0316  199.00      183.97        3.72      -13.04
    0.0631   99.50     1363.37       35.45      -10.32
    0.0947   66.33     2427.09       29.31       39.60
    0.1263   49.75     1346.64      -21.74       29.56
    0.1579   39.80      139.74      -11.69       -1.79
    0.1894   33.17       44.67       -4.65        4.80
    0.2210   28.43      123.47      -11.11       -0.33
    0.2526   24.88      176.04       -4.79      -12.37
    0.2842   22.11      143.06        9.92       -6.69
    0.3157   19.90       44.17        6.43        1.68
    0.3473   18.09       38.95        5.40        3.13
    0.3789   16.58       63.20        7.14        3.49
    0.4105   15.31      537.64        0.89       23.17
    0.4420   14.21      944.68      -30.73       -0.75
    0.4736   13.27      162.02       -0.95      -12.69
    0.5052   12.44      908.09      -24.51      -17.53
    0.5368   11.71     3197.84       34.84      -44.54
    0.5683   11.06     1253.82       19.69       29.43
    0.5999   10.47      850.45       -8.75      -27.82
```

SSWD

more...

Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the time series data.

Required Arguments

X — Vector of length **NOBS** containing the stationary time series. (Input)

F — Vector of length **NF** containing the frequencies at which to evaluate the spectral density estimate. (Input)
The units of **F** correspond to the scale specified by **IFSCAL**. The elements of **F** must be in the range (**-**π/**TINT**,π/**TINT**), inclusive for **IFSCAL** = 0 and (**-**1/(2 ***** **TINT**), 1/(2 ***** **TINT**)) inclusive for **IFSCAL** = 1.

M — Vector of length **NM** containing the values of the spectral window parameters **M**. (Input)
For the Parzen spectral window (**ISWVER** = 5), all values of the spectral window parameters **M** must be even.

PM — (⌊*N*/2⌋ + 1) by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For *k* = 0, 1, ..., ⌊*N*/2⌋, the (*k* + 1)-st element of the *j*-th column of **PM** is defined as

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for **IFSCAL** = 0 or $\omega_k = k/N$ for **IFSCAL** = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for **IFSCAL** = 0 and $p_k = 1/\omega_k$ for **IFSCAL** = 1. If $\omega_k = 0$, p_k is set to missing.
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

Note *N* = **NOBS** + **NPAD**.

SM — NF by NM + 2 matrix containing a summarization of the spectral analysis. (Output)
The *k*-th element of the *j*-th column of SM is defined as

Col	Description
1	Frequency, F(*k*).
2	Period, p_k where $p_k = 2\pi/F(k)$ for IFSCAL = 0 and $p_k = 1/F(k)$ for IFSCAL = 1. If F(*k*) = 0, p_k is set to missing.
3	Spectral density estimate at F(*k*) using the spectral window parameter M(1).
4	Spectral density estimate at F(*k*) using the spectral window parameter M(2).
⋮	
⋮	
NM +2	Spectral density estimate at F(*k*) using the spectral window parameter M(NM)

where *k* = 1, ..., NF.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine transform and sine transform of the centered and padded time series, and the spectral density estimate based on a specified version of a spectral window for a given set of spectral window parameters.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the spectral density estimate. (Input)
Default: NF = size (F,1).

TINT — Time interval at which the series is sampled. (Input)
For a discrete parameter process, usually TINT = 1. For a continuous parameter process, TINT > 0.
TINT is used to adjust the spectral density estimate.
Default: TINT = 1.0.

ISWVER — Option for version of the spectral window. (Input)
Default: ISWVER = 1

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Algorithm" section for further details.

NM — Number of spectral window parameters M used to compute the spectral density estimate for a given spectral window version. (Input)
NM must be greater than or equal to one.
Default: NM = size (M,1).

LDPM — Leading dimension of PM exactly as specified in the dimension statement of the calling program. (Input)
LDPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDPM = size (PM,1).

LDSM — Leading dimension of SM exactly as specified in the dimension statement of the calling program. (Input)
LDSM must be greater than or equal to NF.
Default: LDSM = size (SM,1).

FORTRAN 90 Interface

Generic: CALL SSWD (X, F, M, PM, SM [, ...])

Specific: The specific interface names are S_SSWD and D_SSWD.

FORTRAN 77 Interface

Single: CALL SSWD (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT, ISWVER, NM, M, PM, LDPM, SM, LDSM)

Double: The double precision name is DSSWD.

Description

Routine SSWD estimates the nonnormalized spectral density function of a stationary time series using a spectral window given a sample of n = NOBS observations $\{X_t\}$ for t = 1, 2, ..., n.

Let

$$\left\{\tilde{X}_t\right\}$$

for t = 1, ..., N represent the centered and padded data where N = NOBS + NPAD,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

The modified periodogram of

$$\left\{\tilde{X}_t\right\}$$

for t = 1, ..., N is estimated by

$$I_{n,N,\tilde{X}}(\omega_k) = A^2_{\tilde{X}}(\omega_k) + B^2_{\tilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \quad k = 0,\ 1,\ \ldots,\ \lfloor N/2 \rfloor$$

The routine PFFT is used to compute the modified periodogram of

$$\tilde{X}_t$$

The estimate of the nonnormalized spectral density $h_X(\omega)$ is computed according to

$$\hat{h}_X(\omega) = \frac{2\pi}{N}\sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\tilde{X}}(\omega_k)W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument ISWVER. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$ and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2\left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

The argument **NM** specifies the number of window parameters M and corresponds to the number of spectral density estimates to be computed for a given spectral window. The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i \qquad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale specified by the argument **IFSCAL** but are transformed to the scale of radians per unit time for computational purposes.

The above formula for

$$\hat{h}_X(\omega)$$

assumes the data $\{X_t\}$ correspond to a realization of a discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = **TINT** equivalent to one. However, if the data correspond to a realization of a continuous-parameter stationary process recorded at equal time intervals, then the estimate of the nonnormalized spectral density must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t \hat{h}_X(\omega) \qquad |\omega| \leq \pi / \Delta t$$

Note that the frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt. Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2WD/DS2WD**. The reference is:

   ```
   CALL S2WD (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT, ISWVER, NM, M, PM,
      LDPM, SM, LDSM, CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex vector of length *N* containing the centered and padded time series **X**. (Output)

   COEF — Complex vector of length *N* containing the Fourier coefficients of the finite Fourier transform of **CX**. (Output)
   Note that **COEF**(*k*) is the appropriately scaled Fourier coefficient at frequency ω_k, $k = 0, 1, \ldots, N - 1$.

   WFFTC — Vector of length $4N + 15$.

   CPY — Vector of length $2N$.

2. The normalized spectral density estimate is obtained by dividing the nonnormalized spectral density estimate in matrix **SM** by an estimate of the variance of **X**.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine **SSWD** to these data produces the following results:

```
      USE GDATA_INT
      USE SSWD_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    LDPM, LDSM, NF, NM, NOBS
```

```
      REAL       PI
      PARAMETER  (NF=20, NM=3, NOBS=100, PI=3.141592654, &
                 LDPM=NOBS, LDSM=NF)
!
      INTEGER    I, ISWVER, M(NM), NCOL, NROW
      REAL       F(NF), PM(LDPM,5), RDATA(176,2), FLOAT, SM(LDSM,5), &
                  X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6, TITLE*60
      INTRINSIC  REAL
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
          'M = 10', 'M = 20', 'M = 30'/
!                                 Wolfer Sunspot Data for
!                                 years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                 Center on arithmetic mean
!                                 Pad standard amount (Default)
!     USE Default Frequency in radians per unit time
!                                 Determine frequencies at which
!                                 to evaluate spectral density
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!     USE Default Time interval for discrete data
!                                 Spectral window parameters
      M(1) = 10
      M(2) = 20
      M(3) = 30
!                                 Compute spectral density using
!                                 the Parzen window
      ISWVER = 5
      CALL SSWD (X, F, M, PM, SM,  ISWVER=ISWVER)
!                                 Print results
      TITLE = 'Spectral Density Using the Parzen Window'
      FMT   = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!                                 Compute spectral density using
!                                 the Bartlett-Priestley window
      ISWVER = 6
      CALL SSWD (X, F, M, PM, SM, ISWVER=ISWVER)
!                                 Print results
      TITLE = '%/Spectral Density Using the Bartlett-Priestley '// &
            'Window'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
      Spectral Density Using the Parzen Window
 Frequency  Period      M = 10      M = 20      M = 30
    0.1571   40.00      659.64      617.42      619.73
    0.3142   20.00      666.95      554.70      339.61
    0.4712   13.33      653.73      770.64      860.49
    0.6283   10.00      598.77      857.80     1046.13
    0.7854    8.00      497.47      582.85      550.77
```

0.9425	6.67	367.72	266.33	186.98
1.0996	5.71	240.65	121.46	104.79
1.2566	5.00	142.41	76.17	76.74
1.4137	4.44	81.28	54.20	47.19
1.5708	4.00	49.13	40.16	41.39
1.7279	3.64	32.57	27.58	26.46
1.8850	3.33	22.44	16.52	14.40
2.0420	3.08	15.53	10.93	9.87
2.1991	2.86	11.19	8.30	8.32
2.3562	2.67	8.66	6.18	5.86
2.5133	2.50	6.93	4.75	4.22
2.6704	2.35	5.51	4.62	4.35
2.8274	2.22	4.47	4.91	5.24
2.9845	2.11	3.61	4.23	4.75
3.1416	2.00	2.62	2.44	2.27

Spectral Density Using the Bartlett-Priestley Window

Frequency	Period	M = 10	M = 20	M = 30
0.1571	40.00	604.34	712.73	757.61
0.3142	20.00	564.28	176.81	107.08
0.4712	13.33	767.63	927.14	981.10
0.6283	10.00	900.32	1190.30	1172.23
0.7854	8.00	607.45	494.85	571.65
0.9425	6.67	237.16	127.65	87.36
1.0996	5.71	103.34	113.93	135.34
1.2566	5.00	75.74	74.88	57.57
1.4137	4.44	52.64	44.98	38.59
1.5708	4.00	38.50	44.56	50.59
1.7279	3.64	27.35	25.28	21.76
1.8850	3.33	15.68	13.84	13.10
2.0420	3.08	10.33	9.79	7.41
2.1991	2.86	7.95	8.31	8.67
2.3562	2.67	6.04	5.86	7.08
2.5133	2.50	4.56	3.67	2.90
2.6704	2.35	4.44	4.38	4.06
2.8274	2.22	4.99	5.62	5.40
2.9845	2.11	4.31	5.07	5.08
3.1416	2.00	2.43	2.23	2.44

SSWP

Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the periodogram.

Required Arguments

N — Number of observations in the centered and padded time series **X**. (Input)
N must be greater than or equal to two.

PX — Vector of length ⌊*N*/2⌋ + 1 containing the (modified) periodogram of **X**. (Input)
The periodogram ordinate evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by **PX**(*k* + 1), *k* = 0, 1, …, ⌊*N*/2⌋.

F — Vector of length **NF** containing the (angular) frequencies at which the spectral density is estimated. (Input)

M — Spectral window parameter. (Input)
M must be greater than or equal to one and less than **N**.

SX — Vector of length **NF** containing the estimate of the spectral density of the time series **X**. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one.
Default: **NF** = size (**F**,1).

ISWVER — Option for version of the spectral window. (Input)
Default: **ISWVER** = 1.

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the "Algorithm" section for further details.

FORTRAN 90 Interface

Generic: CALL SSWP (N, PX, F, M, SX [, ...])
Specific: The specific interface names are S_SSWP and D_SSWP.

FORTRAN 77 Interface

Single: CALL SSWP (N, PX, NF, F, ISWVER, M, SX)
Double: The double precision name is DSSWP.

Description

Routine SSWP estimates the nonnormalized spectral density function of a stationary time series using a spectral window given the modified periodogram of the appropriately centered and padded data

$$\{\widetilde{X}_t\} \qquad \text{for } t = 1, \ \ldots, N$$

The routine PFFT may be used to obtain the modified periodogram

$$I_{N,\,\widetilde{X}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0,1, \ \ldots, \lfloor N/2 \rfloor$$

The symmetry of the periodogram is used to recover the ordinates at negative frequencies.

The estimate of the nonnormalized spectral density $h_X(\omega)$ is computed according to

$$\hat{h}_X(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} I_{n,N,\widetilde{X}}(\omega_k) W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument `ISWVER`. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$ and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta)\right]^2\left\{1 - \frac{2}{3}\sin^2(\theta/2)\right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

Only one window parameter M may be specified so that only one estimate of $h_X(\omega)$ is computed. The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one.

Comments

1. The periodogram of **X** may be computed using the routine **PFFT**. Estimation of the spectral density of **X** using the modified periodogram preserves the scale of the spectral density up to adjustment for the time sampling interval.
2. The time sampling interval, **TINT**, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (**TINT** > 0.0) involves multiplication of the frequency vector **F** by 1/**TINT** and multiplication of the spectral density estimate by **TINT**.
3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply **F** by 1/(2π).

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine **SSWP** to these data produces the following results:

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDPM, LDSM, NF, NM, NOBS
      REAL       PI, NPAD
      PARAMETER  (NF=20, NM=3, NOBS=100, PI=3.141592654, &
                 LDPM=NOBS, LDSM=NF)
!
      INTEGER    I, IPVER, ISWVER, J, M(NM), N, NCOL, NROW
      REAL       F(NF), PM(LDPM,5), PX(LDPM), RDATA(176,2), FLOAT, &
                 SM(NF,5), SX(NF), X(NOBS)
      CHARACTER  CLABEL(6)*9, FMT*20, RLABEL(1)*6, TITLE*60
      INTRINSIC  FLOAT
!
      EQUIVALENCE (PX(1), PM(1,3)), (F(1), SM(1,1))
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Frequency', 'Period', &
          'M = 10', 'M = 20', 'M = 30'/
!                                Wolfer Sunspot Data for
!                                years 1770 through 1869
      CALL GDATA (2, RDATA, NROW, NCOL)
!                                Center on arithmetic mean
```

```
!                                  Pad standard amount
      NPAD = NOBS-1
!                                  Frequency in radians per unit time
!                                  Modified periodogram version
      IPVER = 1
!                                  Compute periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!                                  Number of observations used to
!                                  compute the periodogram
      N = NOBS + NPAD
!                                  Determine frequency and period
!                                  at which to evaluate the spectral
!                                  density
      DO 10  I=1, NF
         SM(I,1) = PI*FLOAT(I)/FLOAT(NF)
         SM(I,2) = 2.0*FLOAT(NF)/FLOAT(I)
   10 CONTINUE
!                                  Spectral window parameters
      M(1) = 10
      M(2) = 20
      M(3) = 30
!                                  Compute spectral density using
!                                  the Parzen window
      ISWVER = 5
      DO 20  J=1, NM
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
!                                  Copy into SM
         CALL SCOPY (NF, SX, 1, SM(1:,2+J), 1)
   20 CONTINUE
!                                  Print results
      TITLE = 'Spectral Density Using the Parzen Window'
      FMT   = '(F9.4, F6.2, 3F10.2)'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!                                  Compute spectral density using
!                                  the Bartlett-Priestley window
      ISWVER = 6
      DO 30  J=1, NM
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
!                                  Copy into SM
         CALL SCOPY (NF, SX, 1, SM(1:,2+J), 1)
   30 CONTINUE
!                                  Print results
      TITLE = '%/Spectral Density Using the Bartlett-Priestley '// &
              & 'Window'
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
      Spectral Density Using the Parzen Window
 Frequency  Period      M = 10      M = 20      M = 30
    0.1571   40.00      659.64      617.42      619.73
    0.3142   20.00      666.95      554.70      339.61
    0.4712   13.33      653.73      770.64      860.49
    0.6283   10.00      598.77      857.80     1046.13
    0.7854    8.00      497.47      582.85      550.77
    0.9425    6.67      367.72      266.33      186.98
```

1.0996	5.71	240.65	121.46	104.79
1.2566	5.00	142.41	76.17	76.74
1.4137	4.44	81.28	54.20	47.19
1.5708	4.00	49.13	40.16	41.39
1.7279	3.64	32.57	27.58	26.46
1.8850	3.33	22.44	16.52	14.40
2.0420	3.08	15.53	10.93	9.87
2.1991	2.86	11.19	8.30	8.32
2.3562	2.67	8.66	6.18	5.86
2.5133	2.50	6.93	4.75	4.22
2.6704	2.35	5.51	4.62	4.35
2.8274	2.22	4.47	4.91	5.24
2.9845	2.11	3.61	4.23	4.75
3.1416	2.00	2.62	2.44	2.27

Spectral Density Using the Bartlett-Priestley Window

Frequency	Period	M = 10	M = 20	M = 30
0.1571	40.00	604.34	712.73	757.61
0.3142	20.00	564.28	176.81	107.08
0.4712	13.33	767.63	927.14	981.10
0.6283	10.00	900.32	1190.30	1172.23
0.7854	8.00	607.45	494.85	571.65
0.9425	6.67	237.16	127.65	87.36
1.0996	5.71	103.34	113.93	135.34
1.2566	5.00	75.74	74.88	57.57
1.4137	4.44	52.64	44.98	38.59
1.5708	4.00	38.50	44.56	50.59
1.7279	3.64	27.35	25.28	21.76
1.8850	3.33	15.68	13.84	13.10
2.0420	3.08	10.33	9.79	7.41
2.1991	2.86	7.95	8.31	8.67
2.3562	2.67	6.04	5.86	7.08
2.5133	2.50	4.56	3.67	2.90
2.6704	2.35	4.44	4.38	4.06
2.8274	2.22	4.99	5.62	5.40
2.9845	2.11	4.31	5.07	5.08
3.1416	2.00	2.43	2.23	2.44

SWED

more...

Estimations of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the time series data.

Required Arguments

X — Vector of length **NOBS** containing the stationary time series. (Input)

F — Vector of length **NF** containing the frequencies at which to evaluate the spectral density estimate. (Input)
The units of **F** correspond to the scale specified by **IFSCAL**. The elements of **F** must be in the range (**-**π/**TINT**, π/**TINT**), inclusive, for **IFSCAL** = 0 and (**-**1/(2 ***** **TINT**), 1/(2 ***** **TINT**)), inclusive, for **IFSCAL** = 1.

WT — Vector of length **NWT** containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in **WT** normalized to sum to 1 with the current periodogram ordinate taking the middle weight for **NWT** odd or the weight to the right of the middle for **NWT** even.

PM — (⌊*N*/2⌋ + 1) by 5 matrix that contains a summarization of the periodogram analysis. (Output)
For *k* = 0, 1, …, ⌊*N*/2⌋, the (*k* + 1)-st element of the *j*-th column of **PM** is defined as

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for **IFSCAL** = 0 or $\omega_k = k/N$ for **IFSCAL** = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for **IFSCAL** = 0 and $p_k = 1/\omega_k$ for **IFSCAL** = 1. If $\omega_k = 0$, p_k is set to the missing value or NaN (not a number).
3	Periodogram ordinate, $I(\omega_k)$.
4	Cosine transformation coefficient, $A(\omega_k)$.
5	Sine transformation coefficient, $B(\omega_k)$.

SM — NF by 3 matrix containing a summarization of the spectral analysis. (Output)
The *k*-th element of the *j*-th column of SM is defined as

Col	Description
1	Frequency, F(*k*).
2	Period, p_k where $p_k = 2\pi/F(k)$ for IFSCAL = 0 and $p_k = 1/F(k)$ for IFSCAL = 1. If F(*k*) = 0, p_k is set to missing.
3	Spectral density estimate at F(*k*) using the specified weights WT.

where *k* = 1, …, NF.

Optional Arguments

NOBS — Number of observations in the stationary time series X. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine transforms of the centered and padded time series, and the spectral density estimate based on a specified weight sequence.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad the centered time series. (Input)
NPAD must be greater than or equal to zero. The length of the centered and padded time series is *N* = NOBS + NPAD.
Default: NPAD = NOBS –1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the spectral density estimate. (Input)
NF must be greater than zero.
Default: NF = size (F,1).

TINT — Time interval at which the series is sampled. (Input)
For a discrete parameter process, usually TINT = 1.0. For a continuous parameter process, TINT > 0.0. TINT is used to adjust the spectral density estimate.
Default: TINT = 1.0.

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT = size (WT,1).

LDPM — Leading dimension of PM exactly as specified in the dimension statement in the calling program. (Input)
LDPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDPM = size (PM,1).

LDSM — Leading dimension of SM exactly as specified in the dimension statement in the calling program. (Input)
LDSM must be greater than or equal to NF.
Default: LDSM = size (SM,1).

FORTRAN 90 Interface

Generic: CALL SWED (X, F, WT, PM, SM [, ...])
Specific: The specific interface names are S_SWED and D_SWED.

FORTRAN 77 Interface

Single: CALL SWED (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT, NWT, WT, PM, LDPM, SM, LDSM)
Double: The double precision name is DSWED.

Description

Routine **SWED** estimates the nonnormalized spectral density function of a stationary time series using a fixed sequence of weights, given a sample of n = **NOBS** observations $\{X_t\}$, for $t = 1, 2, \ldots, n$.

Let

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

represent the centered and padded data where N = **NOBS** + **NPAD**,

$$\widetilde{X}_t = \begin{cases} X_t - \hat{\mu}_X, & t = 1, \ \ldots, n \\ 0, & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = XCNTR$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X, & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t, & \mu_X \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

is estimated by

$$I_{n,N,\widetilde{X}}(\omega_k) = A^2_{\widetilde{X}}(\omega_k) + B^2_{\widetilde{X}}(\omega_k)$$

where

$$A_{\widetilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\widetilde{X}_t\cos(\omega_k t)$$

and

$$B_{\widetilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\widetilde{X}_t\sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and *K* is the scale factor equal to 1/(2π*n*). Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0,1, \ \ldots, \ \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to *a*.) The routine PFFT is used to compute the modified periodogram of

$$\left\{ \tilde{X}_t \right\}$$

Consider the sequence of *m* = **NWT** weights

$$\{w_j\} \text{ for } j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$$

where

$$\Sigma_j w_j = 1$$

These weights are fixed in the sense that they do not depend on the frequency **ω** at which to estimate the non-normalized spectral density $h_X(\omega)$. The estimate of the nonnormalized spectral density is computed according to

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\tilde{X}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega) + j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to **ω**. The weights specified by argument **WT** may be relative since they are normalized to sum to one in the actual computation of

$$\hat{h}_X(\omega)$$

Usually, *m* is odd with the weights symmetric about the middle weight w_0. If *m* is even, the weight to the right of the middle is considered w_0. Note that periodogram ordinate

$$I_{n,N,\tilde{X}}(0)$$

is replaced by

$$I_{n,N,\tilde{X}}(\omega_1)$$

and the sum reflects at each end.

The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale specified by the argument **IFSCAL** but are transformed to the scale of radians per unit time for computational purposes.

The above formula for

$$\hat{h}_X(\omega)$$

assumes the data $\{X_t\}$ correspond to a realization of a discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = **TINT** equivalent to one. However, if the data correspond to a realization of a continuous-parameter stationary process recorded at equal time intervals, then the estimate of the nonnormalized spectral density must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t \hat{h}_X(\omega), \quad |\omega| \leq \pi / \Delta t$$

Note that the frequency ω of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval Δt.

Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2ED/DS2ED**. The reference is:

   ```
   CALL S2ED (NOBS, X, IPRINT, XCNTR, NPAD, IFSCAL, NF, F, TINT, NWT, WT, PM, LDPM, SM,
      LDSM, CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex vector of length *N* containing the centered and padded time series **X**. (Output)

   COEF — Complex vector of length *N* containing the Fourier coefficients of the finite Fourier transform of **CX**. (Output)
   Note that **COEF**(*k* + 1) is the appropriately scaled Fourier coefficient at frequency ω_k, *k* = 0, 1, …, *N* - 1.

   WFFTC — Work vector of length 4*N* + 15.

   CPY — Work vector of length 2*N*.

2. The centered and padded time series is defined by

   CX(*j*) = **X**(*j*) **- XCNTR** for *j* = 1, …, **NOBS**

   CX(*j*) = 0 for *j* = **NOBS** + 1, …, *N*

   where *N* = **NOBS** + **NPAD**.

3. The normalized spectral density estimate is obtained by dividing the nonnormalized spectral density estimate in matrix **SM** by an estimate of the variance of **X**.

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine **SWED** to these data produces the following results:

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDPM, LDRDAT, LDSM, NDRDAT, NF, NOBS, NPAD, NWT
      PARAMETER  (LDRDAT=176, NDRDAT=2, NF=20, NOBS=100, NWT=7, &
                  LDSM=NF, NPAD=NOBS-1, LDPM=(NOBS+NPAD)/2+1)
!
      INTEGER    I, NROW, NVAR
      REAL       F(NF), PI, PM(LDPM,5), RDATA(LDRDAT,NDRDAT), &
                 REAL, SM(LDSM,3), WT(NWT), X(NOBS), IFSCAL, IPRINT, TINT
      CHARACTER  CLABEL(4)*20, FMT*20, RLABEL(1)*4, TITLE*28
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
      DATA IPRINT/0/, IFSCAL/0/, TINT/1.0/
      DATA FMT/'(F9.4, F6.2, F9.4)'/
      DATA RLABEL/'NONE'/
      DATA CLABEL/' ', '%/Frequency', '%/Period', 'Spectral%/Estimates' &
          /
      DATA TITLE/'Results of Spectral Analysis'/
!                                Initializations
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Wolfer Sunspot Data for years
!                                1770 through 1869
      CALL GDATA (2, RDATA, NROW, NVAR)
!                                Center on arithmetic mean
!                                Spectral density
      CALL SWED (X, F, WT, PM, SM)
!                                Print Results
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
    Results of Spectral Analysis
                        Spectral
    Frequency  Period  Estimates
       0.1571   40.00   710.8386
       0.3142   20.00   116.3940
       0.4712   13.33   937.1508
       0.6283   10.00  1209.8268
       0.7854    8.00   538.9236
       0.9425    6.67    84.9561
       1.0996    5.71   128.0791
       1.2566    5.00    55.0304
       1.4137    4.44    40.2022
       1.5708    4.00    46.4240
       1.7279    3.64    21.0053
       1.8850    3.33    12.1449
       2.0420    3.08     8.8654
       2.1991    2.86     7.2589
       2.3562    2.67     6.8078
       2.5133    2.50     3.3873
       2.6704    2.35     3.9504
       2.8274    2.22     5.7418
       2.9845    2.11     4.4652
       3.1416    2.00     4.1216
```

SWEP

Estimations of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the periodogram.

Required Arguments

N — Number of observations in the appropriately centered and padded time series **X**. (Input)
N must be greater than or equal to two.

PX — Vector of length $\lfloor N/2 \rfloor + 1$ containing the (modified) periodogram of **X**. (Input)
The periodogram ordinate evaluated at (angular) frequency $\omega_k = 2\pi k/\mathbf{N}$ is given by **PX**(k + 1), k = 0, 1, …, $\lfloor N/2 \rfloor$.

F — Vector of length **NF** containing the (angular) frequencies at which the spectral density is estimated. (Input)

WT — Vector of length **NWT** containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in **WT** normalized to sum to 1 with the current periodogram ordinate taking the middle weight for **NWT** odd or the weight to the right of the middle for **NWT** even.

SX — Vector of length **NF** containing the estimate of the spectral density of the time series **X**. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one.
Default: **NF** = size (**F**,1).

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: **NWT** = size (**WT**,1).

FORTRAN 90 Interface

Generic: CALL SWEP (N, PX, F, WT, SX [, …])
Specific: The specific interface names are S_SWEP and D_SWEP.

FORTRAN 77 Interface

Single: **CALL SWEP** (**N**, **PX**, **NF**, **F**, **NWT**, **WT**, **SX**)

Double: The double precision name is **DSWEP**.

Description

Routine **SWEP** estimates the nonnormalized spectral density function of a stationary time series using a fixed sequence of weights given the modified periodogram of the appropriately centered and padded data

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

The routine **PFFT** may be used to obtain the modified periodogram

$$I_{n,N,\widetilde{X}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$w_k = \frac{2\pi k}{N}, \ \ k = 0, \ 1, \ \ldots, \ \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) The symmetry of the periodogram is used to recover the ordinates at negative frequencies.

Consider the sequence of m = **NWT** weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ where $\Sigma_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency **ω** at which to estimate the nonnormalized spectral density $h_X(\omega)$. The estimate of the nonnormalized spectral density is computed according to

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\widetilde{X}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega) + j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to **ω**. The weights specified by argument **WT** may be relative since they are normalized to sum to one in the actual computation of

$$\hat{h}_X(\omega)$$

Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. Note that periodogram ordinate

$$I_{n,N,\tilde{X}}(0)$$

is replaced by

$$I_{n,N,\tilde{X}}(\omega_1)$$

and the sum reflects at each end.

The nonnormalized spectral density estimate is computed over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one.

Approximate confidence intervals for $h(\omega)$ can be computed using formulas given in the introduction.

Comments

1. The periodogram of **X** may be computed using the routine **PFFT**. Estimation of the spectral density of **X** using the modified periodogram preserves the scale of the spectral density up to adjustment for the time sampling interval.

2 The time sampling interval, **TINT**, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (**TINT** > 0) involves multiplication of the frequency vector **F** by 1/**TINT** and multiplication of the spectral density estimate by **TINT**.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply **F** by 1/(2π).

Example

Consider the Wölfer Sunspot Data (Anderson 1971, page 660) consisting of the number of sunspots observed each year from 1749 through 1924. The data set for this example consists of the number of sunspots observed from 1770 through 1869. Application of routine **SWEP** to these data produces the following results:

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    LDPM, LDRDAT, N, NDRDAT, NF, NOBS, NPAD, NWT
          PARAMETER  (LDRDAT=176, NDRDAT=2, NF=20, NOBS=100, NWT=7, &
                     NPAD=NOBS-1, LDPM=(NOBS+NPAD)/2+1, N=NOBS+NPAD)
    !
          INTEGER    I, IFSCAL, IPVER, NROW, NVAR
```

```
      REAL       F(NF), PI, PM(LDPM,5), RDATA(LDRDAT,NDRDAT), &
                 FLOAT, SM(NF,2), SX(NF), WT(NWT), X(NOBS)
      CHARACTER  CLABEL(3)*30, FMT*20, RLABEL(1)*4, TITLE*28
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(22,2))
!
      DATA WT/1., 2., 3., 4., 3., 2., 1./
      DATA IPVER/1/, IFSCAL/0/
      DATA FMT/'(F9.4)'/
      DATA CLABEL/'     ', '%/Frequency', 'Spectral%/Estimates'/
      DATA RLABEL/'NONE'/
      DATA TITLE/'Results of Spectral Analysis'/
!                                Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Wolfer Sunspot Data for years
!                                1770 through 1869
      CALL GDATA (2, RDATA, NROW, NVAR)
!                                Compute modified periodogram
      CALL PFFT (X, PM, IPVER=IPVER)
!
!                                Compute spectral density
      CALL SWEP (N, PM(:,3), F, WT, SX)
!
!                                Print results
!
!                                Copy the frequencies to the output
!                                matrix
      CALL SCOPY (NF, F, 1, SM(1:,1), 1)
!                                Copy the spectral estimates to the
!                                output matrix
      CALL SCOPY (NF, SX, 1, SM(1:,2), 1)
!                                Call printing routine
      CALL WRRRL (TITLE, SM, RLABEL, CLABEL, FMT=FMT)
!
      END
```

Output

```
Results of Spectral Analysis
            Spectral
Frequency  Estimates
   0.1571   710.8386
   0.3142   116.3940
   0.4712   937.1508
   0.6283  1209.8268
   0.7854   538.9236
   0.9425    84.9561
   1.0996   128.0791
   1.2566    55.0304
   1.4137    40.2022
   1.5708    46.4240
   1.7279    21.0053
   1.8850    12.1449
   2.0420     8.8654
```

```
2.1991    7.2589
2.3562    6.8078
2.5133    3.3873
2.6704    3.9504
2.8274    5.7418
2.9845    4.4652
3.1416    4.1216
```

CPFFT

more...

Computes the cross periodogram of two stationary time series using a fast Fourier transform.

Required Arguments

X — Vector of length **NOBS** containing the first stationary time series. (Input)

Y — Vector of length **NOBS** containing the second stationary time series. (Input)

CPM — (⌊*N*/2⌋ + 1) by 10 matrix containing a summarization of the results of the cross periodogram analysis. (Output)
For *k* = 0, 1, ..., ⌊*N*/2⌋, the (*k* + 1)-st element of the *j*-th column of **CPM** is defined as

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for IFSCAL = 0 or $\omega_k = k/N$ for IFSCAL = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for IFSCAL = 0 and $p_k = 1/\omega_k$ for IFSCAL = 1. If $\omega_k = 0$, p_k is set to missing.
3	X periodogram ordinate, $I_X(\omega_k)$
4	X cosine transformation coefficient, $A_X(\omega_k)$
5	X sine transformation coefficient, $B_X(\omega_k)$
6	Y periodogram ordinate, $I_Y(\omega_k)$
7	Y cosine transformation coefficient, $A_Y(\omega_k)$
8	Y sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.

Optional Arguments

NOBS — Number of observations in each stationary time series X and Y. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine series, and the real and imaginary components of the cross periodogram.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

YCNTR — Constant used to center the time series Y. (Input)
Default: YCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
NPAD must be greater than or equal to zero. The length of each centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL= 0.

IFSCAL	Action
0	Frequency in radians per unit time
1	Frequency in cycles per unit time

IPVER — Option for version of the periodogram. (Input)
Default: IPVER = 0.

IPVER	Action
0	Compute usual periodogram.
1	Compute modified periodogram.

Refer to the "Description" section for further details.

LDCPM — Leading dimension of CPM exactly as specified in the dimension statement of the calling program. (Input)
LDCPM must be greater than or equal to ⌊*N*/2⌋ + 1.
Default: LDCPM = size (CPM,1).

FORTRAN 90 Interface

Generic: CALL CPFFT (X, Y, CPM [, ...])
Specific: The specific interface names are S_CPFFT and D_CPFFT.

FORTRAN 77 Interface

Single: CALL CPFFT (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL, IPVER, CPM, LDCPM)
Double: The double precision name is DCPFFT.

Description

Routine CPFFT computes the cross periodogram of two jointly stationary time series given a sample of n = NOBS observations $\{X_t\}$ and $\{Y_t\}$ for t = 1, 2, ..., n.

Let

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

represent the centered and padded data where *N* = NOBS + NPAD,

$$\widetilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\{\widetilde{Y}_t\} \text{ for } t = 1, \ \ldots, N$$

represent the centered and padded data where

$$\widetilde{Y}_t = \begin{cases} Y_t - \hat{\mu}_Y & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_Y & \mu_Y \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} Y_t & \mu_Y \text{ unknown} \end{cases}$$

The periodogram of the sample sequence $\{X_t\}$, $t = 1, \ldots, n$ computed with the padded sequence

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

is defined by

$$I_{n,N,\widetilde{X}}(\omega_k) = A^2_{\widetilde{X}}(\omega_k) + B^2_{\widetilde{X}}(\omega_k)$$

where

$$A_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t\sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and *K* is the scale factor

$$K = \begin{cases} \frac{2}{n} & \text{for the usual periodogram,} \\ \frac{1}{2\pi n} & \text{for the modified periodogram} \end{cases}$$

The periodogram of the sample sequence $\{Y_t\}$, $t = 1, \ldots, n$ computed with the padded sequence

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ \ldots, N$$

is defined by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t\cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t\sin(\omega_k t)$$

represent the

$$\{\tilde{Y}_t\}$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N} \quad k = 0,\ 1,\ \ldots,\ \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a). The routine PFFT is used to compute the periodograms of both

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

according to the version specified by the argument IPVER. The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\,\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The real part of the (modified) cross periodogram represents the 'raw' sample cospectrum and the negative of the imaginary part of the (modified) cross periodogram represents the 'raw' sample quadrature spectrum (Priestley 1981, page 695). The relationship between the cross periodogram and its complex conjugate is given by

$$I_{n,N,\tilde{X}\tilde{Y}}\left(-\omega_k\right) \equiv I^*_{n,N,\tilde{X}\tilde{Y}}\left(\omega_k\right), \quad 0 \le \omega_k \le \pi$$

and may be used to recover the cross periodogram at negative frequencies.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2FFT/DC2FFT. The reference is:

   ```
   CALL C2FFT (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL, IPVER, CPM, LDCPM,
       CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length *N*.

   COEF — Complex work vector of length *N*.

   WFFTC — Work vector of length 4*N* + 15.

   CPY — Work vector of length 2*N*.

2. The centered and padded time series are defined by

I	AOV(I)
CX(*j*) = X(*j*) - XCNTR	for *j* = 1, …, NOBS
CX(*j*) = 0	for *j* = NOBS + 1, …, *N*
and	
CY(*j*) = Y(*j*) - YCNTR	for *j* = 1, …, NOBS
CY (*j*) = 0	for *j* = NOBS + 1, …, *N* where *N* = NOBS + NPAD.

3. The cross periodogram $I_{XY}(\omega)$ is complex valued in general. The relation $I_{XY}(-\omega) = \text{conj}(I_{XY}(\omega))$ for $w > 0.0$ recovers the cross periodogram for negative frequencies since $\text{real}(I_{XY}(-\omega)) = \text{real}(I_{XY}(\omega))$ and $\text{imag}(I_{XY}(-\omega)) = -\text{imag}(I_{XY}(\omega))$. The periodogram $I(\omega)$ is an even function of the frequency ω. The relation $I(-\omega) = I(\omega)$ for $\omega > 0.0$ recovers the periodogram for negative frequencies.

4. Since $\cos(\omega)$ is an even function of ω and $\sin(\omega)$ is an odd function of ω, the cosine and sine transformations, respectively, satisfy $A(-\omega) = A(\omega)$ and $B(-\omega) = -B(\omega)$ for $\omega > 0.0$. Similarly, the complex Fourier coefficients, stored in COEF, satisfy COEF(-ω) = conj(COEF(ω)).

5. Computation of the 2 * NOBS - 1 cross-covariances of X and Y using the inverse Fourier transform of the cross periodogram requires NPAD = NOBS - 1.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the time period from 1770 through 1869. Application of routine CPFFT to these data produces the following results. Note that CPFFT sets CPM (1, 2) to the missing value code via routine AMACH.The printing of CPM (1, 2) depends on the computer.

```
          USE GDATA_INT
          USE CPFFT_INT
          USE WRRRL_INT

          IMPLICIT   NONE
          INTEGER    LDCPM, LDRDAT, NDRDAT, NOBS, NPAD
          PARAMETER  (LDRDAT=100, NDRDAT=4, NOBS=100, NPAD=NOBS-1, &
                     LDCPM=(NOBS+NPAD)/2+1)
    !
          INTEGER    IPVER, NRCOL, NRROW
          REAL       CPM(LDCPM,10), FLOAT, RDATA(LDRDAT,NDRDAT), &
                     X(NOBS), Y(NOBS)
          CHARACTER  CLABEL1(6)*9, CLABEL2(6)*9, FMT*7, RLABEL(1)*6, &
                      TITLE*41
          INTRINSIC  FLOAT
    !
          EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
```

```
!
      DATA TITLE/'Results of the Cross Periodogram Analysis'/
      DATA FMT/'(F10.3)'/
      DATA CLABEL1/'k+1', 'w(k)', 'p(k)', 'IX(w(k))', 'AX(w(k))', &
          'BX(w(k))'/
      DATA CLABEL2/'k+1', 'IY(w(k))', 'AY(w(k))', 'BY(w(k))', &
          'Real IXY', 'Imag. IXY'/
      DATA RLABEL/'NUMBER'/
!
!                                 Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                 Center on arithmetic means
!                                 Frequency in radians per unit time
!                                 Modified periodogram version
      IPVER = 1
!                                 Compute the cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!
!                                 Print results (First 10 rows)
      CALL WRRRL (TITLE, CPM, RLABEL, CLABEL1, 10, 5, FMT=FMT)
      CALL WRRRL ('%/', CPM(1:,6), RLABEL, CLABEL2, 10, 5, FMT=FMT)
!
      END
```

Output

```
            Results of the Cross Periodogram Analysis
k+1        w(k)        p(k)    IX(w(k))    AX(w(k))    BX(w(k))
  1       0.000         NaN       0.000       0.000       0.000
  2       0.032     199.000     184.159       3.742     -13.044
  3       0.063      99.500    1364.408      35.457     -10.354
  4       0.095      66.333    2433.933      29.411      39.610
  5       0.126      49.750    1351.002     -21.749      29.631
  6       0.158      39.800     140.421     -11.716      -1.773
  7       0.189      33.167      44.117      -4.671       4.722
  8       0.221      28.429     121.186     -11.003      -0.343
  9       0.253      24.875     176.275      -4.782     -12.386
 10       0.284      22.111     144.867      10.038      -6.642

k+1    IY(w(k))    AY(w(k))    BY(w(k))    Real IXY   Imag. IXY
  1       0.000       0.000       0.000       0.000       0.000
  2    1689.212     -37.480     -16.866      79.776    -552.014
  3    4113.003      41.232     -49.122    1970.577   -1314.779
  4    3255.785      44.214      36.068    2729.031    -690.474
  5    1757.663      -8.162      41.122    1396.006    -652.513
  6    1002.050     -30.107       9.778     335.410    -167.954
  7      62.360      -6.825       3.972      50.636      13.678
  8    1481.396     -38.096       5.487     417.288     -73.451
  9    1274.161     -17.176     -31.291     469.704     -63.095
 10     488.479     -12.442     -18.267      -3.570    -265.992
```

CSSWD

more...

Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the time series data.

Required Arguments

X — Vector of length **NOBS** containing the first stationary time series. (Input)

Y — Vector of length **NOBS** containing the second stationary time series. (Input)

F — Vector of length **NF** containing the frequencies at which to evaluate the cross-spectral density estimate. (Input)
The units of **F** correspond to the scale specified by **IFSCAL**. The elements of **F** must be in the range (**-**π/**TINT**, π /**TINT**), inclusive, for **IFSCAL** = 0 and
(**-**1/(2 ***** **TINT**), 1/(2 ***** **TINT**)), inclusive, for **IFSCAL** = 1.

ISWVER — Option for version of the spectral window. (Input)

ISWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the “Algorithm” section for further details.

M — Vector of length **NM** containing the values of the spectral window parameter **M**. (Input)
For the Parzen spectral window (**ISWVER** = 5), all values of the spectral window parameters **M** must be even.

CPM — ($\lfloor N/2 \rfloor + 1$) by 10 matrix containing a summarization of the cross periodogram analysis. (Output)
For $k = 0, 1, \ldots, \lfloor N/2 \rfloor$, the $(k + 1)$-st element of the j-th column of **CPM** is defined as

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for `IFSCAL` = 0 or $\omega_k = k/N$ for `IFSCAL` = 1.
2	Period, p_k where $p_k = 2\pi/\omega_k$ for `IFSCAL` = 0 and $p_k = 1/\omega_k$ for `IFSCAL` = 1. If $\omega_k = 0$, p_k is set to missing.
3	X periodogram ordinate, $I_X(\omega k)$
4	X cosine transformation coefficient, $A_X(\omega_k)$
5	X sine transformation coefficient, $B_X(\omega_k)$
6	Y periodogram ordinate, $I_Y(\omega_k)$
7	Y cosine transformation coefficient, $A_Y(\omega_k)$
8	Y sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.

Note N = **NOBS** + **NPAD**.

CSM — **NF** by (**NM** * 7 + 2) matrix containing a summarization of the cross-spectral analysis. (Output)
The k-th element of the j-th column of **CSM** is defined as

Col	Description
1	Frequency, F(k).
2	Period, p_k where $p_k = 2\pi/$F(k) for `IFSCAL` = 0 and $p_k = 1/$F(k) for `IFSCAL` = 1. If F(k) = 0, p_k is set to missing.
3	spectral density estimate at F(k) using the spectral window parameter `M`(1).
4	Y spectral density estimate at F(k) using the spectral window parameter `M`(1).
5	Cospectrum estimate at F(k) using the spectral window parameter `M`(1).
6	Quadrature spectrum estimate at F(k) using the spectral window parameter `M`(1).
7	Cross-amplitude spectrum estimate at F(k) using the spectral window parameter `M`(1).
8	Phase spectrum estimate at F(k) using the spectral window parameter `M`(1).
9	Coherence estimate at F(k) using the spectral window parameter `M`(1).
⋮	
`NM` * 7 + 2	Coherence estimate at F(k) using the spectral window parameter `M`(`NM`).

where $k = 1, \ldots,$ **NF**.

Optional Arguments

NOBS — Number of observations in each stationary time series X and Y. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the cross periodogram and cross-spectral density estimate based on a specified version of a spectral window for a given set of spectral window parameters.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

YCNTR — Constant used to center the time series Y. (Input)
Default: YCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
NPAD must be greater than or equal to zero. The length of each centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the cross-spectral density estimate. (Input)
Default: NF = size (F,1).

TINT — Time interval at which the series are sampled. (Input)
For a discrete parameter process, usually TINT = 1. For a continuous parameter process, TINT > 0. TINT is used to adjust the cross-spectral density estimate.
Default: TINT = 1.0.

NM — Number of spectral window parameters **M** used to compute the cross-spectral density estimate for a given spectral window version. (Input)
NM must be greater than or equal to one.
Default: **NM** = size (**M**,1).

LDCPM — Leading dimension of **CPM** exactly as specified in the dimension statement of the calling program. (Input)
LDCPM must be greater than or equal to ⌊*N*/2⌋, + 1.
Default: **LDCPM** = size (**CPM**,1).

LDCSM — Leading dimension of **CSM** exactly as specified in the dimension statement of the calling program. (Input)
LDCSM must be greater than or equal to **NF**.
Default: **LDCSM** = size (**CSM**,1).

FORTRAN 90 Interface

Generic: **CALL CSSWD** (**X**, **Y**, **F**, **ISWVER**, **M**, **CPM**, CSM[, ...])
Specific: The specific interface names are **S_CSSWD** and **D_CSSWD**.

FORTRAN 77 Interface

Single: **CALL CSSWD** (**NOBS**, **X**, **Y**, **IPRINT**, **XCNTR**, **YCNTR**, **NPAD**, **IFSCAL**, **NF**, **F**, **TINT**, **ISWVER**, **NM**, **M**, **CPM**, **LDCPM**, **CSM**, **LDCSM**)
Double: The double precision name is **DCSSWD**.

Description

Routine **CSSWD** estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a spectral window given a sample of *n* = **NOBS** observations $\{X_t\}$ and $\{Y_t\}$ for $t = 1, 2, \ldots, n$.

Let

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

represent the centered and padded data where *N* = **NOBS** + **NPAD**,

$$\widetilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\{\widetilde{Y}_t\} \text{ for } t = 1, \ \ldots, N$$

represent the centered and padded data where

$$\widetilde{Y}_t = \begin{cases} Y_t - \hat{\mu}_Y & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_Y & \mu_Y \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} Y_t & \mu_Y \text{ unknown} \end{cases}$$

The modified periodogram of

$$\{\widetilde{X}_t\} \text{ for } t = 1, \ \ldots, N$$

is estimated by

$$I_{n,N,\widetilde{X}}(\omega_k) = A^2_{\widetilde{X}}(\omega_k) + B^2_{\widetilde{X}}(\omega_k)$$

where

$$A_{\widetilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N} \widetilde{X}_t \cos(\omega_k t)$$

and

$$B_{\tilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{X}_t \sin(\omega_k t)$$

represent the

$$\tilde{X}_t$$

cosine and sine transforms, respectively, and *K* is the scale factor equal to 1/(2π*n*). The modified periodogram of

$$\{\tilde{Y}_t\} \text{ for } t = 1, \ \ldots, N$$

is estimated by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t \cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t \sin(\omega_k t)$$

represent the

$$\tilde{Y}_t$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \ k = 0, 1, \ \ldots, \lfloor N/2 \rfloor$$

The routine **PFFT** is used to compute the modified periodograms of both

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \ \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\,\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The routine **CPFFT** is used to compute the modified cross periodogram between

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

The nonnormalized spectral density of X_t is estimated by

$$\hat{h}_X(\omega) = \frac{2\pi}{N}\sum_{k=-\lfloor N/2\rfloor}^{\lfloor N/2\rfloor} I_{n,N,\tilde{X}}(\omega_k)W_n(\omega-\omega_k)$$

and the nonnormalized spectral density of Y_t is estimated by

$$\hat{h}_Y(\omega) = \frac{2\pi}{N}\sum_{k=-\lfloor N/2\rfloor}^{\lfloor N/2\rfloor} I_{n,N,\tilde{Y}}(\omega_k)W_n(\omega-\omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument **ISWVER**. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M}\left\{\frac{\sin(M\theta/2)}{\sin(\theta/2)}\right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$ and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta) \right]^2 \left\{ 1 - \frac{2}{3}\sin^2(\theta/2) \right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{ 1 - \left(\frac{M\theta}{\pi}\right)^2 \right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

The argument **NM** specifies the number of window parameters M and, hence, corresponds to the number of spectral density estimates to be computed for a given spectral window. Note that the same spectral window $W_n(\theta)$ and set of parameters M are used to obtain both

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

The above spectral density formulas assume the data $\{X_t\}$ and $\{Y_t\}$ correspond to a realization of a bivariate discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = **TINT** equal to one. However, if the data correspond to a realization of a bivariate continuous-parameter stationary process recorded at equal time intervals, then the spectral density estimates must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t \hat{h}_X(\omega), \quad |\omega| \le \pi/\Delta t$$

and the estimate of $h_Y(\omega)$ is given by

$$\hat{h}_Y(\omega) = \Delta t \hat{h}_Y(\omega), \quad |\omega| \leq \pi / \Delta t$$

The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale specified by the argument **IFSCAL** but are transformed to the scale of radians per unit time for computational purposes. The frequency **ω** of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval **Δ***t*.

The cross-spectral density function is complex-valued in general and may be written in the following form:

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Re \left\{ I_{n,N,\widetilde{X}\widetilde{Y}}(\omega_k) \right\} W_n(\omega - \omega_k)$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Im \left\{ I_{n,N,\widetilde{X}\widetilde{Y}}(\omega_k) \right\} W_n(\omega - \omega_k)$$

Note that the same spectral window $W_n(\theta)$ and window parameter *M* used to derive

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

are also used to compute

$$\hat{h}_{XY}(\omega)$$

The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates with a similar adjustment for **Δ***t*.

An equivalent representation of $h_{XY}(\omega)$ is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega) e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude spectrum* is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{ \hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega) \right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1} \left\{ -\hat{q}_{XY}(\omega) / \hat{c}_{XY}(\omega) \right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left| \hat{w}_{XY}(\omega) \right| = \left\{ \frac{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega) \hat{h}_Y(\omega)} \right\}^{1/2}$$

The *coherence* or squared coherency is output.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2SWD/DC2SWD**. The reference is:

   ```
   CALL C2SWD (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL, NF, F, TINT, ISWVER,
       NM, M, CPM, LDCPM, CSM, LDCSM, CX, COEF, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CX — Complex work vector of length *N*. (Output)
   COEF — Complex work vector of length *N*. (Output)
   WFFTC — Vector of length 4*N* + 15.
   CPY — Vector of length 2*N*.

2. The centered and padded time series are defined by

   CX(*j*) = **X**(*j*) **- XCNTR** for *j* = 1, ..., **NOBS**

   CX(*j*) = 0 for *j* = **NOBS** + 1, ..., *N*

   and

   CY(*j*) = **Y**(*j*) **- YCNTR** for *j* = 1, ..., **NOBS**

   CY(*j*) = 0 for *j* = **NOBS** + 1, ..., *N*

   where *N* = NOBS + NPAD.

3. The normalized cross-spectral density estimate is obtained by dividing the nonnormalized cross-spectral density estimate in matrix **CSM** by the product of the estimated standard deviation of **X** and the estimated standard deviation of **Y**.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the time period from 1770 through 1869. Application of routine **CSSWD** to these data produces the following results:

```
      USE UMACH_INT
      USE GDATA_INT
      USE CSSWD_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NM, &
                 NOBS, NPAD
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NM=2, &
                  NOBS=100, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD,&
                  LDCPM=N/2+1)
!
      INTEGER    I, ISWVER, J, JPT, M(NM), NOUT, NRCOL, NRROW
      REAL       ASIN, CPM(LDCPM,10), CSM(LDCSM,NM*7+2), F(NF), FLOAT, &
                 PI, RDATA(LDRDAT,NDRDAT), TINT, X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(3)*9, CLABEL2(6)*16, FMT*7, RLABEL(1)*6, &
                 TITLE*80
      INTRINSIC  ASIN, FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
!
      DATA FMT/'(F10.4)'/
      DATA CLABEL1/' k', 'Frequency', 'Period'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
!                                Initialization
      CALL UMACH (2, NOUT)
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                Center on arithmetic means
!                                Frequency in radians per unit time
!                                Spectral window parameters
      M(1) = 10
      M(2) = 30
!                                Time interval for discrete data
!                                Compute cross-spectral density
!                                using the Parzen window
      ISWVER = 5
      CALL CSSWD (X, Y, F, ISWVER, M, CPM, CSM)
!                                Print results
      TITLE = 'Cross-Spectral Analysis Using Parzen Window'
      CALL WRRRL (TITLE, CSM, RLABEL, CLABEL1, NF, 2, FMT=FMT)
      DO 20  J=1, NM
         JPT   = 7*(J-1) + 5
         TITLE = '%/Results of the Cross-Spectral Analysis With '// &
                 'Spectral Window Parameter M = '
```

```
            WRITE (TITLE(77:78),'(I2)') M(J)
            CALL WRRRL (TITLE, CSM(1:,JPT:), RLABEL, CLABEL2, NF, 5, FMT=FMT)
         20 CONTINUE
      !
            END
```

Output

```
      Cross-Spectral Analysis Using Parzen Window
               k   Frequency      Period
               1      0.3142     20.0000
               2      0.6283     10.0000
               3      0.9425      6.6667
               4      1.2566      5.0000
               5      1.5708      4.0000
               6      1.8850      3.3333
               7      2.1991      2.8571
               8      2.5133      2.5000
               9      2.8274      2.2222
              10      3.1416      2.0000

      Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 10
                                          Cross
               k  Cospectrum  Quadrature   Amplitude       Phase   Coherence
               1    463.5888    -65.9763    468.2600      0.1414      0.2570
               2    286.5450    -75.0209    296.2029      0.2561      0.1710
               3    150.1073    -57.8263    160.8604      0.3677      0.1438
               4     52.9840    -32.3642     62.0866      0.5483      0.0998
               5     21.5435    -15.0888     26.3020      0.6110      0.0794
               6     21.4228     -9.8188     23.5658      0.4298      0.1716
               7     15.7005     -5.3704     16.5936      0.3296      0.2112
               8      8.0118     -1.8887      8.2314      0.2315      0.1272
               9      2.7682      0.2007      2.7754     -0.0724      0.0446
              10      0.5777      0.1008      0.5864     -0.1727      0.0091

      Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 30
                                          Cross
               k  Cospectrum  Quadrature   Amplitude       Phase   Coherence
               1    169.7542   -193.4384    257.3615      0.8505      0.1620
               2    452.6187     32.3813    453.7755     -0.0714      0.2213
               3     94.5221    -90.8159    131.0800      0.7654      0.2629
               4     -0.2096     -6.1127      6.1163      1.6051      0.0019
               5     27.4711    -22.1946     35.3166      0.6796      0.2492
               6     29.1329     -4.0128     29.4080      0.1369      0.3170
               7     11.2058     -9.3403     14.5881      0.6948      0.2594
               8      8.0017      0.8813      8.0501     -0.1097      0.1928
               9     -0.4199      2.2893      2.3275     -1.7522      0.0468
              10      0.5570     -1.0767      1.2123      1.0934      0.0678
```

CSSWP

Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the spectral densities and cross periodogram.

Required Arguments

N — Number of observations in each of the appropriately centered and padded time series **X** and **Y**. (Input)
N must be greater than or equal to two.

SX — Vector of length **NF** containing the estimate of the spectral density of the first time series **X**. (Input)

SY — Vector of length **NF** containing the estimate of the spectral density of the second time series **Y**. (Input)

CPREAL — Vector of length $\lfloor N/2 \rfloor + 1$ containing the real part of the cross periodogram between **X** and **Y**. (Input)
The real part of the cross periodogram evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by **CPREAL**$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

CPIMAG — Vector of length $\lfloor N/2 \rfloor + 1$ containing the imaginary part of the cross periodogram between **X** and **Y**. (Input)
The imaginary part of the cross periodogram evaluated at (angular) frequency $w_k = 2\pi k/N$ is given by **CPIMAG**$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length **NF** containing the (angular) frequencies at which the spectral and cross-spectral densities are estimated. (Input)

ISWVER — Option for version of the spectral window. (Input)

SWVER	Action
1	Modified Bartlett
2	Daniell
3	Tukey-Hamming
4	Tukey-Hanning
5	Parzen
6	Bartlett-Priestley

Refer to the “Description” section for further details.

M — Spectral window parameter. (Input)
M must be greater than or equal to one and less than N. For the Parzen spectral window (ISWVER = 5), the spectral window parameter M must be even.

COSPEC — Vector of length NF containing the estimate of the cospectrum. (Output)

QUADRA — Vector of length NF containing the estimate of the quadrature spectrum. (Output)

CRAMPL — Vector of length NF containing the estimate of the cross-amplitude spectrum. (Output)

PHASE — Vector of length NF containing the estimate of the phase spectrum. (Output)

COHERE — Vector of length NF containing the estimate of the coherence or squared coherency. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one. Default: NF = size (F,1).

FORTRAN 90 Interface

Generic: CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, F, ISWVER, M, COSPEC, QUADRA, CRAMPL, PHASE, COHERE [, ...])

Specific: The specific interface names are S_CSSWP and D_CSSWP.

FORTRAN 77 Interface

Single: CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, NF, F, ISWVER, M, COSPEC, QUADRA, CRAMPL, PHASE, COHERE)

Double: The double precision name is DCSSWP.

Description

Routine CSSWP estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a spectral window given the modified cross-periodogram and spectral densities of the appropriately centered and padded data

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

for $t = 1, \ldots, N$.

The routine CPFFT may be used to compute the modified periodograms

$$I_{n,N,\tilde{X}}(\omega_k) \text{ and } I_{n,N,\tilde{Y}}(\omega_k)$$

and cross periodogram

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N},\ k = 0,\ 1,\ \ \ldots,\ \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) Either routine SSWP or routine SWEP may be applied to the periodograms to obtain nonnormalized spectral density estimates

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

over the set of frequencies

$$\omega = f_i, \qquad i = 1, \ldots, n_f$$

where n_f= NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one. Note that the spectral window or weight sequence used to compute

$$\hat{h}_X(\omega)$$

may differ from that used to compute

$$\hat{h}_Y(\omega)$$

The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Re\left\{ I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) \right\} W_n(\omega - \omega_k)$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \frac{2\pi}{N} \sum_{k=-\lfloor N/2 \rfloor}^{\lfloor N/2 \rfloor} \Im \left\{ I_{n,N,\widetilde{X}\widetilde{Y}}(\omega_k) \right\} W_n(\omega - \omega_k)$$

where the spectral window $W_n(\theta)$ is specified by argument `ISWVER`. The following spectral windows $W_n(\theta)$ are available.

Modified Bartlett

$$W_n(\theta) = \frac{1}{2\pi M} \left\{ \frac{\sin(M\theta/2)}{\sin(\theta/2)} \right\}^2 = F_M(\theta)$$

where $F_M(\theta)$ corresponds to the Fejér kernel of order M.

Daniell

$$W_n(\theta) = \begin{cases} M/2\pi & -\pi/M \le \theta \le \pi/M \\ 0 & \text{otherwise} \end{cases}$$

Tukey

$$W_n(\theta) = aD_M\left(\theta - \frac{\pi}{M}\right) + (1-2a)D_M(\theta) + aD_M\left(\theta + \frac{\pi}{M}\right), \quad 0 < a \le 0.25$$

where $D_M(\theta)$ represents the Dirichlet kernel. The Tukey-Hamming window is obtained when $a = 0.23$ and the Tukey-Hanning window is obtained when $a = 0.25$.

Parzen

$$W_n(\theta) = \frac{6\pi}{M}\left[F_{M/2}(\theta) \right]^2 \left\{ 1 - \frac{2}{3}\sin^2(\theta/2) \right\}$$

where M is even. If M is odd, then $M + 1$ is used instead of M in the above formula.

Bartlett-Priestley

$$W_n(\theta) = \begin{cases} \frac{3M}{4\pi}\left\{1 - \left(\frac{M\theta}{\pi}\right)^2\right\} & |\theta| \le \pi/M \\ 0 & |\theta| > \pi/M \end{cases}$$

Only one window parameter *M* may be specified so that only one estimate of h_{XY} (**ω**) is computed. The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates discussed above. However, the particular spectral window used to compute

$$\hat{h}_{XY}(\omega)$$

need not correspond to either the spectral window or the weight sequence used to compute either

$$\hat{h}_X(\omega) \text{ or } \hat{h}_Y(\omega)$$

An equivalent representation of h_{XY}(**ω**) is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega)/\hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The *coherence* or squared coherency is output.

Comments

1. The periodograms of **X** and **Y** and cross periodogram between **X** and **Y** may be computed using the routine CPFFT. The spectral densities of **X** and **Y** may then be estimated using any of the routines SSWD, SWED, SSWP, or SWEP. Thus, different window types and/or weight sequences may be used to

estimate the spectral and cross-spectral densities given either the series or their periodograms. Note that use of the modified periodograms and modified cross periodogram ensures that the scale of the spectral and cross-spectral densities and their estimates is equivalent.

2 The time sampling interval, **TINT**, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (**TINT** > 0.0) involves multiplication of the frequency vector **F** by 1/**TINT** and multiplication of the spectral and cross-spectral density estimates by **TINT**.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply **F** by 1/(2π).

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the years 1770 through 1869. Application of routine **CSSWP** to these data produces the following results.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NM, &
                 NOBS, NPAD
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NM=2, &
                 NOBS=100, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD, &
                 LDCPM=N/2+1)
!
      INTEGER    I, IPVER, ISWVER, J, JPT, JST, M(NM), NRCOL, NRROW
      REAL       COHERE(NF), COSPEC(NF), CPIMAG(LDCPM), &
                 CPM(LDCPM,10), CPREAL(LDCPM), CRAMPL(NF), &
                 CSM(LDCSM,7*NM+2), F(NF), FLOAT, P(NF), PHASE(NF), &
                 PI, PX(LDCPM), PY(LDCPM), QUADRA(NF), &
                 RDATA(LDRDAT,NDRDAT), SX(NF), SY(NF), X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(3)*9, CLABEL2(6)*16, FMT*8, RLABEL(1)*6,&
                 TITLE*80
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
      EQUIVALENCE (PX(1), CPM(1,3)), (PY(1), CPM(1,6))
      EQUIVALENCE (CPREAL(1), CPM(1,9)), (CPIMAG(1), CPM(1,10))
      EQUIVALENCE (CSM(1,1), F(1)), (CSM(1,2), P(1))
!
      DATA FMT/'(F12.4)'/
      DATA CLABEL1/' k', 'Frequency', 'Period'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature',&
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
!                                Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
         P(I) = 2.0*FLOAT(NF)/FLOAT(I)
```

```
   10 CONTINUE
!                                  Robinson Data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                  Center on arithmetic means
!                                  Frequency in radians per unit time
!                                  Modified periodogram version
      IPVER = 1
!                                  Compute cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!                                  Spectral window parameters
      M(1) = 10
      M(2) = 30
!                                  Compute cross-spectral density
!                                  using the Parzen window
!
!                                  Print frequency and period
      TITLE = 'Cross-Spectral Analysis Using Parzen Window'
      CALL WRRRL (TITLE, CSM, RLABEL, CLABEL1, NF, 2, FMT=FMT)
      ISWVER = 5
      DO 20  J=1, NM
!                                  Estimate the spectral densities
         CALL SSWP (N, PX, F, M(J), SX, ISWVER=ISWVER)
         CALL SSWP (N, PY, F, M(J), SY, ISWVER=ISWVER)
!                                  Estimate the cross-spectral density
         CALL CSSWP (N, SX, SY, CPREAL, CPIMAG, F, ISWVER, M(J), &
                     COSPEC, QUADRA, CRAMPL, PHASE, COHERE)
!                                  Copy results to output matrices
         JPT = 7*(J-1) + 2
         JST = 7*(J-1) + 5
         CALL SCOPY (NF, SX, 1, CSM(1:,JPT+1), 1)
         CALL SCOPY (NF, SY, 1, CSM(1:,JPT+2), 1)
         CALL SCOPY (NF, COSPEC, 1, CSM(1:,JPT+3), 1)
         CALL SCOPY (NF, QUADRA, 1, CSM(1:,JPT+4), 1)
         CALL SCOPY (NF, CRAMPL, 1, CSM(1:,JPT+5), 1)
         CALL SCOPY (NF, PHASE, 1, CSM(1:,JPT+6), 1)
         CALL SCOPY (NF, COHERE, 1, CSM(1:,JPT+7), 1)
!                                  Print results
         TITLE = '%/Results of the Cross-Spectral Analysis With '// &
                 'Spectral Window Parameter M = '
         WRITE (TITLE(77:78),'(I2)') M(J)
         CALL WRRRL (TITLE, CSM(1:,JST), RLABEL, CLABEL2, NF, 5, FMT=FMT)
   20 CONTINUE
!
      END
```

Output

```
 Cross-Spectral Analysis Using Parzen Window
        k     Frequency        Period
        1        0.3142       20.0000
        2        0.6283       10.0000
        3        0.9425        6.6667
        4        1.2566        5.0000
        5        1.5708        4.0000
        6        1.8850        3.3333
        7        2.1991        2.8571
        8        2.5133        2.5000
        9        2.8274        2.2222
```

```
          10        3.1416        2.0000

    Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 10
                                          Cross
       k    Cospectrum    Quadrature     Amplitude         Phase     Coherence
       1      463.5888      -65.9763      468.2600        0.1414        0.2570
       2      286.5450      -75.0209      296.2029        0.2561        0.1710
       3      150.1073      -57.8263      160.8604        0.3677        0.1438
       4       52.9840      -32.3642       62.0866        0.5483        0.0998
       5       21.5435      -15.0888       26.3020        0.6110        0.0794
       6       21.4228       -9.8188       23.5658        0.4298        0.1716
       7       15.7005       -5.3704       16.5936        0.3296        0.2112
       8        8.0118       -1.8887        8.2314        0.2315        0.1272
       9        2.7682        0.2007        2.7754       -0.0724        0.0446
      10        0.5777        0.1008        0.5864       -0.1727        0.0091

    Results of the Cross-Spectral Analysis With Spectral Window Parameter M = 30
                                          Cross
       k    Cospectrum    Quadrature     Amplitude         Phase     Coherence
       1      169.7542     -193.4384      257.3615        0.8505        0.1620
       2      452.6187       32.3813      453.7755       -0.0714        0.2213
       3       94.5221      -90.8159      131.0800        0.7654        0.2629
       4       -0.2096       -6.1127        6.1163        1.6051        0.0019
       5       27.4711      -22.1946       35.3166        0.6796        0.2492
       6       29.1329       -4.0128       29.4080        0.1369        0.3170
       7       11.2058       -9.3403       14.5881        0.6948        0.2594
       8        8.0017        0.8813        8.0501       -0.1097        0.1928
       9       -0.4199        2.2893        2.3275       -1.7522        0.0468
      10        0.5570       -1.0767        1.2123        1.0934        0.0678
```

CSWED

more...

Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the time series data.

Required Arguments

X — Vector of length **NOBS** containing the first stationary time series. (Input)

Y — Vector of length **NOBS** containing the second stationary time series. (Input)

F — Vector of length **NF** containing the frequencies at which to evaluate the cross-spectral density estimate. (Input)
The units of **F** correspond to the scale specified by **IFSCAL**. The elements of **F** must be in the range (**-**π/**TINT**, π/**TINT**) inclusive, for **IFSCAL** = 0 and
(**-**1/(2 ***** **TINT**), 1/(2 ***** **TINT**)) inclusive, for **IFSCAL** = 1.

WT — Vector of length **NWT** containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in **WT** normalized to sum to 1 with the current periodogram ordinate taking the middle weight for **NWT** odd or the weight to the right of the middle for **NWT** even.

CPM — (⌊*N*/2⌋ + 1) by 10 matrix containing a summarization of the cross periodogram analysis. (Output)
For *k* = 0, 1, …, ⌊*N*/2⌋, the (*k* + 1)-st element of the *j*-th column of **CPM** is defined as

Col	Description
1	Frequency, ω_k where $\omega_k = 2\pi k/N$ for **IFSCAL** = 0 or $\omega_k = k/N$ for **IFSCAL** = 1
2	Period, p_k where $p_k = 2\pi/\omega_k$ for **IFSCAL** = 0 and $p_k = 1/\omega_k$ for **IFSCAL** = 1. If $\omega_k = 0$, p_k is set to missing.
3	X periodogram ordinate, $I_X(\omega_k)$
4	X cosine transformation coefficient, $A_X(\omega_k)$
5	X sine transformation coefficient, $B_X(\omega_k)$

Col	Description
6	Y periodogram ordinate, $I_Y(\omega_k)$
7	Y cosine transformation coefficient, $A_Y(\omega_k)$
8	Y sine transformation coefficient, $B_Y(\omega_k)$
9	Real part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.
10	Imaginary part of the XY cross periodogram ordinate $I_{XY}(\omega_k)$.

CSM — NF by 9 matrix containing a summarization of the cross-spectral analysis. (Output)
The *k*-th element of the *j*-th column of CSM is defined as

Col	Description
1	Frequency, F(*k*).
2	Period, p_k where $p_k = 2\pi/F(k)$ for IFSCAL = 0 and $p_k = 1/F(k)$ for IFSCAL = 1. If F(*k*) = 0, p_k is set to missing.
3	X spectral density estimate at F(*k*) using the specified relative weights contained in WT.
4	Y spectral density estimate at F(*k*) using the specified relative weights contained in WT.
5	Co-spectrum estimate at F(*k*) using the specified relative weights contained in WT.
6	Quadrature spectrum estimate at F(*k*) using the specified relative weights contained in WT.
7	Cross-amplitude spectrum estimate at F(*k*).
8	Phase spectrum estimate at F(*k*).
9	Coherence estimate at F(*k*).

where *k* = 1, …, NF.

Optional Arguments

NOBS — Number of observations in each stationary time series X and Y. (Input)
NOBS must be greater than or equal to two.
Default: NOBS = size (X,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Prints the periodogram, cosine and sine transformations of each centered and padded time series, the real and imaginary components of the cross periodogram, and the cross-spectral density estimate based on a specified weight sequence.

XCNTR — Constant used to center the time series X. (Input)
Default: XCNTR = the arithmetic mean.

YCNTR — Constant used to center the time series Y. (Input)
Default: YCNTR = the arithmetic mean.

NPAD — Number of zeroes used to pad each centered time series. (Input)
NPAD must be greater than or equal to zero. The length of each centered and padded time series is N = NOBS + NPAD.
Default: NPAD = NOBS – 1.

IFSCAL — Option for frequency scale. (Input)
Default: IFSCAL = 0.

IFSCAL	Action
0	Frequency in radians per unit time.
1	Frequency in cycles per unit time.

NF — Number of frequencies at which to evaluate the cross-spectral density estimate. (Input)
Default: NF = size (F,1).

TINT — Time interval at which the series are sampled. (Input)
For a discrete parameter process, usually TINT = 1.0. For a continuous parameter process, TINT > 0.0. TINT is used to adjust the cross-spectral density estimate.
Default: TINT = 1.0.

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT= size (WT,1).

LDCPM — Leading dimension of CPM exactly as specified in the dimension statement of the calling program. (Input)
LDCPM must be greater than or equal to $\lfloor N/2 \rfloor + 1$.
Default: LDCPM = size (CPM,1).

LDCSM — Leading dimension of **CSM** exactly as specified in the dimension statement of the calling program. (Input)
LDCSM must be greater than or equal to **NF**.
Default: **LDCSM** = size (**CSM**,1).

FORTRAN 90 Interface

Generic: **CALL CSWED** (**X**, **Y**, **F**, **WT**, **CPM**, **CSM** [, ...])
Specific: The specific interface names are **S_CSWED** and **D_CSWED**.

FORTRAN 77 Interface

Single: **CALL CSWED** (**NOBS**, **X**, **Y**, **IPRINT**, **XCNTR**, **YCNTR**, **NPAD**, **IFSCAL**, **NF**, **F**, **TINT**, **NWT**, **WT**, **CPM**, **LDCPM**, **CSM**, **LDCSM**)
Double: The double precision name is **DCSWED**.

Description

Routine **CSWED** estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a fixed sequence of weights given a sample of n = **NOBS** observations $\{X_t\}$ and $\{Y_t\}$ for t = 1, 2, ..., n. Let

$$\left\{\tilde{X}_t\right\}$$

for t = 1, ..., N represent the centered and padded data where N = **NOBS** + **NPAD**,

$$\tilde{X}_t = \begin{cases} X_t - \hat{\mu}_X & t = 1, \ \ldots, n \\ 0 & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_X = \text{XCNTR}$$

is determined by

$$\hat{\mu}_X = \begin{cases} \mu_X, & \mu_X \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} X_t, & \mu_X \text{ unknown} \end{cases}$$

Similarly, let

$$\left\{\widetilde{Y}_t\right\}$$

for $t = 1, \ldots, N$ represent the centered and padded data where

$$\widetilde{Y}_t = \begin{cases} Y_t - \hat{\mu}_Y, & t = 1, \ \ldots, n \\ 0, & t = (n+1), \ \ldots, N \end{cases}$$

and

$$\hat{\mu}_Y = \text{YCNTR}$$

is determined by

$$\hat{\mu}_Y = \begin{cases} \mu_{Y,} & \mu_Y \text{ known} \\ \frac{1}{n}\sum_{t=1}^{n} Y_{t,} & \mu_Y \text{ unknown} \end{cases}$$

The modified periodogram of

$$\left\{\widetilde{X}_t\right\}$$

for $t = 1, \ldots, N$ is estimated by

$$I_{n,N,\widetilde{X}}(\omega_k) = A^2_{\widetilde{X}}(\omega_k) + B^2_{\widetilde{X}}(\omega_k)$$

where

$$A_{\widetilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\widetilde{X}_t\cos(\omega_k t)$$

and

$$B_{\widetilde{X}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\widetilde{X}\sin(\omega_k t)$$

represent the

$$\widetilde{X}_t$$

cosine and sine transforms, respectively, and K is the scale factor equal to $1/(2\pi n)$. The modified periodogram of $\{Y_t\}$ for $t = 1, \ldots, N$ is estimated by

$$I_{n,N,\tilde{Y}}(\omega_k) = A^2_{\tilde{Y}}(\omega_k) + B^2_{\tilde{Y}}(\omega_k)$$

where

$$A_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t\cos(\omega_k t)$$

and

$$B_{\tilde{Y}}(\omega_k) = K^{1/2}\sum_{t=1}^{N}\tilde{Y}_t\sin(\omega_k t)$$

represent the

$$\tilde{Y}_t$$

cosine and sine transforms, respectively. Since the periodogram is an even function of the frequency, it is sufficient to estimate the periodogram at the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0,\ 1,\ \ldots,\ \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a). The routine PFFT is used to compute the modified periodograms of both

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

The computational formula for the cross periodogram is given by

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k) = \Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} + i\,\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\}$$

where

$$\Re\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k) + B_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k)$$

and

$$\Im\left\{I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)\right\} = A_{\tilde{X}}(\omega_k)B_{\tilde{Y}}(\omega_k) - B_{\tilde{X}}(\omega_k)A_{\tilde{Y}}(\omega_k)$$

The routine CPFFT is used to compute the modified cross periodogram between

$$\left\{\tilde{X}_t\right\} \text{ and } \left\{\tilde{Y}_t\right\}$$

The nonnormalized spectral density of X_t is estimated by

$$\hat{h}_X(\omega) = \sum_j \omega_j I_{n,N,\tilde{X}}(\omega_{k,j})$$

and the nonnormalized spectral density of Y_t is estimated by

$$\hat{h}_Y(\omega) = \sum_j \omega_j I_{n,N,\tilde{Y}}(\omega_{k,j})$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega)j\}}{N}$$

and $k(\omega)$ is the integer such that $\omega_{k,0}$ is closest to ω. The sequence of m = NWT weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ satisfies $\Sigma_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency ω at which to estimate the spectral density. Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. The argument WT may contain relative weights since they are normalized to sum to one in the actual computations. The above spectral density formulas assume the data $\{X_t\}$ and $\{Y_t\}$ correspond to a realization of a bivariate discrete-parameter stationary process observed consecutively in time. In this case, the observations are equally spaced in time with interval Δt = TINT equivalent to one. However, if the data correspond to a realization of a bivariate continuous-parameter stationary process recorded at equal time intervals, then the spectral density estimates must be adjusted for the effect of aliasing. In general, the estimate of $h_X(\omega)$ is given by

$$\hat{h}_X(\omega) = \Delta t\, \hat{h}_X(\omega) \quad |\omega| \le \pi/\Delta t$$

and the estimate of $h_Y(\omega)$ is given by

$$\hat{h}_Y(\omega) = \Delta t\, \hat{h}_Y(\omega), \quad |\omega| \le \pi/\Delta t$$

The nonnormalized spectral density is estimated over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f= **NF**. These frequencies are in the scale specified by the argument **IFSCAL** but are transformed to the scale of radians per unit time for computational purposes. The frequency **ω** of the desired spectral estimate is assumed to be input in a form already adjusted for the time interval **Δ***t*. The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \sum_j w_j \, \Re \left\{ I_{n,N,\widetilde{X}\widetilde{Y}}(\omega_{k,j}) \right\}$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \sum_j w_j \, \Im \left\{ I_{n,N,\widetilde{X}\widetilde{Y}}(\omega_{k,j}) \right\}$$

Note that the same sequence of weights $\{w_j\}$ used to estimate

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

is used to estimate

$$\hat{c}_{XY}(\omega) \text{ and } \hat{q}_{XY}(\omega)$$

The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates discussed above with a similar adjustment for **Δ***t*. An equivalent representation of $h_{XY}(\omega)$ is the polar form defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega) e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{ \hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega) \right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1} \left\{ -\hat{q}_{XY}(\omega) / \hat{c}_{XY}(\omega) \right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left| \hat{w}_{XY}(\omega) \right| = \left\{ \frac{\hat{c}_{XY}^2(w) + \hat{q}_{XY}^2(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)} \right\}^{1/2}$$

The *coherence* or squared coherency is output.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2WED/DC2WED**. The reference is:

   ```
   CALL C2WED (NOBS, X, Y, IPRINT, XCNTR, YCNTR, NPAD, IFSCAL, NF, F, TINT, NWT, WT,
       CPM, LDCPM, CSM, LDCSM, CWK, COEFWK, WFFTC, CPY)
   ```

   The additional arguments are as follows:

   CWK — Complex work vector of length *N*. (Output)

   COEFWK — Complex work vector of length *N*. (Output)

   WFFTC — Vector of length 4*N* + 15.

   CPY — Vector of length 2*N*.

2. The normalized cross-spectral density estimate is obtained by dividing the nonnormalized cross-spectral density estimate in matrix **CSM** by the product of the estimated standard deviation of **X** and the estimated standard deviation of **Y**.

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the years 1770 through 1869. Application of routine **CSWED** to these data produces the following results.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NOBS, &
                 NPAD, NWT
      PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NOBS=100, &
                 NWT=7, LDCSM=NF, NPAD=NOBS-1, N=NPAD+NOBS, &
                 LDCPM=N/2+1)
!
      INTEGER    I, NRCOL, NRROW
      REAL       CPM(LDCPM,10), CSM(LDCSM,9), F(NF), FLOAT, PI, &
                 RDATA(LDRDAT,NDRDAT), WT(NWT), X(NOBS), Y(NOBS)
      CHARACTER  CLABEL1(5)*24, CLABEL2(6)*16, FMT*7, RLABEL(1)*6, &
                 TITLE1*32, TITLE2*40
      INTRINSIC  FLOAT
!
      EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
!
      DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
```

```
      DATA FMT/'(F12.4)'/
      DATA CLABEL1/'%/%/ k', '%/%/Frequency', '%/%/Period', &
          'Spectral%/Estimate%/of X', 'Spectral%/Estimate%/of Y'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
      DATA TITLE1/'Results of the Spectral Analyses'/
      DATA TITLE2/'%/Results of the Cross-Spectral Analysis'/
!                                 Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
   10 CONTINUE
!                                 Robinson data
      CALL GDATA (8, RDATA, NRROW, NRCOL)
!                                 Center on arithmetic means
!                                 Frequency in radians per unit time
!                                 Time interval for discrete data
!                                 Compute the cross periodogram
      CALL CSWED (X, Y, F, WT, CPM, CSM)
!                                 Print results
      CALL WRRRL (TITLE1, CSM, RLABEL, CLABEL1, NF, 4, FMT=FMT)
      CALL WRRRL (TITLE2, CSM(1:,5), RLABEL, CLABEL2, NF, 5, FMT=FMT)
!
      END
```

Output

```
               Results of the Spectral Analyses
                                    Spectral      Spectral
                                    Estimate      Estimate
 k     Frequency        Period          of X          of Y
 1        0.3142       20.0000      116.9550     1315.8370
 2        0.6283       10.0000     1206.6086     1005.1219
 3        0.9425        6.6667       84.8369      317.2589
 4        1.2566        5.0000       55.2120      270.2111
 5        1.5708        4.0000       46.5748      115.6768
 6        1.8850        3.3333       12.4050      250.0125
 7        2.1991        2.8571        7.0934       82.6773
 8        2.5133        2.5000        3.4091       62.3267
 9        2.8274        2.2222        5.6828       12.8970
10        3.1416        2.0000        4.0346       17.6441

                  Results of the Cross-Spectral Analysis
                                      Cross
 k    Cospectrum    Quadrature     Amplitude         Phase     Coherence
 1       94.0531     -254.0125      270.8659        1.2162        0.4767
 2      702.5118       21.9823      702.8557       -0.0313        0.4073
 3       70.2379      -31.4431       76.9547        0.4209        0.2200
 4       -1.8715      -36.1639       36.2123        1.6225        0.0879
 5       36.6366      -18.5925       41.0843        0.4696        0.3133
 6       32.7071       -6.6569       33.3776        0.2008        0.3592
 7        9.4887       -9.1692       13.1950        0.7683        0.2969
 8        9.2534       -0.3000        9.2583        0.0324        0.4034
 9       -0.5568        2.9455        2.9977       -1.7576        0.1226
10        1.7640       -1.8321        2.5433        0.8043        0.0909
```

CSWEP

Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the spectral densities and cross periodogram.

Required Arguments

N — Number of observations in each of the appropriately centered and padded time series **X** and **Y**. (Input)
N must be greater than or equal to two.

SX — Vector of length **NF** containing the estimate of the spectral density of the first time series **X**. (Input)

SY — Vector of length **NF** containing the estimate of the spectral density of the second time series **Y**. (Input)

CPREAL — Vector of length $\lfloor N/2 \rfloor + 1$ containing the real part of the cross periodogram between **X** and **Y**. (Input)
The real part of the cross periodogram evaluated at (angular) frequency $\omega_k = 2\pi k/N$ is given by **CPREAL**$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

CPIMAG — Vector of length $\lfloor N/2 \rfloor + 1$ containing the imaginary part of the cross periodogram between **X** and **Y**. (Input)
The imaginary part of the cross periodogram evaluated at (angular) frequency $\omega_k = 2\pi k/N$ is given by **CPIMAG**$(k + 1)$, $k = 0, 1, \ldots, \lfloor N/2 \rfloor$.

F — Vector of length **NF** containing the (angular) frequencies at which the spectral density is estimated. (Input)

WT — Vector of length **NWT** containing the weights used to smooth the periodogram. (Input)
The actual weights are the values in **WT** normalized to sum to 1 with the current periodogram ordinate taking the middle weight for **NWT** odd or the weight to the right of the middle for **NWT** even.

COSPEC — Vector of length **NF** containing the estimate of the cospectrum. (Output)

QUADRA — Vector of length **NF** containing the estimate of the quadrature spectrum. (Output)

CRAMPL — Vector of length **NF** containing the estimate of the cross-amplitude spectrum. (Output)

PHASE — Vector of length **NF** containing the estimate of the phase spectrum. (Output)

COHERE — Vector of length **NF** containing the estimate of the coherence. (Output)

Optional Arguments

NF — Number of (angular) frequencies. (Input)
NF must be greater than or equal to one.
Default: NF = size (F,1).

NWT — Number of weights. (Input)
NWT must be greater than or equal to one.
Default: NWT = size (WT,1).

FORTRAN 90 Interface

Generic: CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, F, WT, COSPEC, QUADRA, CRAMPL, PHASE, COHERE [, ...])

Specific: The specific interface names are S_CSWEP and D_CSWEP.

FORTRAN 77 Interface

Single: CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, NF, F, NWT, WT, COSPEC, QUADRA, CRAMPL, PHASE, COHERE)

Double: The double precision name is DCSWEP.

Description

Routine CSWEP estimates the nonnormalized cross-spectral density function of two jointly stationary time series using a fixed sequence of weights given the modified cross-periodogram and spectral densities of the appropriately centered and padded data

$$\left\{\tilde{X}_t\right\}$$

and

$$\left\{\tilde{Y}_t\right\}$$

for $t = 1, \ldots, N$. The routine CPFFT may be used to compute the modified periodograms

$$I_{n,N,\tilde{X}}\left(\omega_k\right) \text{ and } I_{n,N,\tilde{Y}}\left(\omega_k\right)$$

and cross-periodogram

$$I_{n,N,\tilde{X}\tilde{Y}}(\omega_k)$$

over the discrete set of nonnegative frequencies

$$\omega_k = \frac{2\pi k}{N}, \quad k = 0, 1, \ldots, \lfloor N/2 \rfloor$$

(Here, $\lfloor a \rfloor$ means the greatest integer less than or equal to a.) Either routine SSWP or routine SWEP may be applied to the periodograms to obtain nonnormalized spectral density estimates

$$\hat{h}_X(\omega) \text{ and } \hat{h}_Y(\omega)$$

over the set of frequencies

$$\omega = f_i, \quad i = 1, \ldots, n_f$$

where n_f= NF. These frequencies are in the scale of radians per unit time. The time sampling interval Δt is assumed to be equal to one. Note that the spectral window or weight sequence used to compute

$$\hat{h}_X(\omega)$$

may differ from that used to compute

$$\hat{h}_Y(\omega)$$

The cross-spectral density function is complex-valued in general and may be written as

$$h_{XY}(\omega) = c_{XY}(\omega) - iq_{XY}(\omega)$$

The *cospectrum* is estimated by

$$\hat{c}_{XY}(\omega) = \sum_j w_j \, \Re \left\{ I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j}) \right\}$$

and the *quadrature spectrum* is estimated by

$$\hat{q}_{XY}(\omega) = \sum_j w_j \, \Im \left\{ I_{n,N,\tilde{X}\tilde{Y}}(\omega_{k,j}) \right\}$$

where

$$\omega_{k,j} = \frac{2\pi\{k(\omega) + j\}}{N}$$

and $k(\boldsymbol{\omega})$ is the integer such that $\boldsymbol{\omega}_{k,0}$ is closest to $\boldsymbol{\omega}$. The sequence of m = **NWT** weights $\{w_j\}$ for $j = -\lfloor m/2 \rfloor, \ldots, (m - \lfloor m/2 \rfloor - 1)$ satisfies $\boldsymbol{\Sigma}_j w_j = 1$. These weights are fixed in the sense that they do not depend on the frequency $\boldsymbol{\omega}$ at which to estimate $h_{XY}(\boldsymbol{\omega})$. Usually, m is odd with the weights symmetric about the middle weight w_0. If m is even, the weight to the right of the middle is considered w_0. The argument **WT** may contain relative weights since they are normalized to sum to one in the actual computations. The nonnormalized cross-spectral density estimate is computed over the same set of frequencies as the nonnormalized spectral density estimates. However, the particular weight sequence used to compute

$$\hat{h}_{XY}(\omega)$$

need not correspond to either the weight sequence or spectral window used to compute either

$$\hat{h}_X(\omega) \text{ or } \hat{h}_Y(\omega)$$

An equivalent representation of $h_{XY}(\boldsymbol{\omega})$ is the *polar form* defined by

$$h_{XY}(\omega) = \alpha_{XY}(\omega)e^{i\phi_{XY}(\omega)}$$

The *cross-amplitude* spectrum is estimated by

$$\hat{\alpha}_{XY}(\omega) = \left\{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)\right\}^{1/2}$$

and the *phase spectrum* is estimated by

$$\hat{\phi}_{XY}(\omega) = \tan^{-1}\left\{-\hat{q}_{XY}(\omega)/\hat{c}_{XY}(\omega)\right\}$$

Finally, the *coherency spectrum* is estimated by

$$\left|\hat{w}_{XY}(\omega)\right| = \left\{\frac{\hat{c}^2_{XY}(\omega) + \hat{q}^2_{XY}(\omega)}{\hat{h}_X(\omega)\hat{h}_Y(\omega)}\right\}^{1/2}$$

The coherence or squared coherency is output.

Comments

1. The periodograms of X and Y and cross periodogram between X and Y may be computed via the routine CPFFT. The spectral densities of X and Y may then be estimated using any of the routines SSWD, SWED, SSWP, or SWEP. Thus, different window types and/or weight sequences may be used to estimate the spectral and cross-spectral densities given either the series or their periodograms. Note that use of the modified periodograms and modified cross periodogram ensures that the scales of the spectral and cross-spectral densities and their estimates are equivalent.

2 The time sampling interval, TINT, is assumed to be equal to one. This assumption is appropriate for discrete parameter processes. The adjustment for continuous parameter processes (TINT > 0.0) involves multipication of the frequency vector F by 1/TINT and multiplication of the spectral and cross-spectral density estimates by TINT.

3. To convert the frequency scale from radians per unit time to cycles per unit time, multiply F by 1/(2π).

Example

Consider the Robinson Multichannel Time Series Data (Robinson 1967, page 204) where *X* is the Wölfer sunspot number and *Y* is the northern light activity for the years 1770 through 1869. Application of routine CSWEP to these data produces the following results.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    LDCPM, LDCSM, LDRDAT, N, NDRDAT, NF, NOBS, &
                     NPAD, NWT
          PARAMETER  (LDRDAT=100, NDRDAT=4, NF=10, NOBS=100, &
                     NWT=7, LDCSM=NF, NPAD=NOBS-1, N=NOBS+NPAD, &
                     LDCPM=N/2+1)
    !
          INTEGER    I, IPVER, NROW, NVAR
          REAL       COHERE(NF), COSPEC(NF), CPIMAG(LDCPM), &
                     CPM(LDCPM,10), CPREAL(LDCPM), CRAMPL(NF), &
                     CSM(LDCSM,9), F(NF), FLOAT, PHASE(NF), PI, PX(LDCPM), &
                     PY(LDCPM), QUADRA(NF), RDATA(LDRDAT,NDRDAT), &
                     SX(NF), SY(NF), WT(NWT), X(NOBS), Y(NOBS)
          CHARACTER  CLABEL1(5)*24, CLABEL2(6)*16, FMT*8, RLABEL(1)*6, &
                     TITLE1*32, TITLE2*40
          INTRINSIC  FLOAT

    !
          EQUIVALENCE (X(1), RDATA(1,2)), (Y(1), RDATA(1,3))
          EQUIVALENCE (PX(1), CPM(1,3)), (PY(1), CPM(1,6))
          EQUIVALENCE (CPREAL(1), CPM(1,9)), (CPIMAG(1), CPM(1,10))
    !
          DATA WT/1.0, 2.0, 3.0, 4.0, 3.0, 2.0, 1.0/
          DATA FMT/'(F12.4)'/
          DATA CLABEL1/'%/%/ k', '%/%/Frequency', '%/%/Period', &
```

```
             'Spectral%/Estimate%/of X', 'Spectral%/Estimate%/of Y'/
      DATA CLABEL2/'%/ k', '%/Cospectrum', '%/Quadrature', &
          'Cross%/Amplitude', '%/Phase', '%/Coherence'/
      DATA RLABEL/'NUMBER'/
      DATA TITLE1/'Results of the Spectral Analyses'/
      DATA TITLE2/'%/Results of the Cross-Spectral Analysis'/
!                                 Initialization
      PI = 2.0*ASIN(1.0)
      DO 10  I=1, NF
         F(I) = PI*FLOAT(I)/FLOAT(NF)
         CALL SCOPY (NF, F, 1, CSM(1:,1), 1)
         CSM(I,2) = 2.0*FLOAT(NF)/FLOAT(I)
   10 CONTINUE
!                                 Robinson data
      CALL GDATA (8, RDATA, NROW, NVAR)
!                                 Center on arithmetic means
!                                 Frequency in radians per unit time
!                                 Modified periodogram version
      IPVER = 1
!                                 Compute the cross periodogram
      CALL CPFFT (X, Y, CPM, IPVER=IPVER)
!                                 Estimate the spectral densities
      CALL SWEP (N, PX, F, WT, SX)
      CALL SWEP (N, PY, F, WT, SY)
!                                 Estimate the cross-spectral density
      CALL CSWEP (N, SX, SY, CPREAL, CPIMAG, F, WT, COSPEC, &
                  QUADRA, CRAMPL, PHASE, COHERE)
!                                 Print results
!
!                                 Copy results to output matrices
      CALL SCOPY (NF, SX, 1, CSM(1:,3), 1)
      CALL SCOPY (NF, SY, 1, CSM(1:,4), 1)
      CALL SCOPY (NF, COSPEC, 1, CSM(1:,5), 1)
      CALL SCOPY (NF, QUADRA, 1, CSM(1:,6), 1)
      CALL SCOPY (NF, CRAMPL, 1, CSM(1:,7), 1)
      CALL SCOPY (NF, PHASE, 1, CSM(1:,8), 1)
      CALL SCOPY (NF, COHERE, 1, CSM(1:,9), 1)
!                                 Call printing routines
      CALL WRRRL (TITLE1, CSM, RLABEL, CLABEL1, NF, 4, FMT=FMT)
      CALL WRRRL (TITLE2, CSM(1:,5), RLABEL, CLABEL2, NF, 5, FMT=FMT)
!
      END
```

Output

Results of the Spectral Analyses

k	Frequency	Period	Spectral Estimate of X	Spectral Estimate of Y
1	0.3142	20.0000	116.9550	1315.8370
2	0.6283	10.0000	1206.6086	1005.1219
3	0.9425	6.6667	84.8369	317.2589
4	1.2566	5.0000	55.2120	270.2111
5	1.5708	4.0000	46.5748	115.6768
6	1.8850	3.3333	12.4050	250.0125
7	2.1991	2.8571	7.0934	82.6773
8	2.5133	2.5000	3.4091	62.3267
9	2.8274	2.2222	5.6828	12.8970
10	3.1416	2.0000	4.0346	17.6441

Results of the Cross-Spectral Analysis

k	Cospectrum	Quadrature	Cross Amplitude	Phase	Coherence
1	94.0531	-254.0125	270.8659	1.2162	0.4767
2	702.5118	21.9823	702.8557	-0.0313	0.4073
3	70.2379	-31.4431	76.9547	0.4209	0.2200
4	-1.8715	-36.1639	36.2123	1.6225	0.0879
5	36.6366	-18.5925	41.0843	0.4696	0.3133
6	32.7071	-6.6569	33.3776	0.2008	0.3592
7	9.4887	-9.1692	13.1950	0.7683	0.2969
8	9.2534	-0.3000	9.2583	0.0324	0.4034
9	-0.5568	2.9455	2.9977	-1.7576	0.1226
10	1.7640	-1.8321	2.5433	0.8043	0.0909

Covariance Structures and Factor Analysis

Routines

Usage Notes

Notation that is consistently used throughout this chapter is given in the following table. The FORTRAN equivalent of the symbols used are also given.

Notation Used

Symbol	FORTRAN Symbol	Meaning
p	NVAR	Number of variables in the observed variables
k	NF	Number of factors
Σ	COV	Population (or sample) covariance (correlation) matrix
A	A	Unrotated factor loadings
B	B	Rotated factor loadings
T	T	Factor rotation matrix
	TI	Image transformation matrix
β	SCOEF	Factor score coefficients

The routines in this chapter can generally be used for one or more of several purposes. Among these purposes are the following:

1. Data description: The information in the data is summarized by the factor loadings or by the eigenvectors and eigenvalues.
2. Data reduction: The information in a multivariate sample is reduced to a much smaller number of factors or principal components.
3. Variable clustering: The principal component coefficients or factor loadings lead to a grouping (clustering) of the variables.
4. Model building: Linear models relating the variables to the factors or principal components are estimated. Hypothesis tests may be used to obtain parsimonious and/or other descriptions of the data.

Principal Components

The idea in principal components is to find a small number of linear combinations of the original variables that maximize the variance accounted for in the original data. This amounts to an eigensystem analysis of the covariance (or correlation) matrix. In addition to the eigensystem analysis, routine `PRINC` computes standard errors for the eigenvalues. Correlations of the original variables with the principal component scores are also computed.

The computation of common principal components via routine `KPRIN` is equivalent to finding the "eigenvectors" that best simultaneously diagonalize one or more variance-covariance matrices. For only one input variance-covariance matrix, the vectors computed actually are the eigenvectors of the matrix.

Factor Analysis

Factor analysis and principal component analysis, while quite different in assumptions, often serve the same ends. Unlike principal components in which linear combinations yielding the highest possible variances are obtained, factor analysis generally obtains linear combinations of the observed variables according to a model relating the observed variables to hypothesized underlying factors, plus a random error term called the unique error or uniqueness. In factor analysis, the unique errors associated with each variable are usually assumed to be independent of the factors. In addition, in the common factor model, the unique errors are assumed to be mutually independent. The factor analysis model is

$$x - \mu = \Lambda f + e$$

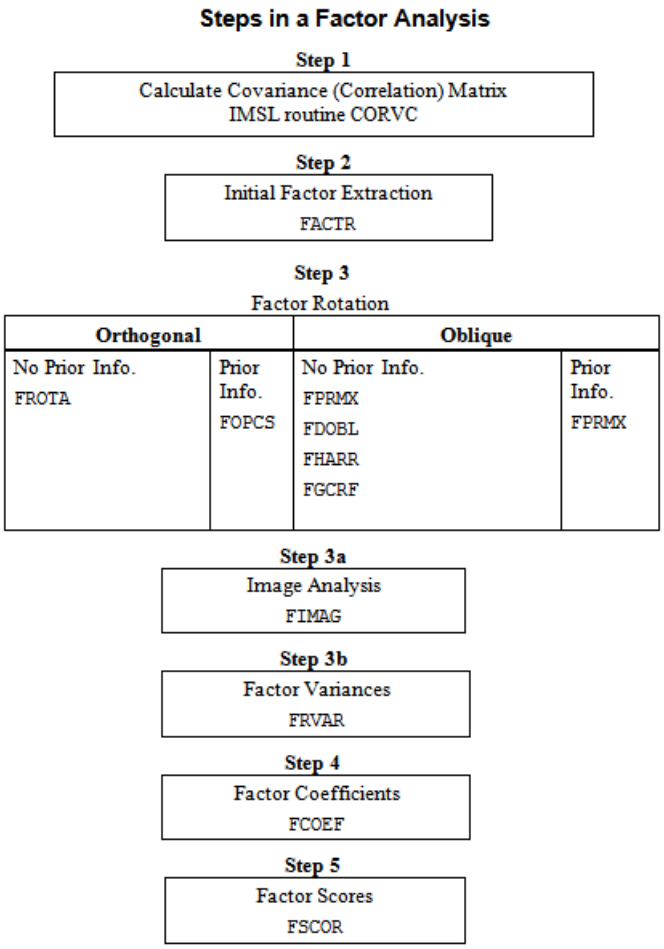
where x is the p vector of observed variables, μ is the p vector of variable means, Λ is the $p \times k$ matrix of factor loadings, f is the k vector of hypothesized underlying random factors, and e is the p vector of hypothesized unique random errors.

Because much of the computation in factor analysis was originally done by hand or was expensive on early computers, quick (but dirty) algorithms that made the calculations possible were developed. One result is the many factor extraction methods available today. Generally speaking, in the exploratory or model building phase of a factor analysis, a method of factor extraction that is not computationally intensive (such as principal components, principal factor, or image analysis) is used. If desired, a computationally intensive method is then used to obtain (what is hoped will be) the final factors.

In exploratory factor analysis, the unrotated factor loadings obtained from the factor extraction are generally transformed (rotated) to simplify the interpretation of the factors. Rotation is possible because of the overparameterization in the factor analysis model. The method used for rotation may result in factors that are independent (orthogonal rotations) or correlated (oblique rotations). Prior information may be available (or hypothesized) in which case a Procrustes rotation could be used. When no prior information is available, an analytic rotation can be performed.

Once the factor loadings have been extracted and rotated (if desired), estimates for the hypothesized underlying factors can be computed. First, one of several available methods in routine FCOEF is used to compute the factor score coefficients. Routine FSCOR is then called with these factor score coefficients to compute the factor scores.

The steps generally used in a factor analysis are summarized as follows:



Independence of Sets of Variables and Canonical Correlation Analysis

Routine MVIND computes the Wilks likelihood-ratio test of independence among several sets of variables. Routines CANCR and CANVC compute some other tests of independence between exactly two sets of variables. Routine **CANCR** uses the raw data as input while **CANVC** uses the sample variance-covariance matrix. Furthermore, **CANCR** and **CANVC** perform a canonical correlation analysis. Since **CANCR** uses a better algorithm in terms of numerical stability (it does not compute the covariance matrix), **CANCR** should be used if possible. How-

ever, if the raw data is not available, or if there is too much data for all of it to reside in memory at the same time, or if multiple canonical correlation analyses are to be performed based on the same pre-computed sample variance-covariance matrix, then the use of **CANVC** may be necessary. Canonical correlation analysis is useful for characterizing the independent linear statistical relationships that exist between the two sets of variables. This involves computing linear combinations of the variables in the two separate sets and their associated correlation. The coefficients of the variables in the linear combinations are called the "canonical coefficients," and the correlations are called "canonical correlations." Evaluation of the linear combinations using the canonical coefficients gives the "canonical scores." Routine **CANCR** computes the canonical scores for the observed data. Routine **FSCOR** can be used to compute the canonical scores for new data or for the observed data if **CANVC** is used.

PRINC

Computes principal components from a variance-covariance matrix or a correlation matrix.

Required Arguments

NDF — Number of degrees of freedom in **COV**. (Input)
If **NDF** is less than or equal to 0, 100 degrees of freedom are assumed.

COV — **NVAR** by **NVAR** matrix containing the covariance or correlation matrix. (Input)
Only the upper triangular part of **COV** is referenced.

EVAL — Vector of length **NVAR** containing the eigenvalues from matrix **COV** ordered from largest to smallest. (Output)

Optional Arguments

NVAR — Order of matrix **COV**. (Input)
Default: **NVAR** = size (**COV**,2).

LDCOV — Leading dimension of **COV** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOV** = size (**COV**,1).

ICOV — Covariance/Correlation matrix option parameter. (Input)
ICOV = 0 means that a covariance matrix is input. Otherwise, a correlation matrix is input.
Default: **ICOV** = 0.

PCT — Vector of length **NVAR** containing the cumulative percent of the total variance explained by each principal component. (Output)

STD — Vector of length **NVAR** containing the estimated asymptotic standard errors of the eigenvalues. (Output)

EVEC — **NVAR** by **NVAR** matrix containing the eigenvectors of **COV**, stored columnwise. (Output)
Each vector is normalized to have Euclidean length equal to the value one. Also, the sign of each vector is set so that the largest component in magnitude (the first of the largest if there are ties) is made positive.

LDEVEC — Leading dimension of **EVEC** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDEVEC** = size(**EVEC**, 1).

A — **NVAR** by **NVAR** matrix containing the correlations of the principal components (the columns) with the observed/standardized variables (the rows). (Output)
If **ICOV** = 0, then the correlations are with the observed variables. Otherwise, the correlations are with the standardized (to a variance of 1.0) variables. In the principal component model for factor analysis, matrix **A** is the matrix of unrotated factor loadings.

LDA — Leading dimension of **A** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDA** = size(**A**, 1).

FORTRAN 90 Interface

Generic: **CALL PRINC** (**NDF**, **COV**, **EVAL** [, ...])
Specific: The specific interface names are **S_PRINC** and **D_PRINC**.

FORTRAN 77 Interface

Single: **CALL PRINC** (**NDF**, **NVAR**, **COV**, **LDCOV**, **ICOV**, **EVAL**, **PCT**, **STD**, **EVEC**, **LDEVEC**, **A**, **LDA**)
Double: The double precision name is **DPRINC**.

Description

Routine **PRINC** finds the principal components of a set of variables from a sample covariance or correlation matrix. The characteristic roots, characteristic vectors, standard errors for the characteristic roots, and the correlations of the principal component scores with the original variables are computed. Principal components obtained from correlation matrices are the same as principal components obtained from standardized (to unit variance) variables.

The principal component scores are the elements of the vector $y = \Gamma^T x$ where Γ is the matrix whose columns are the characteristic vectors (eigenvectors) of the sample covariance (or correlation) matrix and x is the vector of observed (or standardized) random variables. The variances of the principal component scores are the characteristic roots (eigenvalues) of the the covariance (correlation) matrix.

Asymptotic variances for the characteristic roots were first obtained by Girshick (1939) and are given more recently by Kendall, Stuart, and Ord (1983, page 331). These variances are computed either for covariance matrices (**ICOV** = 0) or for correlation matrices (**ICOV** ≠ 0).

The correlations of the principal components with the observed (or standardized) variables are given in the matrix A. When the principal components are obtained from a correlation matrix, A is the same as the matrix of unrotated factor loadings obtained for the principal components model for factor analysis.

Comments

Informational Errors

Type	Code	Description
3	1	Because NDF is zero or less, 100 degrees of freedom will be used.
3	2	One or more eigenvalues much less than zero are computed. The matrix COV is not nonnegative definite. In order to continue computations of STD and A, these eigenvalues are treated as zero.

Example

Principal components are computed for a nine-variable matrix.

```
      USE PRINC_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    ICOV, LDA, LDCOV, LDEVEC, NDF, NVAR
      PARAMETER  (ICOV=1, LDA=9, LDCOV=9, LDEVEC=9, NDF=100, NVAR=9)
!
      REAL       A(LDA,NVAR), COV(LDCOV,NVAR), EVAL(NVAR), &
           EVEC(LDEVEC,NVAR), PCT(NVAR), STD(NVAR)
!
      DATA COV/&
      1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, 0.639, &
      0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, &
      0.283, 0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, &
      0.452, 0.219, 0.504, 0.471, 0.506, 0.355, 1.000, 0.691, &
      0.791, 0.443, 0.285, 0.505, 0.346, 0.418, 0.270, 0.691, &
      1.000, 0.679, 0.383, 0.149, 0.409, 0.426, 0.462, 0.254, &
      0.791, 0.679, 1.000, 0.372, 0.314, 0.472, 0.576, 0.547, &
      0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, 0.434, &
      0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
      0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
      1.000/
!
      CALL PRINC (NDF, COV, EVAL, ICOV=ICOV, PCT=PCT, STD=STD, &
                  EVEC=EVEC, A=A)
!
      CALL WRRRN ('EVAL', EVAL, 1, NVAR, 1)
      CALL WRRRN ('PCT', PCT, 1, NVAR, 1)
      CALL WRRRN ('STD', STD, 1, NVAR, 1)
      CALL WRRRN ('EVEC', EVEC)
      CALL WRRRN ('A', A)
      END
```

Output

```
                                EVAL
     1       2       3       4       5       6       7       8       9
 4.677   1.264   0.844   0.555   0.447   0.429   0.310   0.277   0.196

                                PCT
     1       2       3       4       5       6       7       8       9
 0.520   0.660   0.754   0.816   0.865   0.913   0.947   0.978   1.000

                                STD
      1       2       3       4       5       6       7       8
 0.6498  0.1771  0.0986  0.0879  0.0882  0.0890  0.0944  0.0994

      9
 0.1113

                                EVEC
          1       2       3       4       5       6       7       8
1   0.3462  -0.2354   0.1386  -0.3317  -0.1088   0.7974   0.1735  -0.1240
2   0.3526  -0.1108  -0.2795  -0.2161   0.7664  -0.2002   0.1386  -0.3032
3   0.2754  -0.2697  -0.5585   0.6939  -0.1531   0.1511   0.0099  -0.0406
4   0.3664   0.4031   0.0406   0.1196   0.0017   0.1152  -0.4022  -0.1178
5   0.3144   0.5022  -0.0733  -0.0207  -0.2804  -0.1796   0.7295   0.0075
6   0.3455   0.4553   0.1825   0.1114   0.1202   0.0696  -0.3742   0.0925
7   0.3487  -0.2714  -0.0725  -0.3545  -0.5242  -0.4355  -0.2854  -0.3408
8   0.2407  -0.3159   0.7383   0.4329   0.0861  -0.1969   0.1862  -0.1623
9   0.3847  -0.2533  -0.0078  -0.1468   0.0459  -0.1498  -0.0251   0.8521

          9
1  -0.0488
2  -0.0079
3  -0.0997
4   0.7060
5   0.0046
6  -0.6780
7  -0.1089
8   0.0505
9   0.1225

                                 A
          1       2       3       4       5       6       7       8
1   0.7487  -0.2646   0.1274  -0.2471  -0.0728   0.5224   0.0966  -0.0652
2   0.7625  -0.1245  -0.2568  -0.1610   0.5124  -0.1312   0.0772  -0.1596
3   0.5956  -0.3032  -0.5133   0.5170  -0.1024   0.0990   0.0055  -0.0214
4   0.7923   0.4532   0.0373   0.0891   0.0012   0.0755  -0.2240  -0.0620
5   0.6799   0.5646  -0.0674  -0.0154  -0.1875  -0.1177   0.4063   0.0039
6   0.7472   0.5119   0.1677   0.0830   0.0804   0.0456  -0.2084   0.0487
7   0.7542  -0.3051  -0.0666  -0.2641  -0.3505  -0.2853  -0.1589  -0.1794
8   0.5206  -0.3552   0.6784   0.3225   0.0576  -0.1290   0.1037  -0.0854
9   0.8319  -0.2848  -0.0072  -0.1094   0.0307  -0.0981  -0.0140   0.4485

          9
1  -0.0216
2  -0.0035
3  -0.0442
4   0.3127
5   0.0021
6  -0.3003
```

```
    7  -0.0482
    8   0.0224
    9   0.0543
```

KPRIN

more...

Maximums likelihood or least-squares estimates for principal components from one or more matrices.

Required Arguments

COV — NVAR by NVAR by NMAT array containing the NMAT covariance or correlation matrices. (Input)
Only the upper triangular elements of each matrix are referenced.

ANI — Vector of length NMAT containing the number of observations in each of the covariance matrices. (Input)
For least-squares estimation, the square root of ANI(I) is the weight to be used for the I-th covariance matrix. Since the elements of ANI are used as weights, they need not be integers.

EVEC — NVAR by NVAR matrix containing the estimated principal components. (Output)
Each column of EVEC contains a principal component vector (an "eigenvector"). The ordering of the eigenvectors is such that the sum of the corresponding eigenvalues are ordered from largest to smallest. Each vector is normalized to have Euclidean length equal to the value one.

Optional Arguments

NVAR — Number of variables in each matrix. (Input)
NVAR must be 2 or greater.
Default: NVAR = size (COV,2).

NMAT — Number of matrices. (Input)
Default: NMAT = size (COV,3).

LDCOV — Leading and second dimensions of COV exactly as specified in the dimension statement of the calling program. (Input)
The first two dimensions of COV must be equal.
Default: LDCOV = size (COV,1).

IMETH — Method to be used for extracting the estimated principal components. (Input)
For **IMETH** = 0, maximum likelihood estimation is used. For **IMETH** = 1, least-squares estimation is used.
Default: **IMETH** = 0.

LDEVEC — Leading dimension of **EVEC** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDEVEC** = size (**EVEC**,1).

FORTRAN 90 Interface

Generic: **CALL KPRIN** (**COV**, **ANI**, **EVEC** [, ...])
Specific: The specific interface names are **S_KPRIN** and **D_KPRIN**.

FORTRAN 77 Interface

Single: **CALL KPRIN** (**NVAR**, **NMAT**, **COV**, **LDCOV**, **ANI**, **IMETH**, **EVEC**, **LDEVEC**)
Double: The double precision name is **DKPRIN**.

Description

Routine **KPRIN** is the IMSL version of the F-G diagonalization routine of Flury and Constantine (1985) with modifications as discussed by Clarkson (1988a, 1988b). Let k = **NMAT**. Routine **KPRIN** computes the common principal components of $k \geq 1$ covariance (or correlation) matrices using either a least-squares or a maximum likelihood criterion. Computing common principal components is equivalent to finding the "eigenvectors" that best simultaneously diagonalize k symmetric matrices. (Note that when k = 1, both least-squares and maximum likelihood estimation yield the eigenvectors of the input matrix.) See Flury (1988) for applications of common principal components.

The algorithm proceeds by accumulating simple rotations as follows: Initial estimates of the diagonalizing principal components are found as the eigenvectors of the summed covariance matrices (unless **K3RIN** is used, see Comment 3). The covariance matrices are then pre- and post-multiplied by the initial estimates to obtain approximately diagonal matrices. Let

$$w_l^{ij}$$

denote the l-th 2×2 matrix obtained from the (i, j), (i, i), and (j, j) elements of S_l, where S_l is the l-th covariance matrix in **COV**. Then, for each i and j, a Jacobi rotation is found and applied such that the least-squares or maximum likelihood criterion is optimized over all k matrices in

$$w_l^{ij}$$

An iteration consists of computing and applying a Jacobi rotation for all $p(p - 1)/2$ possible off-diagonal elements (i, j) where p = **NVAR**. A maximum of 50 iterations are allowed before convergence. Convergence is assumed when the maximum change in the any element in the eigenvectors from one iteration to the next is less than 0.0001.

Let **Γ** denote the current estimates of the optimizing principal components. Then, maximizing the multivariate normal likelihood is equivalent to minimizing the criterion

$$L = \prod_{i=1}^{k} \left(\frac{\det \hat{\Sigma}_i}{\det S_i} \right)^{n_i}$$

where S_i is the i-th covariance matrix, n_i is its degrees of freedom, and

$$\hat{\Sigma}_i$$

is the estimate of the covariance matrix under the common principal components model.

During each Jacobi iteration, an optimal orthogonal matrix T_{ij} is found that rotates the two vectors in columns i and j of **Γ**. When restricted to T_{ij}, the criterion above becomes

$$L = \prod_{i=1}^{k} \left(\frac{\det(\mathrm{diag}(T_{ij}^T \Gamma^T S_i \Gamma) T_{ij})}{\det(S_i)} \right)^{n_i}$$

Γ is updated as **Γ**T_{ij}. When convergence has been reached (the maximum change in **Γ** is less than 0.0001), **Γ** contains the optimizing principal components. Initially, **Γ** is taken as the eigenvectors of the matrix $\mathbf{\Sigma}_i S_i$.

In least-squares estimation, the matrices T_{ij} are found such that the sum of the squared off-diagonal elements in the resulting "diagonalized" matrices are minimized. That is, T_{ij} is found to minimize

$$v_{ij}^T v_{ij}$$

where v_{ij} is the vector of length k containing the off-diagonal elements in the matrices

$$w_l^{ij}$$

See Flury and Gautschi (1986) for further details on the general algorithm, especially in maximum likelihood estimation. See Clarkson (1988b) for details of the least-squares algorithm.

If the "residual" matrices $\Gamma^T S_i \Gamma$ are desired, they may be obtained in the work vector **H** returned from **K2RIN** or from the matrix **COV** returned from **K3RIN**. If the least-squares criterion is needed, it is easily computed as the sum of the squared off-diagonal elements in **H** (or **COV**). To compute the likelihood ratio criterion, the eigenvalues of each matrix in **COV** first need to be computed. Denote the eigenvalues from the *l*-th matrix by λ_{lj}, and let

$$\hat{\lambda}_{lj}$$

be the eigenvalues obtained under the common principal component model (and returned as the diagonal elements of **H** or, from **K3RIN**, **COV**). Then, the log-likelihood-ratio statistic for testing,

$$H_o: \Gamma^T \Sigma_l \Gamma$$

is diagonal, $l = 1, \ldots, k$, is computed as:

$$\ell = \sum_{l=1}^{k} \sum_{j=1}^{p} n_l \log \frac{\hat{\lambda}_{lj}}{\lambda_{lj}}$$

The distribution of ℓ under H_0 is asymptotically X^2 with $(k - 1)p(p - 1)/2$ degrees of freedom (see Flury 1984).

Comments

1. Workspace may be explicitly provided, if desired, by use of **K2RIN/DK2RIN**. The reference is:

   ```
   CALL K2RIN (NVAR, NMAT, COV, LDCOV, ANI, IMETH, EVEC, LDEVEC, H, AUX, BOLD, G, T)
   ```

   The additional arguments are as follows:

   H — Work vector of length **NVAR**^{2} * **NMAT**. On return from K2RIN, **H** may be treated as an array dimensioned as **H**(**NVAR**, **NVAR**, **NMAT**). Each **NVAR** by **NVAR** matrix in **H** is computed as (**EVEC**)**T** * **COV**(**I**) * **EVEC**, i.e., **H** contains the "eigenvalues" and the "residuals" for each covariance matrix. Here, **COV**(**I**) is the **I**-th covariance matrix.

   AUX — Work vector of length max(3 * **NMAT**, **NVAR**^{2}).

   BOLD — Work vector of length **NVAR**^{2}.

   G — Work vector of length 2 * **NVAR**.

   T — Work vector of length 4 * **NMAT**.

2. Informational errors

Type	Code	Description
3	1	Convergence did not occur within 50 iterations. Convergence is assumed.
4	2	An input matrix is singular. Singular input matrices are not allowed in maximum likelihood estimation.

3. If user specified initial estimates for EVEC are desired (and argument error checking is not needed), then the routine K3RIN (DK3RIN) may be used. The calling sequence is

   CALL K3RIN (NVAR, NMAT, COV, LDCOV, ANI, IMETH, EVEC, LDEVEC, AUX, BOLD, G, T)

   On input, EVEC contains the initial estimates of the common principal components (EVEC must be an orthogonal matrix). On output, COV contains the NMAT matrices (EVEC)T * COV (I) * EVEC. The user should be wary of stationary points in the likelihood if K3RIN is used.

Example

The following example is taken from Flury and Constantine (1985). It involves two 4 by 4 covariance matrices. The two covariance matrices are given as:

$$\begin{pmatrix} 1.3 & -0.3 & -0.6 & 0 \\ -0.3 & 2.1 & 0 & -0.6 \\ -0.6 & 0 & 2.9 & -0.3 \\ 0 & -0.6 & -0.3 & 3.7 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

```
      USE KPRIN_INT
      USE WRRRN_INT

      IMPLICIT   NONE
!                                SPECIFICATIONS FOR PARAMETERS

      INTEGER    LDCOV, LDEVEC, NGROUP, NVAR
      PARAMETER  (LDCOV=4, LDEVEC=4, NGROUP=2, NVAR=4)
!
      REAL       ANI(NGROUP), COV(LDCOV,LDCOV,NGROUP), EVEC(LDEVEC,NVAR)
!
      DATA COV/1.3, -0.3, -0.6, 0, -0.3, 2.1, 0, -0.6, -0.6, 0, 2.9, &
      -0.3, 0, -0.6, -0.3, 3.7, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 3, &
      0, 0, 0, 0, 4/
!
      ANI(1) = 1
      ANI(2) = 1
```

```
!
      CALL KPRIN (COV, ANI, EVEC)
!
      CALL WRRRN ('EVEC', EVEC)
!
      END
```

Output

```
                  EVEC
          1       2       3       4
1    0.9743  -0.1581  -0.1581   0.0257
2    0.1581   0.9743  -0.0257  -0.1581
3    0.1581  -0.0257   0.9743  -0.1581
4    0.0257   0.1581   0.1581   0.9743
```

FACTR

Extracts initial factor loading estimates in factor analysis.

Required Arguments

COV — NVAR by NVAR matrix containing the variance-covariance or correlation matrix. (Input)

NF — Number of factors in the model. (Input)

UNIQ — Vector of length NVAR containing the unique variances. (Input/Output, if INIT = 1; output, otherwise)
If INIT = 1, UNIQ contains the initial estimates of these variances on input. On output, UNIQ contains the estimated unique variances. For IMTH = 0, the unique variances are assumed to be known and are not changed from the input values when INIT = 1.

A — NVAR by NF matrix of unrotated factor loadings. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

IMTH — Method used to obtain the estimates. (Input)
Default: IMTH = 0.

IMTH	Method
0	Principal component (principal component model) or principal factor (common factor model). If INIT = 1 and UNIQ contains zeros, then this option results in the principal component method. Otherwise, the principal factor method is used.
1	Unweighted least squares (common factor model).
2	Generalized least squares (common factor model).
3	Maximum likelihood (common factor model).
4	Image factor analysis (common factor model).
5	Alpha factor analysis (common factor model).

NDF — Number of degrees of freedom in COV. (Input)
NDF is not required when IMTH = 0, 1, or 4. NDF defaults to 100 if NDF = 0.
Default: NDF = 0.

INIT — Method used to obtain initial estimates of the unique variances. (Input)
Default: INIT = 0.

INIT	Method
0	Initial estimates are taken as the constant 1 - NF/(2 * NVAR) divided by the diagonal elements of the inverse of COV.
1	Initial estimates are input in vector UNIQ.

MAXIT — Maximum number of iterations in the iterative procedure. (Input)
Typical for methods 1 to 3 is 30, while 60 is typical for method 5. MAXIT is not referenced when IMTH = 0 or 4.
Default: MAXIT = 30.

MAXSTP — Maximum number of step halvings allowed during any one iteration. (Input)
Typical is 8. MAXSTP is not referenced when IMTH = 0, 4, or 5.
Default: MAXSTP = 8.

EPS — Convergence criterion used to terminate the iterations. (Input)
For methods 1 to 3, convergence is assumed when the relative change in the criterion is less than EPS. For method 5, convergence is assumed when the maximum change (relative to the variance) of a uniqueness is less than EPS. EPS is not referenced when IMTH = 0 or 4. EPS = 0.0001 is typical.
Default: EPS = 0.0001.

EPSE — Convergence criterion used to switch to exact second derivatives. (Input)
When the largest relative change in the unique standard deviation vector is less than EPSE, exact second derivative vectors are used. Typical is 0.1. EPSE is not referenced when IMTH = 0, 4, or 5.
Default: EPSE = 0.1.

IPRINT — Printing option. (Input)
If IPRINT = 0, then no printing is performed. If IPRINT = 1, then printing of the final results is performed. If IPRINT = 2, then printing of an iteration summary and the final results is performed. Default: IPRINT = 0.

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

EVAL — Vector of length NVAR containing the eigenvalues of the matrix from which the factors were extracted. (Output)
If IMTH = 5, then the first NF positions of EVAL contain the ALPHA coefficients. Note that EVAL does not usually contain eigenvalues for matrix COV.

STAT — Vector of length 6 containing some output statistics. (Output)

I	STAT(I)
1	Value of the function minimum.
2	Tucker reliability coefficient.
3	Chi-squared test statistic for testing that NF common factors are adequate for the data.
4	Degrees of freedom in chi-squared. This is computed as $((NVAR - NF)^2 - NVAR - NF)/2$.
5	Probability of a greater chi-squared statistic.
6	Number of iterations.

STAT is not used when IMTH = 0, 4, or 5.

DER — Vector of length NVAR containing the parameter updates when convergence was reached (or the iterations terminated). (Output)

FORTRAN 90 Interface

Generic: CALL FACTR (COV, NF, UNIQ, A [, ...])
Specific: The specific interface names are S_FACTR and D_FACTR.

FORTRAN 77 Interface

Single: CALL FACTR (NVAR, COV, LDCOV, NF, IMTH, NDF, INIT, MAXIT, MAXSTP, EPS, EPSE, IPRINT, UNIQ, A, LDA, EVAL, STAT, DER)

Double: The double precision name is DFACTR.

Description

Routine FACTR computes unrotated factor loadings in exploratory factor analysis models. Models available in FACTR are the principal component model for factor analysis and the common factor model with additions to the common factor model in alpha factor analysis and image analysis. Methods of estimation include principal components, principal factor, image analysis, unweighted least squares, generalized least squares, and maximum likelihood.

In the factor analysis model used for factor extraction, the basic model is given as $\Sigma = \Lambda\Lambda^T + \Psi$ where Σ is the $p \times p$ population covariance matrix, Λ is the $p \times k$ matrix of factor loadings relating the factors f to the observed variables x, and Ψ is the $p \times p$ matrix of covariances of the unique errors e. Here, p = NVAR and k = NF. The relationship between the factors, the unique errors, and the observed variables is given as $x = \Lambda f + e$ where, in addition, it is assumed that the expected values of e, f, and x are zero. (The sample means can be subtracted from x if the expected value of x is not zero.) It is also assumed that each factor has unit variance, the factors are independent of each other, and that the factors and the unique errors are mutually independent. In the common factor model, the elements of the vector of unique errors e are also assumed to be independent of one another so that the matrix Ψ is diagonal. This is not the case in the principal component model in which the errors may be correlated.

Further differences between the various methods concern the criterion that is optimized and the amount of computer effort required to obtain estimates. Generally speaking, the least-squares and maximum likelihood methods, which use iterative algorithms, require the most computer time with the principal factor, principal component and the image methods requiring much less time since the algorithms in these methods are not iterative. The algorithm in alpha factor analysis is also iterative, but the estimates in this method generally require somewhat less computer effort than the least-squares and maximum likelihood estimates. In all algorithms, one eigensystem analysis is required on each iteration.

The Principal Component and Principal Factor Methods

Both the principal component and the principal factor methods compute the factor loading estimates as

$$\hat{\Gamma}\hat{\Delta}^{-1/2}$$

where **Γ** and the diagonal matrix **Δ** are the eigenvectors and eigenvalues of a matrix. In the principal component model, the eigensystem analysis is performed on the sample covariance (correlation) matrix *S* while in the principal factor model the matrix (*S* **-** **Ψ**) is used. If the unique error variances **Ψ** are not known (i.e., if **INIT** = 0) in the principal factor model, then **FACTR** obtains estimates for them as discussed in Comment 3. If the principal components model is to be used, then the **INIT** = 1 option should be set, and the vector **UNIQ** should be set so that all elements are zero. If **UNIQ** is not set, principal factor model estimates are computed.

The basic idea in the principal component method is to find factors that maximize the variance in the original data that is explained by the factors. Because this method allows the unique errors to be correlated, some factor analysts insist that the principal component method is not a factor analytic method. Usually however, the estimates obtained via the principal component model and other models in factor analysis will be quite similar.

It should be noted that both the principal component and the principal factor methods give different results when the correlation matrix is used in place of the covariance matrix. Indeed, any rescaling of the sample covariance matrix can lead to different estimates with either of these methods. A further difficulty with the principal factor method is the problem of estimating the unique error variances. Theoretically, these must be known in advance and passed to **FACTR** through **UNIQ**. In practice, the estimates of these parameters produced via the **INIT** = 0 option in **FACTR** are often used. In either case, the resulting adjusted covariance (correlation) matrix

$$\left(S - \hat{\Psi}\right)$$

may not yield the **NF** positive eigenvalues required for **NF** factors to be obtained. If this occurs, the user must either lower the number of factors to be estimated or give new unique error variance values.

The Least-Squares and Maximum Likelihood Methods

Unlike the previous two methods, the algorithm used to compute estimates in this section is iterative (see Joreskog 1977). As with the principal factor model, the user may either initialize **UNIQ** or allow **FACTR** to compute initial estimates for the unique error variances. Unlike the principal factor method, **FACTR** then optimizes the criterion function with respect to both **Ψ** and **Γ**. (In the principal factor method, **Ψ** is assumed to be known. Given **Ψ**, estimates for **Λ** may be obtained.)

The major differences between the methods discussed in this section are in the criterion function that is optimized. Let *S* denote the sample covariance (correlation) matrix, and let **Σ** denote the covariance matrix that is to be estimated by the factor model. In the unweighted least-squares method, also called the iterated principal factor method or the minres method (see Harman 1976, page 177), the function minimized is the sum of the squared differences between *S* and **Σ**. This is written as ϕ_{ul} = .5 trace((*S* **-** **Σ**)^2).

Generalized least-squares and maximum likelihood estimates are asymptotically equivalent methods. Maximum likelihood estimates maximize the (normal theory) likelihood $\{\phi_{ml} = \text{trace}(\Sigma^{-1}S) - \log(|\Sigma^{-1}S|) - p\}$ while generalized least squares optimizes the function $\phi_{gs} = \text{trace}((\Sigma S^{-1} - I)^2)$.

In all three methods, a two-stage optimization procedure is used. This proceeds by first solving the likelihood equations for Λ in terms of Ψ and substituting the solution into the likelihood. This gives a criterion $\phi(\Psi,\Lambda(\Psi))$, which is optimized with respect to Ψ. In the second stage, the estimates

$$\hat{\Lambda}$$

are obtained from the estimates for Ψ.

The generalized least-squares and the maximum likelihood methods allow for the computation of a statistic (STAT(3)) for testing that NF common factors are adequate to fit the model. This is a chi-squared test that all remaining parameters associated with additional factors are zero. If the probability of a larger chi-squared is small (see STAT(5)) so that the null hypothesis is rejected, then additional factors are needed (although these factors may not be of any practical importance). Failure to reject does not legitimize the model. The statistic STAT(3) is a likelihood ratio statistic in maximum likelihood estimation. As such, it asymptotically follows a chi-squared distribution with degrees of freedom given in STAT(4).

The Tucker and Lewis (1973) reliability coefficient, ρ, is returned in STAT(2) when the maximum likelihood or generalized least-squares methods are used. This coefficient is an estimate of the ratio of explained to the total variation in the data. It is computed as follows:

$$\rho = \frac{mM_o - mM_k}{mM_o - 1}$$

$$m = d - \frac{2p+5}{6} - \frac{2k}{6}$$

$$M_o = \frac{-\ln\left(|S|\right)}{p\left(p-1\right)/2}$$

$$M_k = \frac{\phi}{\left(\left(p-k\right)^2 - p - k\right)/2}$$

where $|S|$ is determinant of COV, p = NVAR, k = NF, ϕ is the optimized criterion, and d = NDF.

Image Analysis

The term “image analysis” is used here to denote the noniterative image method of Kaiser (1963). It is not the image factor analysis discussed by Harman (1976, page 226). The image method (as well as the alpha factor analysis method) begins with the notion that only a finite number from an infinite number of possible variables have been measured. The image factor pattern is calculated under the assumption that the ratio of the number of fac-

tors to the number of observed variables is near zero so that a very good estimate for the unique error variances (for standardized variables) is given as one minus the squared multiple correlation of the variable under consideration with all variables in the covariance matrix.

First, the matrix $D^2 = (\text{diag}(S^{-1}))^{-1}$ is computed where the operator "diag" results in a matrix consisting of the diagonal elements of its argument, and S is the sample covariance (correlation) matrix. Then, the eigenvalues Λ and eigenvectors Γ of the matrix $D^{-1}S\,D^{-1}$ are computed. Finally, the unrotated image factor pattern matrix is computed as $A = D\Gamma[(\Lambda - I)^2\Lambda^{-1}]^{1/2}$.

Alpha Factor Analysis

The alpha factor analysis method of Kaiser and Caffrey (1965) finds factor-loading estimates to maximize the correlation between the factors and the complete universe of variables of interest. The basic idea in this method is as follows: only a finite number of variables out of a much larger set of possible variables is observed. The population factors are linearly related to this larger set while the observed factors are linearly related to the observed variables. Let f denote the factors obtainable from a finite set of observed random variables, and let ξ denote the factors obtainable from the universe of observable variables. Then, the alpha method attempts to find factor-loading estimates so as to maximize the correlation between f and ξ. In order to obtain these estimates, the iterative algorithm of Kaiser and Caffrey (1965) is used.

Comments

1. **FACTR** makes no attempt to solve for **NF**, the number of factors. In general, if **NF** is not known in advance, several different values of **NF** should be used, and the most reasonable value kept in the final solution.

2. The iterative methods are generally thought to be superior from a theoretical point of view but, in practice, often lead to solutions which differ little from the noniterative methods. For this reason, it is usually suggested that a non-iterative method be used in the initial stages of the factor analysis, and that the iterative methods be used when issues such as the number of factors have been resolved.

3. Initial estimates for the unique variances are input when **INIT** = 1. If the iterative methods fail for these values, new initial estimates should be tried. These may be obtained by use of another factoring method (use the final estimates from the new method as initial estimates in the old method).

   Another alternative is to let **FACTR** compute initial estimates of the unique error variances. When **INIT** = 0, the initial estimates are taken as a constant

$$\left(1-\frac{k}{2p}\right)$$

divided by the diagonal elements of the

$$\hat{\sum}^{-1}$$

matrix. When the correlation matrix is factor analyzed, this is a constant times one minus the squared multiple correlation coefficient.

4. Workspace may be explicitly provided, if desired, by use of F2CTR/DF2CTR. The reference is:

   ```
   CALL F2CTR (NVAR, COV, LDCOV, NF, IMTH, NDF, INIT, MAXIT, MAXSTP, EPS, EPSE,
       IPRINT, UNIQ, A, LDA, EVAL, STAT, DER, IS, COVI, WK, OLD, EVEC, HESS)
   ```

   The additional arguments are as follows:

   IS — Integer work vector of length equal to NVAR.

   COVI — Real work vector of length equal to $NVAR^2$.

   WK — Real work vector of length equal to NVAR.

   OLD — Real work vector of length equal to NVAR.

   EVEC — Real work vector of length equal to $NVAR^2$.

   HESS — Real work vector of length equal to $NVAR^2$.

5. Informational errors

Type	Code	Description
3	1	Too many iterations. Convergence is assumed.
3	2	Too many step halvings. Convergence is assumed.
3	4	There are no degrees of freedom for the significance testing.

Example

The following data were originally analyzed by Emmett (1949). There are 211 observations on 9 variables. Following Lawley and Maxwell (1971), three factors will be obtained by the method of maximum likelihood.

```
          USE FACTR_INT

          IMPLICIT   NONE
          INTEGER    IMTH,IPRINT, LDA, LDCOV, MAXSTP, NDF, NF, NVAR
          REAL       EPS, EPSE
          PARAMETER  (EPS=0.000001, EPSE=0.01, IMTH=3, IPRINT=1, &
                      LDA=9, LDCOV=9, MAXSTP=10, NDF=210, NF=3, NVAR=9)
    !
          REAL       A(LDA,NF), COV(LDCOV,NVAR), DER(NVAR), EVAL(NVAR), &
```

```
            STAT(6), UNIQ(NVAR)
!
      DATA COV/ &
       1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, 0.639, &
       0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, 0.283, 0.645, &
       0.395, 0.479, 1.000, 0.355, 0.270, 0.254, 0.452, 0.219, 0.504, &
       0.471, 0.506, 0.355, 1.000, 0.691, 0.791, 0.443, 0.285, 0.505, &
       0.346, 0.418, 0.270, 0.691, 1.000, 0.679, 0.383, 0.149, 0.409, &
       0.426, 0.462, 0.254, 0.791, 0.679, 1.000, 0.372, 0.314, 0.472, &
       0.576, 0.547, 0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, &
       0.434, 0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
       0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, 1.000/
!
      CALL FACTR (COV, NF, UNIQ, A, IMTH=IMTH, MAXSTP=MAXSTP, EPS=EPS,&
                  EPSE=EPSE, IPRINT=IPRINT, NDF=NDF, EVAL=EVAL, &
                  STAT=STAT, DER=DER)
      END
```

Output

```
                        Unique Error Variances
     1        2        3        4        5        6        7        8
0.4505   0.4271   0.6166   0.2123   0.3805   0.1769   0.3995   0.4615

     9
0.2309

     Unrotated Loadings
          1        2        3
1   0.6642  -0.3209   0.0735
2   0.6888  -0.2471  -0.1933
3   0.4926  -0.3022  -0.2224
4   0.8372   0.2924  -0.0354
5   0.7050   0.3148  -0.1528
6   0.8187   0.3767   0.1045
7   0.6615  -0.3960  -0.0777
8   0.4579  -0.2955   0.4913
9   0.7657  -0.4274  -0.0117

                           Eigenvalues
    1       2       3       4       5       6       7       8       9
0.063   0.229   0.541   0.865   0.894   0.974   1.080   1.117   1.140

                           STAT
     1            2            3            4            5
0.0350       1.0000       7.1494      12.0000       0.8476

     6
5.0000

                      Final Parameter Updates
          1            2            3            4            5
2.02042E-07  2.95010E-07  1.80908E-07  6.38808E-08  2.00809E-07

          6            7            8            9
1.48762E-07  1.73797E-08  3.95484E-07  1.42415E-07
```

FROTA

Computes an orthogonal rotation of a factor loading matrix using a generalized orthomax criterion, including quartimax, varimax, and equamax rotations.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Nonnegative constant used to define the rotation. (Input)
W = 0.0 results in quartimax rotations, W = 1.0 results in varimax rotations, and W = NF/2.0 results in equamax rotations. Other nonnegative values of W may also be used, but the best values for W are in the range (0.0, 5 * NF).

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B may share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT ≤ 30 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than **EPS** from one iteration to the next, convergence is assumed. **EPS** = 0.0001 is typical. **EPS** ≤ 0.0 defaults to 0.0001.
Default: **EPS** = 0.0001.

LDB — Leading dimension of **B** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDB** = size (**B**,1).

LDT — Leading dimension of **T** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDT** = size (**T**,1).

FORTRAN 90 Interface

Generic: `CALL FROTA (A, W, B, T [, ...])`
Specific: The specific interface names are **S_FROTA** and **D_FROTA**.

FORTRAN 77 Interface

Single: `CALL FROTA (NVAR, NF, A, LDA, NRM, MAXIT, W, EPS, B, LDB, T, LDT)`
Double: The double precision name is **DFROTA**.

Description

Routine **FROTA** performs an orthogonal rotation according to an orthomax criterion. In this analytic method of rotation, the criterion function

$$Q = \sum_i \sum_r \lambda_{ir}^4 - \frac{\gamma}{p} \sum_r \left[\sum_i \lambda_{ir}^2 \right]^2$$

is minimized by finding an orthogonal rotation matrix T such that $(\lambda_{ij}) = \Lambda = AT$ where A is the matrix of unrotated factor loadings. Here, $\gamma \geq 0$ is a user-specified constant (W) yielding a family of rotations, and p is the number of variables.

Kaiser (row) normalization can be performed on the factor loadings prior to rotation via the option parameter **NRM**. In Kaiser normalization, the rows of **A** are first "normalized" by dividing each row by the square root of the sum of its squared elements (Harman 1976). After the rotation is complete, each row of **B** is "denormalized" by multiplication by its initial normalizing constant.

The method for optimizing *Q* proceeds by accumulating simple rotations where a simple rotation is defined to be one in which *Q* is optimized for two columns in **Λ** and for which the requirement that *T* be orthogonal is satisfied. A single iteration is defined to be such that each of the NF(NF - 1)/2 possible simple rotations is performed where NF is the number of factors. When the relative change in *Q* from one iteration to the next is less than EPS (the user-specified convergence criterion), the algorithm stops. EPS = 0.0001 is usually sufficient. Alternatively, the algorithm stops when the user-specified maximum number of iterations, MAXIT, is reached. MAXIT = 30 is usually sufficient.

The parameter in the rotation, **γ**, is used to provide a family of rotations. When **γ** = 0.0, a direct quartimax rotation results. Other values of **γ** yield other rotations.

Comments

Workspace may be explicitly provided, if desired, by use of F2OTA/DF2OTA. The reference is

```
CALL F2OTA (NVAR, NF, A, LDA, NRM, MAXIT, W, EPS, B, LDB, T, LDT, WORK)
```

The additional argument is:

WORK — Real work vector of length equal to NVAR.

Example

The example is taken from Emmett (1949) and involves factors derived from nine variables. In this example, the varimax method is chosen with row normalization by using W = 1.0 and NRM = 1, respectively. The results correspond to those given by Lawley and Maxwell (1971, page 84).

```
      USE FROTA_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDB, LDT, NF, NVAR
      REAL       W
      PARAMETER  (LDA=9, LDB=9, LDT=3, NF=3, NVAR=9, W=1.0)
!
      REAL       A(LDA,NF), B(LDB,NF), T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, -.2955, &
      -.4274, .0735, -.1933, -.2224, -.0354, -.1528, .1045, -.0778, &
      .4914, -.0117/
!
      CALL FROTA (A, W, B, T)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      END
```

Output

```
                B
            1         2         3
1      0.2638   -0.5734    0.3888
2      0.3423   -0.6610    0.1370
3      0.1625   -0.5943    0.0622
4      0.8124   -0.3197    0.1594
5      0.7356   -0.2800    0.0036
6      0.8510   -0.1890    0.2513
7      0.2164   -0.6906    0.2768
8      0.1144   -0.2431    0.6828
9      0.2687   -0.7431    0.3804

                T
            1         2         3
1      0.7307   -0.5939    0.3367
2      0.6816    0.6623   -0.3112
3     -0.0382    0.4569    0.8887
```

FOPCS

more...

Computes an orthogonal Procrustes rotation of a factor-loading matrix using a target matrix.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

X — NVAR by NF target matrix of the rotation. (Input)

B — NVAR by NF matrix of rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of **T** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDT** = size (**T**,1).

FORTRAN 90 Interface

Generic: **CALL FOPCS** (**A**, **X**, **B**, **T** [, ...])
Specific: The specific interface names are **S_FOPCS** and **D_FOPCS**.

FORTRAN 77 Interface

Single: **CALL FOPCS** (**NVAR**, **NF**, **A**, **LDA**, **X**, **LDX**, **B**, **LDB**, **T**, **LDT**)
Double: The double precision name is **DFOPCS**.

Description

Routine **FOPCS** performs orthogonal Procrustes rotation according to a method proposed by Schöneman (1966). Let k = **NF** denote the number of factors, p = **NVAR** denote the number of variables, A denote the $p \times k$ matrix of unrotated factor loadings, T denote the $k \times k$ orthogonal rotation matrix (orthogonality requires that $T^T T$ be a $k \times k$ identity matrix), and let X denote the target matrix. The basic idea in orthogonal Procrustes rotation is to find an orthogonal rotation matrix T such that $B = AT$ and T provides a least-squares fit between the target matrix X and the rotated loading matrix B. Schöneman's algorithm proceeds by finding the singular value decomposition of the matrix $A^T X = U\Sigma V^T$. The rotation matrix is computed as $T = UV^T$.

Comments

1. Workspace may be explicitly provided, if desired, by use of **F2PCS/DF2PCS**. The reference is:

   CALL F2PCS (NVAR, NF, A, LDA, X, LDX, B, LDB, T, LDT, WK, S)

   The additional arguments are as follows:

   WK — Work vector of length NF * (2 * NF + 3) - 1.

   S — Work vector of length NF * (NF + 1).

2. Informational errors

Type	Code	Description
4	1	NF = 1. No rotation is possible.
4	2	The rank of A^T * X is less than NF.

3. The target matrix is a hypothesized rotated factor loading matrix with loadings chosen (based on prior knowledge) to enhance interpretability. A simple structure solution will have most of the elements in **X** near zero or one (for correlation matrix loadings).

4. This routine may also be used to refine a solution obtained by analytic rotation in routine FROTA. Choose the target matrix so that it closely resembles the analytic solution but modified to have a simple structure.

Example

The following example is taken from Harman (1976, page 355). It involves the orthogonal Procrustes rotation of an 8 × 2 unrotated factor loading matrix. The original variables are measures of physical features ("lankiness" and "stockiness"). The target matrix *X* is also printed. Note that because different methods are used, Harman (1976) gets slightly different results.

```
      USE FOPCS_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDB, LDT, LDX, NF, NVAR
      PARAMETER  (LDA=8, LDB=8, LDT=2, LDX=8, NF=2, NVAR=8)
!
      REAL       A(LDA,NF), B(LDB,NF), T(LDT,NF), X(LDX,NF)
!
      DATA A/0.856, 0.848, 0.808, 0.831, 0.750, 0.631, 0.569, 0.607, &
      -0.324, -0.412, -0.409, -0.342, 0.571, 0.492, 0.510, 0.351/
      DATA X/0.9, 0.9, 0.9, 0.9, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, &
       0.9, 0.9, 0.9, 0.9/
!
      CALL FOPCS (A, X, B, T)
!
      CALL WRRRN ('A', A)
      CALL WRRRN ('X', X)
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      END
```

Output

```
           A
           1       2
1   0.8560  -0.3240
2   0.8480  -0.4120
3   0.8080  -0.4090
```

```
    4   0.8310  -0.3420
    5   0.7500   0.5710
    6   0.6310   0.4920
    7   0.5690   0.5100
    8   0.6070   0.3510
              X
              1       2
    1   0.9000   0.0000
    2   0.9000   0.0000
    3   0.9000   0.0000
    4   0.9000   0.0000
    5   0.0000   0.9000
    6   0.0000   0.9000
    7   0.0000   0.9000
    8   0.0000   0.9000

              B
    1         2
    1   0.8763   0.2643
    2   0.9235   0.1896
    3   0.8900   0.1677
    4   0.8674   0.2348
    5   0.2471   0.9096
    6   0.2009   0.7745
    7   0.1407   0.7510
    8   0.2677   0.6481

              T
              1       2
    1   0.7932   0.6090
    2  -0.6090   0.7932
```

FDOBL

more...

Computes a direct oblimin rotation of a factor loading matrix.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Nonpositive constant used to define the rotation. (Input)

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B may share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT = 0 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS = 0 defaults to 0.0001.
Default: EPS = 0.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: CALL FDOBL (A, W, B, T, FCOR [, ...])
Specific: The specific interface names are S_FDOBL and D_FDOBL.

FORTRAN 77 Interface

Single: CALL FDOBL (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T, LDT, FCOR, LDFCOR)
Double: The double precision name is DFDOBL.

Description

Routine FDOBL performs direct oblimin rotation. In this analytic method of rotation, the criterion function

$$Q = \sum_{r \neq s} \left[\sum_i \lambda_{ir}^2 \lambda_{is}^2 - \frac{\gamma}{p} \sum_i \lambda_{ir}^2 \sum_i \lambda_{is}^2 \right]$$

is minimized by finding a rotation matrix T such that $(\lambda_{ir}) = \Lambda = AT$ and $(T^T T)^{-1}$ is a correlation matrix. Here, $\gamma \leq 0$ is a user-specified constant (W) yielding a family of rotations, and p is the number of variables. The rotation is said to be direct because it minimizes Q with respect to the factor loadings directly, ignoring the reference structure.

Kaiser normalization can be performed on the factor loadings prior to rotation via the option parameter **NRM**. In Kaiser normalization (see Harman 1976), the rows of *A* are first "normalized" by dividing each row by the square root of the sum of its squared elements. After the rotation is complete, each row of *B* is "denormalized" by multiplication by its initial normalizing constant.

The method for optimizing *Q* is essentially the method first proposed by Jennrich and Sampson (1966). It proceeds by accumulating simple rotations where a simple rotation is defined to be one in which *Q* is optimized for a given factor in the plane of a second factor, and for which the requirement that $(T^TT)^{-1}$ be a correlation matrix is satisfied. An iteration is defined to be such that each of the **NF**(**NF** - 1) possible simple rotations is performed, where **NF** is the number of factors. When the relative change in *Q* from one iteration to the next is less than **EPS** (the user-specified convergence criterion), the algorithm stops. **EPS** = .0001 is usually sufficient. Alternatively, the algorithm stops when the user-specified maximum number of iterations, **MAXIT**, is reached. **MAXIT** = 30 is usually sufficient.

The parameter in the rotation, **γ**, is used to provide a family of rotations. Harman (1976) recommends that **γ** be strictly less than or equal to zero. When **γ** = 0.0, a direct quartimin rotation results. Other values of **γ** yield other rotations. Harman (1976) suggests that the direct quartimin rotations yield the most highly correlated factors while more orthogonal factors result as **γ** approaches **-∞**.

Comments

1. Workspace may be explicitly provided, if desired, by use of **F2OBL/DF2OBL**. The reference is:

   ```
   CALL F2OBL (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T, LDT, FCOR, LDFCOR, WK1,
      WK2, WK3)
   ```

   The additional arguments are as follows:

   WK1 — Real work vector of length equal to **NVAR**.

   WK2 — Real work vector of length equal to **NF**.

   WK3 — Real work vector of length equal to **NVAR**.

2 Informational errors

Type	Code	Description
3	1	The algorithm did not converge within MAXIT iterations.
4	1	NF = 1. No rotation is possible.

3. The parameter **W** determines the type of direct **OBLIMIN** rotation to be performed. In general, **W** must be negative. **W** = 0.0 yields direct quartimin rotation. As **W** approaches negative infinity, the orthogonality among the factors will increase.

Example

The example is a continuation of the example given in routine FACTR. It involves factors derived from nine variables and uses $\gamma = -1$.

```
          USE FDOBL_INT
          USE WRRRN_INT

          IMPLICIT   NONE
          INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
          REAL EPS, W
          PARAMETER  (EPS=0.00001, LDA=9, LDB=9, LDFCOR=3, LDT=3,NF=3, NVAR=9, &
          W=-1.0)
    !
          REAL A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), T(LDT,NF)
    !
          DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
          .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, -.2955, &
          -.4274, .0735, -.1933, -.2224, -.0354, -.1528, .1045, -.0778, .4914, &
          -.0117/
    !
          CALL FDOBL (A, W, B, T, FCOR, EPS=EPS)
    !
          CALL WRRRN ('B', B)
          CALL WRRRN ('T', T)
          CALL WRRRN ('FCOR', FCOR)
          END
```

Output

```
                B
          1       2       3
1   0.1127  -0.5145   0.2917
2   0.1847  -0.6602  -0.0019
3   0.0128  -0.6354  -0.0585
4   0.7797  -0.1751   0.0598
5   0.7147  -0.1813  -0.0959
6   0.8520   0.0038   0.1820
7   0.0355  -0.6845   0.1509
8   0.0276  -0.0941   0.6824
9   0.0729  -0.7100   0.2493

          T
          1       2       3
1   0.611  -0.462   0.203
2   0.923   0.813  -0.249
3   0.042   0.728   1.050

             FCOR
          1       2       3
1   1.000  -0.427   0.217
2  -0.427   1.000  -0.411
3   0.217  -0.411   1.000
```

FPRMX

more...

Computes an oblique Promax or Procrustes rotation of a factor loading matrix using a target matrix, including pivot and power vector options.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Constant used to define the orthomax orthogonal rotation. (Input)
Values for W are discussed in the Comments. W must be nonnegative. Not used if IMTH = 3.

F — Vector of length NF containing the power vector or the pivot constants depending upon whether IMTH = 1 or IMTH = 2, respectively. (Input)
Not used if IMTH = 3.

X — NVAR by NF target matrix for the rotation. (Output, if IMTH = 1 or 2; input, if IMTH = 3)
For IMTH = 1 or 2, X is the target matrix derived from the orthomax rotation. For IMTH = 3, X is input.

B — NVAR by NF matrix of rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
NVAR must be greater than or equal to 2.
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
NF must be greater than or equal to 2.
Default: Nf = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

IMTH — Method used for rotation. (Input)
Default: IMTH = 1.

IMTH	Method
1	The Promax method.
2	The pivotal Promax method.
3	Oblique Procrustes method.

NRM — Normalization option parameter. (Input)
NRM = 0 indicates that no row (Kaiser) normalization is to be performed in the orthomax orthogonal rotation. Otherwise, row normalization is performed. Not used when IMTH = 3.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
Thirty is typical. Not used if IMTH = 3.
Default: MAXIT = 30.

EPS — Convergence constant for the orthogonal rotation. (Input)
When the relative change in the orthomax criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS nonpositive defaults to EPS = 0.0001.
Default: EPS = 0.0001.

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of **FCOR** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDFCOR** = size (**FCOR**,1).

FORTRAN 90 Interface

Generic: **CALL FPRMX** (**A**, **W**, **F**, **X**, **B**, **T**, **FCOR** [, ...])
Specific: The specific interface names are **S_FPRMX** and **D_FPRMX**.

FORTRAN 77 Interface

Single: **CALL FPRMX** (**NVAR**, **NF**, **A**, **LDA**, **IMTH**, **NRM**, **W**, **MAXIT**, **EPS**, **F**, **X**, **LDX**, **B**, **LDB**, **T**, **LDT**, **FCOR**, **LDFCOR**)
Double: The double precision name is **DFPRMX**.

Description

Routine **FPRMX** performs oblique rotations via the Promax, the pivotal Promax, or the oblique Procrustes methods. In all of these methods, a target matrix X is first either computed or specified by the user. The differences in the methods relate to how the target matrix is first obtained.

Given a p x k target matrix, X, and a p x k orthogonal matrix of unrotated factor loadings, A, compute the rotation matrix T as follows: First regress each column of A on X yielding a k x k matrix β. Then, let $\gamma = \text{diag}(\beta^T \beta)$ where diag denotes the diagonal matrix obtained from the diagonal of the square matrix. Standardize β to obtain $T = \gamma^{-1/2} \beta$. The rotated loadings are computed as $B = AT$ while the factor correlations can be computed as the inverse of the $T^T T$ matrix.

In the Promax method, the unrotated factor loadings are first rotated according to an orthomax criterion via routine FROTA. The target matrix X is taken as the elements of the B raised to a power greater than one but retaining the same sign as the original loadings. In **FPRMX**, column i of the rotated matrix B is raised to the power **F**(i). A power of four is commonly used. Generally, the larger the power, the more oblique the solution.

In the pivotal Promax method, the unrotated matrix is first rotated to an orthomax orthogonal solution as in the Promax case. Then, rather than raising the i-th column in B to the power **F**(i), the elements x_{ij} of X are obtained from the elements b_{ij} of B by raising the ij element of B to the power **F**(i)/b_{ij}. This has the effects of greatly increasing in X those elements in B that are greater in magnitude than the pivot elements **F**(i), and of greatly decreasing those elements that are less than **F**(i).

In the oblique Procrustes method, the elements of X are specified by the user as input to the **FPRMX** routine. No orthogonal rotation is performed in the oblique Procrustes method.

Comments

1. Workspace may be explicitly provided, if desired, by use of F2RMX/DF2RMX. The reference is:

   ```
   CALL F2RMX (NVAR, NF, A, LDA, IMTH, NRM, W, MAXIT, EPS, F, X, LDX, B, LDB, T, LDT, FCOR,
       LDFCOR, QR, QRAUX, IPVT, WORK)
   ```

   The additional arguments are as follows:

   QR — Work vector of length NVAR * NF.

   QRAUX — Work vector of length NF.

   IPVT — Work vector of length NF.

   WORK — Work vector of length 2 * NF.

2. Arguments W, EPS, and NRM are input arguments to routine FROTA when IMTH = 1 or 2. (They are not used when IMTH = 3.) See FROTA for common values of W. Generally, W can be any positive real number, but the best values lie in the range (1.0, 5.0 * NF). Generally, the variances accounted for by the factors approach the same value as W increases.

3. For IMTH = 1, all F(*j*) should be greater than 1.0, typically 4.0. Generally, the larger the values of F(*j*), the more oblique the solution will be. For IMTH = 2, F(*j*) should be in the interval (0.0, 1.0).

4. When IMTH = 3, the target matrix, X, is a hypothesized rotated factor loading matrix based upon prior knowledge with loadings chosen to enhance interpretability. A simple structure solution will have most of the weights X(*i*, *j*) either zero or large in magnitude. Note that the two options IMTH = 1 or 2 attempt to achieve this simple structure based upon an initial orthogonal rotation.

Example

The following example is a continuation of the example in the FROTA procedure. It involves nine variables and three factors. The pivotal Promax method is illustrated.

```
      USE FPRMX_INT
      USE WRRRN_INT
      IMPLICIT NONE

      INTEGER    IMTH, LDA, LDB, LDFCOR, LDT, LDX, NF, NVAR
      REAL       W
      PARAMETER  (IMTH=2, LDA=9, LDB=9, LDFCOR=3, LDT=3, LDX=9, NF=3, &
                  NVAR=9, W=1.0)
!
      REAL       A(LDA,NF), B(LDB,NF), F(NF), FCOR(LDFCOR,NF), &
                 T(LDT,NF), X(LDX,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
```

```
      DATA F/0.5, 0.5, 0.5/
!
      CALL FPRMX (A, W, F, X, B, T, FCOR, IMTH=IMTH)
!
      CALL WRRRN ('X', X)
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
                X
         1       2       3
1   0.0800  -0.6157   0.2967
2   0.2089  -0.7311   0.0007
3   0.0037  -0.6454   0.0000
4   0.8800  -0.1681   0.0032
5   0.8116  -0.1030   0.0000
6   0.9096  -0.0122   0.0640
7   0.0291  -0.7649   0.0982
8   0.0001  -0.0546   0.7563
9   0.0866  -0.8189   0.2807

             B
         1       2       3
1   0.0997  -0.5089   0.3038
2   0.1900  -0.6463   0.0077
3   0.0163  -0.6270  -0.0421
4   0.7991  -0.1469   0.0285
5   0.7408  -0.1531  -0.1256
6   0.8668   0.0308   0.1436
7   0.0280  -0.6777   0.1699
8  -0.0094  -0.1017   0.6911
9   0.0611  -0.7031   0.2683

            T
        1       2       3
1   0.617  -0.439   0.189
2   0.963   0.839  -0.318
3  -0.015   0.707   1.039

           FCOR
        1       2       3
1   1.000  -0.464   0.316
2  -0.464   1.000  -0.395
3   0.316  -0.395   1.000
```

FHARR

Computes an oblique rotation of an unrotated factor loading matrix using the Harris-Kaiser method.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Constant used to define the rotation. (Input)
The value of W must be nonnegative. See Comments.

C — Constant between zero and one used to define the rotation. (Input)
See Comments.

B — NVAR by NF matrix containing the rotated factor loadings. (Output)

T — NF by NF factor rotation matrix. (Output)

FCOR — NF by NF matrix containing the factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. Otherwise, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
A typical value is 30.
Default: MAXIT = 30.

EPS — Convergence constant for the rotation angle. (Input)
EPS = 0.0001 is typical. If EPS is less that or equal to 0.0, then EPS = 0.0001 is used.
Default: EPS = 0.0.

SCALE — Vector of length NVAR containing a scaling vector. (Input)
All elements in SCALE should be set to one if principal components or unweighted least squares was used to obtain the unrotated factor loadings. The elements of SCALE should be set to the unique error variances (vector UNIQ in subroutine FACTR) if the principal factor, generalized least squares, maximum likelihood, or the image method was used. Finally, in alpha factor analysis, the elements of SCALE should be set to the communalities (one minus the uniquenesses in standardized data).
Default: SCALE = 1.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: CALL FHARR (A, W, C, B, T, FCOR [, ...])
Specific: The specific interface names are S_FHARR and D_FHARR.

FORTRAN 77 Interface

Single: CALL FHARR (NVAR, NF, A, LDA, NRM, MAXIT, W, C, EPS, SCALE, B, LDB, T, LDT, FCOR, LDFCOR)
Double: The double precision name is DFHARR.

Description

Routine FHARR performs an oblique analytic rotation of unrotated factor loadings via a method proposed by Harris and Kaiser (1964). In this method of rotation, the eigenvectors obtained from the factor extraction are weighted by a factor $\Delta^{c/2}$ where Δ is the diagonal matrix of eigenvalues obtained in the factor extraction and c is a specified constant. These transformed eigenvectors are then rotated according to an orthomax criterion.

The transformation used to obtain the weighted eigenvectors, Γ^*, from the unrotated loadings, A, is given as $\Gamma^* = \Psi^{-1/2} A \Delta^{(c-1)/2}$ where Ψ is the matrix of unique error variances output by routine FACTR. The matrix should be set to an identity matrix if the principal component, unweighted least squares, or alpha factor analysis method is used in routine FACTR to obtain the unrotated factor loadings (IMTH = 0,1, or 5). This is required because in these methods of factor analysis, the eigenvectors are not premultiplied by a diagonal matrix when obtaining the unrotated factor loadings.

After Γ^* has been computed, it is rotated according to a user-selected orthomax criterion. The member of the orthomax family to be used is selected via a constant W. (See the description of routine FROTA.) Because Γ^* is used in place of A (the unrotated factor loadings in routine FROTA), the matrix resulting from the rotation is (after standardizing by pre and postmultiplication by the diagonal matrices U^{-1} and Δ^{1-c}) a matrix of obliquely rotated loadings.

Note that the effect of W is less pronounced than the effect of C. Using c = 1.0 yields an orthogonal orthomax rotation while c = 0.0 yields the most oblique factors. A common choice for c is given by c = 0.5. One good choice for W is 1.0. W = 1.0 yields a varimax rotation on the weighted eigenvectors.

Comments

1. Workspace may be explicitly provided, if desired, by use of F2ARR/DF2ARR. The reference is:

   ```
   CALL F2ARR (NVAR, NF, A, LDA, NRM, MAXIT, W, C, EPS, SCALE, B, LDB, T, LDT, FCOR,
       LDFCOR, RWK1, RWK2 )
   ```

   The additional arguments are as follows:

   RWK1 — Real work vector of length equal to 2 * NF.

   RWK2 — Real work vector of length equal to NVAR.

2. Argument C must be between 0.0 and 1.0. The larger C is, the more orthogonal the rotated factors are. Rarely, should C be greater than 0.5.

3. Arguments W, EPS, and NRM are arguments to routine FROTA. See FROTA for common values of W in orthogonal rotations. For FHARR, the best values of W are in the range (0.0, 5.0 * NF). Generally, the variances of the factors converge to the same value as W increases.

Example

The example is a continuation of the example in routine FROTA. It involves 9 variables. A rotation with row normalization and 3 factors is performed.

```
      USE FHARR_INT
      USE WRRRN_INT
      IMPLICIT NONE
```

```
      INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
      REAL       C, W
      PARAMETER  (C=0.5, LDA=9, LDB=9, LDFCOR=3, LDT=3, NF=3, &
                  NVAR=9, W=1.0)
!
      REAL A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), SCALE(NVAR), &
           T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960,  &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA SCALE/.4505, .4271, .6165, .2123, .3805, .1769, .3995, &
      .4616, .2309/
!
      CALL FHARR (A, W, C, B, T, FCOR, SCALE=SCALE)
!
      CALL WRRRN ('B', B)
      CALL WRRRN ('T', T)
      CALL WRRRN ('FCOR', FCOR)
      END
```

Output

```
                B
          1       2       3
1    0.1542  -0.5103   0.2749
2    0.2470  -0.6477  -0.0233
3    0.0744  -0.6185  -0.0750
4    0.7934  -0.1897   0.0363
5    0.7329  -0.1909  -0.1175
6    0.8456  -0.0194   0.1610
7    0.0966  -0.6713   0.1320
8    0.0198  -0.1067   0.6773
9    0.1340  -0.6991   0.2285

            T
        1       2       3
1   0.649  -0.469   0.175
2   0.850   0.777  -0.249
3  -0.053   0.687   1.065

          FCOR
        1       2       3
1   1.000  -0.335   0.250
2  -0.335   1.000  -0.413
3   0.250  -0.413   1.000
```

FGCRF

more...

Computes direct oblique rotation according to a generalized fourth-degree polynomial criterion.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

W — Vector of length 4 containing the constants ω_1, ω_2, ω_3, ω_4 necessary to define the rotation. (Input)
Some common rotations are

Rotation	W(1)	W(2)	W(3)	W(4)
Quartimin	0	1	0	-1
Covarimin	-1/NVAR	1	1/NVAR	-1
Oblimin	-γ/NVAR	1	γ/NVAR	-1
Crawford-Ferguson	0	K_1	K_2	$-K_1 - K_2$

where K_1, K_2, and γ are constants (determined by the user).

B — NVAR by NF matrix of rotated factor loadings. (Output)
If A is not needed, A and B can share the same storage locations.

T — NF by NF matrix containing the rotation transformation matrix. (Output)

FCOR — NF by NF matrix of factor correlations. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

NRM — Row normalization option. (Input)
If NRM = 1, then row (i.e., Kaiser) normalization is performed. If NRM = 0, row normalization is not performed.
Default: NRM = 1.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is typical. MAXIT ≤ 30 defaults to 30 iterations.
Default: MAXIT = 30.

EPS — Convergence constant. (Input)
When the relative change in the criterion function is less than EPS from one iteration to the next, convergence is assumed. EPS = 0.0001 is typical. EPS ≤ 0.0 defaults to 0.0001.
Default: EPS = 0.0.

LDB — Leading dimension of B exactly as specified in the dimension statement in the calling program. (Input)
Default: LDB = size (B,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDFCOR — Leading dimension of FCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDFCOR = size (FCOR,1).

FORTRAN 90 Interface

Generic: CALL FGCRF (A, W, B, T, FCOR [, ...])
Specific: The specific interface names are S_FGCRF and D_FGCRF.

FORTRAN 77 Interface

Single: CALL FGCRF (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T, LDT, FCOR, LDFCOR)
Double: The double precision name is DFGCRF.

Description

Routine **FGCRF** performs direct oblique factor rotation for an arbitrary fourth-degree polynomial criterion function. Let p = **NVAR** denote the number of variables, and let k = **NF** denote the number of factors. Then, the criterion function

$$Q = \omega_1\left(\sum_{i=1}^{p}\sum_{r=1}^{k}\lambda_{ir}^2\right)^2 + \omega_2\sum_{i=1}^{p}\left(\sum_{r=1}^{k}\lambda_{ir}^2\right)^2 + \omega_3\sum_{r=1}^{k}\left(\sum_{i=1}^{p}\lambda_{ir}^2\right)^2 + \omega_4\sum_{i=1}^{p}\sum_{r=1}^{k}\lambda_{ir}^4$$

is minimized by finding a rotation matrix T such that $(\lambda_{ij}) = \Lambda = AT$ and $T^{-1}\ (T^{-1})^T$ is a correlation matrix. Here, ω_i = **W**(i), i = 1, …, 4 are user specified constants. The rotation is said to be direct because it minimizes Q with respect to the factor loadings directly, ignoring the reference structure (see, e.g., Harman, 1976).

Kaiser normalization (Harman, 1976) is specified when option parameter **NRM** = 1. When Kaiser normalization is performed, the rows of A are first "normalized" by dividing each row by the square root of the sum of its squared elements. The rotation is then performed. The rows of **B** are then "denormalized" by multiplying each row by the initial row normalizing constant.

The criterion function Q was first proposed by Jennrich (1973). It generalizes the oblimin criterion function and the criterion function proposed by Crawford and Ferguson (1970) to an arbitrary fourth degree criterion. Q is optimized by accumulating simple rotations where a simple rotation is defined to be an optimal factor rotation (with respect to Q) for two columns of **Λ**, and for which the requirement that $T^{-1}\ (T^{-1})^T$ be a correlation matrix is satisfied. **FGCRF** determines the optimal simple rotation by finding the roots of a cubic polynomial equation. The details are contained in Clarkson and Jennrich (1988).

Table 1 – Specific Criteria in the General Symmetric Family

Criterion	ω_1	ω_2	ω_3	ω_4
Quartimin	0	1	0	-1
Covarimin	-1/p	1	1/p	-1
Oblimin	-**γ**/p	1	**γ**/p	-1
Crawford-Ferguson	0	K_1	K_2	-K_1 - K_2

An iteration is complete after all possible $k(k - 1)$ simple rotations have been performed. When the relative change in Q from one iteration to the next is less than **EPS**, the algorithm stops. **EPS** = .0001 is usually sufficient. Alternatively, the algorithm stops when the user specified maximum number of iterations, **MAXIT**, is reached. **MAXIT** = 30 is typical.

Notes

The parameters in the rotation, ω_1, provide for a two-dimensional family of rotations. When $\omega_1 = -\gamma/p$, $\omega_2 = 1$, $\omega_3 = \gamma/p$, and $\omega_4 = -1$, then a direct oblimin rotation with parameter γ is performed. Direct oblimin rotations are also performed by routine FDOBL, which is somewhat faster. For $\omega_1 = 0$, $\omega_2 = K_1$, $\omega_3 = K_2$, and $\omega_4 = -(K_1 + K_2)$ direct Crawford-Ferguson rotation with parameters K_1 and K_2 results (see Crawford and Ferguson 1970, or Clarkson and Jennrich 1988). Other values of ω yield other rotations. Common values for ω are as in Table 1.

Comments

1. Workspace may be explicitly provided, if desired, by use of F2CRF/DF2CRF. The reference is:

   ```
   CALL F2CRF (NVAR, NF, A, LDA, NRM, W, MAXIT, EPS, B, LDB, T, LDT, FCOR, LDFCOR, RWK1,
       RWK2, RWK3)
   ```

   The additional arguments are as follows:

   RWK1 — Work vector of length NVAR.

   RWK2 — Work vector of length NVAR * (NF + 1).

   RWK3 — Work vector of length NF^2.

2. Informational Error

Type	Code	Description
3	1	The algorithm did not converge within MAXIT iterations.

Example

The example is a continuation of the example in routine FACTR. It involves nine variables. A Crawford-Ferguson rotation with row normalization and 3 factors is performed.

```
      USE FGCRF_INT
      USE WRRRN_INT
      INTEGER    LDA, LDB, LDFCOR, LDT, NF, NVAR
      PARAMETER  (LDA=9, LDB=9, LDFCOR=3, LDT=3, NF=3, NVAR=9)
!
      REAL       A(LDA,NF), B(LDB,NF), FCOR(LDFCOR,NF), T(LDT,NF), W(4)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, -.2955, &
      -.4274, .0735, -.1933, -.2224, -.0354, -.1528, .1045, -.0778, .4914, &
      -.0117/
      DATA W/0.0, 7.0, 1.0, -8.0/
!
      CALL FGCRF (A, W, B, T, FCOR)
!
      CALL WRRRN ('B', B)
```

```
          CALL WRRRN ('T', T)
          CALL WRRRN ('FCOR', FCOR)
          END
```

Output

```
                  B
             1         2         3
 1    0.1156  -0.3875    0.3992
 2    0.2161  -0.5831    0.0924
 3    0.0422  -0.5859    0.0264
 4    0.8051  -0.0906    0.0886
 5    0.7495  -0.1373   -0.0839
 6    0.8639   0.1045    0.1987
 7    0.0527  -0.5792    0.2671
 8   -0.0162   0.0779    0.7748
 9    0.0852  -0.5765    0.3803
                T
           1       2       3
 1    0.632  -0.327   0.290
 2    0.935   0.737  -0.399
 3   -0.060   0.907   1.066

              FCOR
           1       2       3
 1    1.000  -0.434   0.365
 2   -0.434   1.000  -0.498
 3    0.365  -0.498   1.000
```

FIMAG

more...

Computes the image transformation matrix.

Required Arguments

T — NF by NF transformation matrix. (Input)

TI — NF by NF image transformation matrix. (Output)

Optional Arguments

NF — Number of factors. (Input)
Default: NF = size (T,2).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDTI — Leading dimension of TI exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTI = size (TI,1).

FORTRAN 90 Interface

Generic: CALL FIMAG (T, TI [, ...])
Specific: The specific interface names are S_FIMAG and D_FIMAG.

FORTRAN 77 Interface

Single: CALL FIMAG (NF, T, LDT, TI, LDTI)
Double: The double precision name is DFIMAG.

Description

Routine **FIMAG** computes the image transformation matrix **TI** from the factor rotation matrix (*T*). The image transformation matrix takes the unrotated factor loadings into the factor structure matrix when the unrotated loadings are computed from a correlation matrix. It is computed as the inverse of the transpose of the factor rotation matrix *T*. When orthogonal rotations are used, $(T^T)^{-1} = T$ so there is no reason to compute the image transformation matrix.

Comments

1. Workspace may be explicitly provided, if desired, by use of **F2MAG/DF2MAG**. The reference is:

   ```
   CALL F2MAG (NF, T, LDT, TI, LDTI, RWK, IWK)
   ```

   The additional arguments are as follows:

   RWK — Real work vector of length **NF** + **NF**(**NF** - 1)/2.

   IWK — Integer work vector of length **NF**.

2. Informational Error

Type	Code	Description
3	1	T is ill-conditioned. The solution may not be accurate.

Example

This example is a continuation of the example contained in the manual document for routine FROTA. The image transformation matrix is obtained from the orthogonal rotation matrix. Some small differences between the matrix **TI** when compared with the matrix **T** computed via routine FROTA can be seen. These differences are because of roundoff error since for orthogonal rotations, the image transformation matrix is the same as the rotation matrix.

```
      USE FIMAG_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDT, LDTI, NF
      PARAMETER  (LDT=3, LDTI=3, NF=3)
!
      REAL       T(LDT,NF), TI(LDTI,NF)
!
      DATA T/.7307, .6816, -.0382, -.5939, .6623, .4569, .3367, -.3112, &
      .8887/
!
      CALL FIMAG (T, TI)
!
      CALL WRRRN ('TI', TI)
```

```
          END
```

Output

```
                TI
           1        2        3
    1   0.7307  -0.5938   0.3367
    2   0.6816   0.6622  -0.3112
    3  -0.0382   0.4569   0.8887
```

FRVAR

more...

Computes the factor structure and the variance explained by each factor.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

T — NF by NF factor rotation matrix. (Input)

VAR — Vector of length NVAR containing the variances of the original variables. (Input)
If standardized variables were used (i.e., the loadings are from a correlation matrix), then set VAR(1) to any negative number. In this case, VAR may be dimensioned of length one.

S — NVAR by NF factor structure matrix. (Output)

FVAR — Vector of length NF containing the variance accounted for by each of the NF rotated factors. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDS — Leading dimension of S exactly as specified in the dimension statement in the calling program. (Input)
Default: LDS = size (S,1).

FORTRAN 90 Interface

Generic: CALL FRVAR (A, T, VAR, S, FVAR [, ...])
Specific: The specific interface names are S_FRVAR and D_FRVAR.

FORTRAN 77 Interface

Single: CALL FRVAR (NVAR, NF, A, LDA, T, LDT, VAR, S, LDS, FVAR)
Double: The double precision name is DFRVAR.

Description

Routine FRVAR computes the factor structure matrix (the matrix of correlations between the observed variables and the hypothesized factors) and the variance explained by each of the factors (for orthogonal rotations). For oblique rotations, FRVAR computes a measure of the importance of the factors, the sum of the squared elements in each column.

Let Δ denote the diagonal matrix containing the elements of the vector VAR along its diagonal. The estimated factor structure matrix *S* is computed as

$$S = \Delta^{-\frac{1}{2}} A(T^{-1})^T$$

while the elements of FVAR are computed as the diagonal elements of

$$S^T \Delta^{\frac{1}{2}} AT$$

If the factors were obtained from a correlation matrix (or the factor variances for standardized variables are desired), then the elements of the vector VAR should either all be 1.0, or the first element of VAR should be set to any negative number. In either case, variances of 1.0 are used.

The user should be careful to input the unrotated loadings. When obliquely rotated loadings are input, the output vector FVAR contains a measure of each factors importance, but it does not contain the variance of each factor.

Comments

Workspace may be explicitly provided, if desired, by use of F2VAR/DF2VAR. The reference is

```
CALL F2VAR (NVAR, NF, A, LDA, T, LDT, VAR, S, LDS, FVAR, TINV, WK, IWK)
```

The additional arguments are as follows:

TINV — Work vector of length NF^2.

WK — Work vector of NF * (1 + NVAR).

IWK — Work vector of length NF.

Example

The following example illustrates the use of routine FRVAR when the structure and an index of factor importance for obliquely rotated loadings (obtained from routine FDOBL) are desired. Note in this example that the elements of FVAR are not variances since the rotation is oblique.

```
      USE FRVAR_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDS, LDT, NF, NVAR
      PARAMETER  (LDA=9, LDS=9, LDT=3, NF=3, NVAR=9)
!
      REAL       A(LDA,NF), FVAR(NF), S(LDS,NF), T(LDT,NF), VAR(NVAR)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, .1045, &
      -.0778, .4914, -.0117/
!
      DATA T/0.611, 0.923, 0.042, -0.462, 0.813, 0.728, 0.203, -0.249, &
      1.050/
!
      DATA VAR/9*1.0/
!
      CALL FRVAR (A, T, VAR, S, FVAR)
!
      CALL WRRRN ('S', S)
      CALL WRRRN ('FVAR', FVAR, 1, NF, 1)
      END
```

Output

```
                S
          1       2       3
1    0.3958  -0.6825   0.5274
2    0.4662  -0.7385   0.3093
3    0.2715  -0.6171   0.2052
4    0.8673  -0.5328   0.3010
5    0.7712  -0.4473   0.1338
```

```
    6   0.8897  -0.4348   0.3654
    7   0.3606  -0.7618   0.4397
    8   0.2160  -0.3860   0.7270
    9   0.4303  -0.8437   0.5566

            FVAR
        1       2       3
    2.170   2.559   0.915
```

FCOEF

more...

Computes a matrix of factor score coefficients for input to the routine FSCOR.

Required Arguments

A — NVAR by NF matrix of unrotated factor loadings. (Input)

COV — The variance-covariance or correlation matrix of order NVAR from which the factor loadings were obtained. (Input)
COV is not used and may be dimensioned of length 1 if IMTH = 2 or 5.

T — NF by NF factor rotation matrix or transformation matrix. (Input)
If the image method is being used, then routine FIMAG needs to be called after the rotation routine to obtain the image transformation matrix. TI is then input for T in FCOEF. If factor score coefficients for the unrotated loadings are desired, T should be set to the identity matrix prior to calling FCOEF.

SCOEF — NVAR by NF factor score coefficient matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (A,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

IMTH — Method to be used to obtain the factor scores. (Input)
Default: IMTH = 1.

IMTH	Method
1	Regression method
2	Least squares method
3	Bartlett method
4	Anderson and Rubin method
5	Image score for image analysis

See the Comments for a table of the methods that are appropriate for a given type of factor extraction and rotation.

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDT — Leading dimension of T exactly as specified in the dimension statement in the calling program. (Input)
Default: LDT = size (T,1).

LDSCOE — Leading dimension of SCOEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOE = size (SCOEF,1).

FORTRAN 90 Interface

Generic: CALL FCOEF (A, COV, T, SCOEF [, ...])
Specific: The specific interface names are S_FCOEF and D_FCOEF.

FORTRAN 77 Interface

Single: CALL FCOEF (NVAR, NF, A, LDA, IMTH, COV, LDCOV, T, LDT, SCOEF, LDSCOE)
Double: The double precision name is DFCOEF.

Description

Routine FCOEF computes factor score coefficients that may subsequently be used in computing the factor scores in routine FIMAG. Five options for computing the coefficients are available according to the input parameter IMTH. The method that should be used depends upon the method used in extracting the factor loadings. See the Comments section for values to use for IMTH when various methods of factor extraction are used.

Let S denote the covariance (or correlation) matrix from which the factors were obtained, let β denote the factor score coefficients, let $U^2 = \text{diag}(S - AA^T)$ denote the unique error variances, and let $B = AT$ denote the rotated factor loadings (if coefficients for the unrotated loadings are desired, then $B = A$). The various methods for computing the factor score coefficients are discussed in detail in Harman (1976, Chapter 16) and are given as follows:

1. The regression method may be used with any method of factor extraction and rotation (but not with image analysis). The coefficients are computed as follows:

$$\hat{\beta} = S^{-1}B(T^TT)^{-1}$$

2. The least-squares method may also be used with any method of factor extraction and rotation (but not in image analysis). The factor score coefficients are computed as

$$\hat{\beta} = B(B^TB)^{-1}$$

   Note that estimated coefficients in the least-squares method yield different factor scores depending upon the scale of the observed variables. In particular, factor scores computed from standardized data (i.e., for the correlation matrix) will be different from factor scores computed from the raw data (i.e., from a covariance matrix). Generally, the differences will not be great. These differences are not observed in any of the other methods.

3. The Bartlett (1937) method may be used with common factor models only. The coefficients are computed as

$$\hat{\beta} = U^{-2}B(B^TU^{-2}B)^{-1}$$

4. The Anderson and Rubin (1956) method may also be used with common factor models only. It is a modification of the Bartlett method where the modification is used to insure that the factors obtained are orthogonal. The factor score coefficients are computed as

$$\hat{\beta} = U^{-2}B(B^TU^{-2}SU^{-2}B)^{-\frac{1}{2}}$$

5. The image method is appropriate for image analysis. In this method, the coefficients are computed as

$$\hat{\beta} = B_IT_I = A(T^T)^{-1}(T^T)^{-1}$$

   where B_I is the image score coefficient matrix, and T_I is the image transformation matrix (the matrix TI in routine FIMAG).

Harman (1976, pages 385-387) discusses choosing a method for computing factor score coefficients. According to Harman, the most desirable properties of any of the methods can be summarized as follows.

- Validity—The estimated factor scores should have high correlation with the population factor scores.
- Orthogonality—The estimated factor scores should not correlate highly with one another.
- Univocal—The estimated factor scores should correlate only with the corresponding true factor scores.

With these criteria in mind, Harman states that:

1. The regression method yields factor scores which usually have the highest correlation with the true factor scores.
2. The Bartlett and least-squares methods are univocal but not orthogonal.
3. The Anderson and Rubin method is orthogonal but not univocal.
4. Univocality is of more significance than orthogonality.

Comments

1. Workspace may be explicitly provided, if desired, by use of **F2OEF**/**DF2OEF**. The reference is:

   ```
   CALL F2OEF (NVAR, NF, A, LDA, IMTH, COV, LDCOV, T, LDT, SCOEF, LDSCOE, B, RWK1, S,
       UNIQ, RWK2)
   ```

   The additional arguments are as follows:

   B — Real work vector of length 2 * NVAR * NF if **IMTH** = 4, and of length NVAR * NF otherwise.

   RWK1 — Real work vector of length $NVAR^2$ if **IMTH** = 1 or 4, and of length NF^2 if **IMTH** = 2 or 3. Otherwise, **RWK**1 is of length 1.

   S — Real work vector of length NF^2 if **IMTH** = 4. Otherwise, **S** is dimensioned of length 1.

   UNIQ — Real work vector of length NVAR if **IMTH** = 2, 3, or 4. Otherwise, **UNIQ** is dimensioned of length 1.

   RWK2 — Real work vector of length NF if **IMTH** is not 5. If **IMTH** = 5, then **RWK**2 is of length 1.

2. The method used for computing the factor score coefficients depends both upon the method used to extract the factor loadings in routine FACTR and whether the factor loadings were orthogonally or obliquely rotated. In the following table, the numbers in parentheses refer to IMTH in routine FACTR and the numbers in the cells refer to IMTH in FCOEF.

FACTR Method (IMTH)	No Rotation	Orthogonal Rotation	Oblique Rotation
Component (0)	1, 2	1, 2	1, 2
Image (4)	5	5	5
Common Factor			
ULS (1)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
GLS (2)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
ML (3)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4
Alpha (5)	1, 2, 3, 4	1, 2, 3, 4	1, 2, 3, 4

Example

In the following example, the regression method is used to obtain estimated factor score coefficients for a 9-variable problem with 3 factors. An oblique rotation method was used with the maximum likelihood common factor model to obtain the factor loadings. Routine FDOBL was used to obtain the oblique factor loadings.

```
      USE FCOEF_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    IMTH, LDA, LDCOV, LDSCOE, LDT, NF, NVAR
      PARAMETER  (IMTH=1, LDA=9, LDCOV=9, LDSCOE=9, LDT=3, NF=3, NVAR=9)
!
      REAL A(LDA,NF), COV(LDCOV,NVAR), SCOEF(LDSCOE,NF), T(LDT,NF)
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, &
      .1045, -.0778, .4914, -.0117/
!
      DATA T/0.611, 0.923, 0.042, -0.462, 0.813, 0.728, 0.203, &
      -0.249, 1.050/
!
      DATA COV/1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434,&
      0.639, 0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, 0.283, &
      0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, 0.452, 0.219, &
      0.504, 0.471, 0.506, 0.355, 1.000, 0.691, 0.791, 0.443, 0.285, &
      0.505, 0.346, 0.418, 0.270, 0.691, 1.000, 0.679, 0.383, 0.149, &
      0.409, 0.426, 0.462, 0.254, 0.791, 0.679, 1.000, 0.372, 0.314, &
      0.472, 0.576, 0.547, 0.452, 0.443, 0.383, 0.372, 1.000, 0.385, &
      0.680, 0.434, 0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, &
      0.470, 0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
      1.000/
```

```
!
      CALL FCOEF (A, COV, T, SCOEF)
!
      CALL WRRRN ('SCOEF', SCOEF)
      END
```

Output

```
                SCOEF
            1       2       3
1   -0.0102  -0.1350   0.1781
2    0.0269  -0.2191  -0.0825
3   -0.0080  -0.1536  -0.0791
4    0.3788  -0.0597  -0.0596
5    0.2067  -0.0554  -0.1768
6    0.4885   0.1103   0.2084
7   -0.0258  -0.2317   0.0612
8   -0.0474   0.0345   0.5269
9   -0.0431  -0.3967   0.2507
```

FSCOR

Computes a set of factor scores given the factor score coefficient matrix.

Required Arguments

SCOEF — NVAR by NF matrix containing the factor score coefficients as output from routine FCOEF. (Input)

X — NOBS by NVAR data matrix for which factor scores are to be computed. (Input)

XBAR — Vector of length NVAR containing the means of the NVAR variables. (Input)

STD — Vector of length NVAR containing the standard deviations of the NVAR variables. (Input)
If STD(1) is not positive, then it is assumed that the factor score coefficients are from a covariance matrix and the observed variables are not standardized to unit variance.

SCOR — NOBS by NF matrix containing the factor scores. (Output)
If X is not needed, X and SCOR can share the same memory locations.

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (SCOEF,1).

NF — Number of factors. (Input)
Default: NF = size (SCOEF,2).

LDSCOE — Leading dimension of SCOEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOE = size (SCOEF,1).

NOBS — Number of observations for which factor scores are to be computed. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X, 1).

LDSCOR — Leading dimension of SCOR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSCOR = size (SCOR,1).

FORTRAN 90 Interface

Generic: CALL FSCOR (SCOEF, X, XBAR, STD, SCOR [, ...])
Specific: The specific interface names are S_FSCOR and D_FSCOR.

FORTRAN 77 Interface

Single: CALL FSCOR (NVAR, NF, SCOEF, LDSCOE, NOBS, X, LDX, XBAR, STD, SCOR, LDSCOR)
Double: The double precision name is DFSCOR.

Description

Routine FSCOR computes the factor scores from the factor score coefficient matrix. In FSCOR, the data are input as originally observed, and standardization is performed as required according to the value of STD(1). When the factor loadings are computed from the correlation matrix, the observed data must be standardized to a mean of zero and a variance of one prior to computing the factor scores. This requires that STD contain the observed standard deviations of the observed data and that XBAR contain the means. On the other hand, if the factor loadings are computed from the covariance matrix, then the observed data must be standardized to a mean of zero, but the variance must be left unchanged in computing the factor scores. In this case, STD(1) must be negative or zero.

After standardizing the observed data, the factor scores are computed as the product of the factor score coefficient matrix times the standardized data. If factor scores are computed from the same data from which the covariance matrix was computed, then the sample variance (using weights and frequencies as required) of the resulting factor scores will be 1.0.

Comments

Workspace may be explicitly provided, if desired, by use of F2COR/DF2COR. The reference is

```
CALL F2COR (NVAR, NF, SCOEF, LDSCOE, NOBS, X, LDX, XBAR, STD, SCOR, LDSCOR, WK)
```

The additional argument is

WK — Work vector of length NVAR.

Example

The following example is a continuation of the example given in the manual document for routine FACTR. The rotated loadings are those obtained from the manual document for routine FROTA, and the factor score coefficients are as described in the manual document for routine FCOEF.

```
      USE FSCOR_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDSCOE, LDSCOR, LDX, NF, NOBS, NVAR
      PARAMETER  (LDSCOE=2, LDSCOR=5, LDX=5, NF=1, NOBS=5, NVAR=2)
!
      REAL       SCOEF(NVAR,NF), SCOR(LDSCOR,NF), STD(NVAR), X(LDX,NVAR),&
      XBAR(NVAR)
!
      DATA X/40.0, 60.0, 30.0, 15.0, 45.0, 3.0, 9.0, 2.0, 0.0, 4.0/
      DATA SCOEF/0.33563, 0.33562/
      DATA XBAR/38.0, 3.6/, STD/16.80774, 3.361547/
!
      CALL FSCOR (SCOEF, X, XBAR, STD, SCOR)
!
      CALL WRRRN ('Factor Scores', SCOR)
      END
```

Output

```
 Factor Scores
 1  -0.0200
 2   0.9785
 3  -0.3195
 4  -0.8187
 5   0.1797
```

FRESI

Computes communalities and the standardized factor residual correlation matrix.

Required Arguments

COV — NVAR by NVAR matrix containing the variance-covariance or correlation matrix. (Input)
Only the upper triangular part of COV is referenced.

A — NVAR by NF orthogonal factor-loading matrix. (Input)

Y — Vector of length NVAR containing the communalities. (Output)

RESID — NVAR by NVAR matrix containing the normalized residual variance-covariance or correlation matrix. (Output)

Optional Arguments

NVAR — Number of variables. (Input)
Default: NVAR = size (COV,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

NF — Number of factors. (Input)
Default: NF = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

LDRESI — Leading dimension of RESID exactly as specified in the dimension statement in the calling program. (Input)
Default: LDRESI = size (RESID,1).

FORTRAN 90 Interface

Generic: CALL FRESI (COV, A, Y, RESID [, ...])
Specific: The specific interface names are S_FRESI and D_FRESI.

FORTRAN 77 Interface

Single: CALL FRESI (NVAR, COV, LDCOV, NF, A, LDA, Y, RESID, LDRESI)

Double: The double precision name is DFRESI.

Description

Routine FRESI computes the communalities and a standardized residual covariance/correlation matrix for input covariance/correlation matrix COV. The user must also input the orthogonal (unrotated) factor loadings, A, obtained from the matrix COV. Let a_i denote the i-th row of matrix A. Then, the communalities are given as

$$y_i = a_i a_i^T$$

where y_i is the i-th communality. The residual covariance/correlation matrix is given by

$$r_{ij} = s_{ij} - a_i a_j^T$$

where s_{ij} denotes an element of the covariance/correlation matrix and $R = (r_{ij})$ denotes the residual matrix. Standardization is performed by dividing the r_{ij} by

$$\sqrt{u_i u_j}$$

where $u_i = s_{ii} - y_i$ is the unique error variance for the i-th variable. If u_i is zero (or slightly less than zero due to roundoff error), $u_i = 1.0$ is assumed and division by zero is avoided.

Example

The following example computes the residual correlation matrix with communalities in a 9-factor problem. The resulting residual correlations do not seem to exhibit any pattern.

```
      USE FRESI_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDCOV, LDRESI, NF, NVAR
      PARAMETER  (LDA=9, LDCOV=9, LDRESI=9, NF=3, NVAR=9)
!
      REAL       A(9,3), COV(9,9), RESID(9,9), Y(9)
!
      DATA COV/1.000, 0.523, 0.395, 0.471, 0.346, 0.426, 0.576, 0.434, &
      0.639, 0.523, 1.000, 0.479, 0.506, 0.418, 0.462, 0.547, &
      0.283, 0.645, 0.395, 0.479, 1.000, 0.355, 0.270, 0.254, &
      0.452, 0.219, 0.504, 0.471, 0.506, 0.355, 1.000, 0.691, &
      0.791, 0.443, 0.285, 0.505, 0.346, 0.418, 0.270, 0.691, &
      1.000, 0.679, 0.383, 0.149, 0.409, 0.426, 0.462, 0.254, &
      0.791, 0.679, 1.000, 0.372, 0.314, 0.472, 0.576, 0.547, &
```

```
      0.452, 0.443, 0.383, 0.372, 1.000, 0.385, 0.680, 0.434, &
      0.283, 0.219, 0.285, 0.149, 0.314, 0.385, 1.000, 0.470, &
      0.639, 0.645, 0.504, 0.505, 0.409, 0.472, 0.680, 0.470, &
      1.000/
!
      DATA A/.6642, .6888, .4926, .8372, .7050, .8187, .6615, .4579, &
      .7657, -.3209, -.2471, -.3022, .2924, .3148, .3767, -.3960, &
      -.2955, -.4274, .0735, -.1933, -.2224, -.0354, -.1528, .1045, &
      -.0778, .4914, -.0117/
!
      CALL FRESI (COV, A, Y, RESID)
!
      CALL WRRRN ('Communalities', Y, 1, NVAR, 1)
      CALL WRRRN ('Residuals', RESID)
      END
```

Output

```
                          Communalities
     1        2        3        4        5        6        7        8
0.5495   0.5729   0.3834   0.7877   0.6195   0.8231   0.6005   0.5385

     9
0.7691

                                Residuals
         1       2       3       4       5       6       7       8       9
1    1.000   0.001  -0.024   0.037  -0.024  -0.016   0.036  -0.002  -0.018
2    0.001   1.000   0.043  -0.017  -0.048   0.041  -0.052  -0.023   0.031
3   -0.024   0.043   1.000   0.064  -0.033  -0.037  -0.022   0.025  -0.013
4    0.037  -0.017   0.064   1.000   0.012  -0.004   0.008   0.017  -0.052
5   -0.024  -0.048  -0.033   0.012   1.000  -0.003   0.075  -0.014   0.007
6   -0.016   0.041  -0.037  -0.004  -0.003   1.000  -0.046  -0.003   0.036
7    0.036  -0.052  -0.022   0.008   0.075  -0.046   1.000   0.008   0.011
8   -0.002  -0.023   0.025   0.017  -0.014  -0.003   0.008   1.000  -0.004
9   -0.018   0.031  -0.013  -0.052   0.007   0.036   0.011  -0.004   1.000
```

MVIND

more...

Computes a test for the independence of *k* sets of multivariate normal variables.

Required Arguments

NDF — Number of degrees of freedom in COV. (Input)

COV — NVAR by NVAR variance-covariance matrix. (Input)

NVSET — Index vector of length NGROUP. (Input)
NVSET(*i*) gives the number of variables in the *i*-th set of variables. The first NVSET(1) variables in COV define the first set of covariates, the next NVSET(2) variables define the second set of covariates, etc.

STAT — Vector of length 4 containing the output statistics. (Output)

I	STAT(I)
1	Statistic *V* for testing the hypothesis of independence of the NGROUP sets of variables.
2	Chi-squared statistic associated with *V*.
3	Degrees of freedom for STAT(2).
4	Probability of exceeding STAT(2) under the null hypothesis of independence.

Optional Arguments

NVAR — Number of variables in the covariance matrix. (Input)
Default: NVAR = size (COV,2).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

NGROUP — Number of sets of variables to be tested for independence. (Input)
Default: NGROUP = size (NVSET,1).

FORTRAN 90 Interface

Generic: CALL MVIND (NDF, COV, NVSET, STAT [, ...])
Specific: The specific interface names are S_MVIND and D_MVIND.

FORTRAN 77 Interface

Single: CALL MVIND (NDF, NVAR, COV, LDCOV, NGROUP, NVSET, STAT)
Double: The double precision name is DMVIND.

Description

Routine MVIND computes a likelihood ratio test statistic proposed by Wilks (1935) for testing the independence of NGROUP sets of multivariate normal variates. The likelihood ratio statistic is computed as the ratio of the determinant $|S|$ of the sample covariance matrix to the product of the determinants $|S_1|...|S_K|$ of the covariance matrices of each of the k = NGROUP sets of variates. An asymptotic chi-squared statistic obtained from the likelihood ratio, along with corresponding p-value, is computed according to formulas given by Morrison (1976, pages 258-259). The chi-squared statistic is computed as:

$$\chi^2 = -\frac{n}{C}\ln(V)$$

where n = NDF,

$$\begin{aligned} V &= \frac{|S|}{|S_{11}| \cdots |S_{kk}|} \\ C^{-1} &= 1 - \frac{2\sigma_2 + 3\sigma_3}{6n\sigma_2} \\ \sigma_2 &= \left(\sum_{i=1}^{k} p_i\right)^2 - \sum_{i=1}^{k} p_i^2 \\ \sigma_3 &= \left(\sum_{i=1}^{k} p_i\right)^3 - \sum_{i=1}^{k} p_i^3 \end{aligned}$$

where $|S_{ii}|$ is the determinant of the i-th covariance matrix, k = NGROUP, and p_i = NVSET(i), and $|S|$ is the determinant of COV.

Because determinants appear in both the numerator and denominator of the likelihood ratio, the test statistic is unchanged when correlation matrices are substituted for covariance matrices as input to **MVIND**.

In using **MVIND**, the covariance matrix must first be computed (possibly via routine **CORVC**, see Chapter 3, "Correlation"). The covariance matrix may then need to be rearranged (possible via routine **RORDM**) so that the **NVSET**(1) variables in the first set correspond to the first **NVSET**(1) columns (and rows) of the covariance matrix, with the next **NVSET**(2) columns and rows containing the variables for the second set of variables, etc. With this special arrangement of the covariance matrix, routine **MVIND** may then be called.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2IND/DM2IND**. The reference is:

   ```
   CALL M2IND (NDF, NVAR, COV, LD  COV, NGROUP, NVSET, STAT, FACT, WK, IPVT)
   ```

   The additional arguments are as follows:

   FACT — Work vector of length NVAR^2.

   WK — Work vector of length NVAR.

   IPVT — Work vector of length NVAR.

2. Informational errors

Type	Code	Description
4	1	A covariance matrix for a subset of the variables is singular.
4	2	The covariance matrix for all variables is singular.

Example

The example is taken from Morrison (1976, page 258). It involves two sets of covariates, with each set having two covariates. The null hypothesis of no relationship is rejected.

```
      USE MVIND_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LDCOV, NDF, NGROUP, NVAR
      PARAMETER  (NDF=932, NGROUP=2, NVAR=4, LDCOV=NVAR)
!
      INTEGER    NOUT, NVSET(NGROUP)
      REAL       COV(NVAR,NVAR), STAT(4)
!
      DATA COV/1.00, 0.45, -0.19, 0.43, 0.45, 1.00, -0.02, 0.62, &
     -0.19, -0.02, 1.00, -0.29, 0.43, 0.62, -0.29, 1.00/
!
      DATA NVSET/2, 2/
!
```

```
      CALL MVIND (NDF, COV, NVSET, STAT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Likelihood ratio ........... ', F12.4, /, ' ', &
       'Chi-squared ................ ', F9.1, /, ' Degrees of '&
      , 'freedom ......... ', F9.1, /, ' p-value ', &
         '..................... ', F12.4)
      END
```

Output

```
 Likelihood ratio ...........       0.5497
 Chi-squared ................     556.2
 Degrees of freedom .........       4.0
 p-value ....................       0.0000
```

CANCR

more...

Performs canonical correlation analysis from a data matrix.

Required Arguments

X — NOBS by NVAR1 + NVAR2 + *m* data matrix where *m* is 0, 1, or 2 depending on whether any columns of X correspond to frequencies or weights. (Input)
Each row of X contains an observation of the NVAR1 + NVAR2 variables for which canonical correlations are desired (plus a weight and/or a frequency variable if IFRQ and/or IWT(see below) are not zero). If both IWT and IFRQ are zero, *m* is 0; 1, if one of IFRQ or IWT is positive; and 2, otherwise. X may not have any missing values (NaN, not a number).

IND1 — Vector of length NVAR1 containing the column numbers in X of the group 1 variables. (Input)

IND2 — Vector of length NVAR2 containing the column numbers in X of the group 2 variables. (Input)

XX — NOBS by NVAR1 + NVAR2 + *m* matrix containing the canonical scores. (Output)
m is defined in the description for X. X and XX may occupy the same storage locations. Canonical scores are returned in the first NVAR1 + NVAR2 columns of XX. Scores for the NVAR1 variables come first. If one of IFRQ or IWT are not zero, then the last column of XX contains the weight or frequency. If both IFRQ and IWT are not zero, then the frequencies and weights are in the second to last and last column of XX, respectively.

CORR — NV by 6 matrix of output statistics. (Output)
NV is the minimum of NVAR1 and NVAR2. CORR has the following statistics.

Col.	Statistic
1	Canonical correlations sorted from the largest to the smallest.
2	Wilks' lambda for testing that the current and all smaller canonical correlations are zero.
3	Rao's *F* corresponding to Wilks' lambda. If the canonical correlation is greater than 0.99999, then *F* is set to 9999.99.

Col.	Statistic
4	Numerator degrees of freedom for *F*.
5	Denominator degrees of freedom for *F*.
6	Probability of a larger *F* statistic.

If an *F* statistic is negative, then CORR(*i*, 6) is set to one. If either CORR(*i*, 4) or CORR(*i*, 5) is not positive, then CORR(*i*, 6) is set to the missing value code (NaN).

COEF1 — NVAR1 by NVAR1 matrix containing the group 1 canonical coefficients. (Output)
The columns of COEF1 contain the vectors of canonical coefficients for group 1.

COEF2 — NVAR2 by NVAR2 matrix containing the group 2 canonical coefficients. (Output)
The columns of COEF2 contain the vectors of canonical coefficients for group 2.

COEFR1 — NVAR1 by NV matrix containing the correlations between the group 1 variables and the group 1 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

COEFR2 — NVAR2 by NV matrix containing the correlations between the group 2 variables and the group 2 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

STAT — 15 by NVAR1 + NVAR2 matrix containing statistics on all of the variables. (Output)
The first NVAR1 columns of STAT correspond to the group one variables with the last NVAR2 columns corresponding to the group two variables.

Row	Statistic
1	Means
2	Variances
3	Standard deviations
4	Coeffici ents of skewness
5	Coefficients of excess (kurtosis)
6	Minima
7	Maxima
8	Ranges
9	Coefficients of variation, when defined, 0.0 otherwise
10	Numbers of nonmissing observations
11	Lower endpoints of 95% confidence interval for the means
12	Upper endpoints of 95% confidence interval for the means

Row	Statistic
13	Lower endpoints of 95% confidence interval for the variances
14	Upper endpoints of 95% confidence interval for the variances
15	Sums of the weights if IWT greater than zero, 0.0 otherwise

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NVAR1 — Number of variables in group 1. (Input)
Default: NVAR1 = size (IND1,1).

NVAR2 — Number of variables in group 2. (Input)
Default: NVAR2 = size (IND2,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size(X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
If IFRQ = 0, then all frequencies are 1. If IFRQ is positive, then column number IFRQ of X contains the nonnegative frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
If IWT = 0, then there is no weighting, i.e., all weights are 1. If IWT is positive, then column number IWT of X contains the nonnegative weights.
Default: IWT = 0.

TOL — Constant used for determining linear dependence. (Input)
If the squared multiple correlation coefficient of a variable with its predecessors in IND1 (or IND2) is greater than 1 - TOL, then the variable is considered to be linearly dependent upon the previous variables; it is excluded from the analysis. TOL = .001 is a typical value. TOL must be in the exclusive range of 0.0 to 1.0.
Default: TOL = .001.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing.
1	Print CORR, COEF1, COEF2, COEFR1, COEFR2, and STAT.
2	Print all output.

LDXX — Leading dimension of XX exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXX = size (XX,1).

LDCORR — Leading dimension of CORR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCORR = size (CORR,1).

LDCOF1 — Leading dimension of COEF1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF1 = size (COEF1,1).

LDCOF2 — Leading dimension of COEF2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF2 = size (COEF2,1).

LDCFR1 — Leading dimension of COEFR1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR1 = size (COEFR1,1).

LDCFR2 — Leading dimension of COEFR2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR2 = size (COEFR2,1).

LDSTAT — Leading dimension of STAT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSTAT = size(STAT,1).

FORTRAN 90 Interface

Generic: CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, COEF2, COEFR1, COEFR2, STAT [, ...])

Specific: The specific interface names are S_CANCR and D_CANCR.

FORTRAN 77 Interface

Single: CALL CANCR (NOBS, NVAR1, NVAR2, NCOL, X, LDX, IFRQ, IWT, IND1, IND2, TOL, IPRINT, XX, LDXX, CORR, LDCORR, COEF1, LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1, COEFR2, LDCFR2, STAT, LDSTAT)

Double: The double precision name is DCANCR.

Description

Routine CANCR computes the canonical correlations, the canonical coefficients, the canonical scores, Wilks' lambda for testing the independence of two sets of variates, and a series of Bartlett's tests of the hypothesis that the *k*-th largest and all larger canonical correlations are simultaneously zero. A matrix of observations is used in these computations.

Let x_{ij} denote the *j*-th variable on the *i*-th observation, w_i denote the observation weight, f_i denote the observation frequency, Γ_{11} denote the upper triangular Cholesky ($R^T R$) factorization of the sample covariance matrix of the group 1 variables, Γ_{22} denote the upper triangular Cholesky ($R^T R$) factorization of the group 2 variables sample covariance matrix, and

$$\Gamma_{12} = \left(\Gamma_{11}\right)^{-1} \hat{\Sigma}_{12} \left(\Gamma_{22}{}^{T}\right)^{-1}$$

where

$$\hat{\Sigma}_{12}$$

is the sample estimate of the matrix of covariances between the group 1 and the group 2 variables. Then, the computational procedure in obtaining the canonical correlations is as follows:

1. The weighted mean of each variable is computed via the standard formula (see UVSTA, Chapter 1, "Basic Statistics"). The means are then subtracted from the observations.
2. Each element in the *i*-th row of **X** is multiplied by

$$\sqrt{\left(w_i f_i\right)}$$

3. Gram-Schmidt orthogonalization is used on **X** to obtain Y_1 and Y_2, where Y_1 and Y_2 are the results of the Gram-Schmidt orthogonalization of the group 1 and the group 2 variables, respectively. The matrices $\Gamma_{11}$and Γ_{22} are obtained as a by-product of the orthogonalization. Compute

$$\Gamma_{12} = Y_1^T Y_2$$

4. The canonical correlations are obtained as the singular values of the matrix Γ_{12}. Denote the left and right orthogonal matrices obtained as a by-product of this decomposition by *L* and *R*, respectively.

5. The canonical coefficients are obtained from *L* and *R* by multiplying *L* and *R* by the inverses of $\Gamma_{11}$and Γ_{22}, respectively (see Golub 1969).

6. The correlations of the original variables with the canonical variables are obtained by multiplying *L* and *R* by $\Gamma_{11}$and Γ_{22}, respectively.

7. The canonical scores are obtained by multiplying the matrices Y_1 and Y_2 by the matrices *L* and *R*, respectively, and then dividing each row of Y_1 and Y_2 by

$$\sqrt{\left(w_i f_i\right)}$$

8. Wilks' lambda, the Bartlett's tests, Rao's *F* corresponding to these tests, the numerator and denominator degrees of freedom of *F*, and the significance level of *F* are computed as in Rao (1973, page 556). Bartlett's tests are computed as

$$\Lambda_i = \prod_{j=i}^{q} (1 - \rho_j^2)$$

where q = **NVAR2** is the number of canonical correlations, the canonical correlations are ordered from largest to smallest, and ρ_j denotes the *j*-th largest canonical correlation. Wilks' lambda is given as Λ_1. The degrees of freedom in the numerator of the corresponding Rao's *F* statistic is given as

$$d_1 = pu$$

where $p = v_1 - i + 1$, $u = v_2 - i + 1$, v_1 = **NVAR2**, and v_2 = **NVAR1**. Let

$$m = t - \frac{p + u + 1}{2}$$

where *t* is the degrees of freedom in **COV**($\Sigma_i f_i$ - 1), and let

$$s = \sqrt{\frac{p^2u^2 - 4}{p^2 + u^2 - 5}}$$

if $p^2 + u^2 - 5 \neq 0$, and let $s = 2$ otherwise. Then, Rao's *F* corresponding to Bartlett's test is computed as

$$F_i = \frac{1 - \Lambda_i^{\frac{1}{s}}}{\Lambda_i^{\frac{1}{s}}}(ms - pu/2 + 1)/pu$$

Rao's *F* has numerator degrees of freedom $d_2 = ms - pu/2 + 1$. The significance level of *F* is obtained from the standard *F* distribution.

Comments

1. Workspace may be explicitly provided, if desired, by use of `C2NCR/DC2NCR`. The reference is:

```
CALL C2NCR (NOBS, NVAR1, NVAR2, NCOL, X, LDX, IFRQ, IWT, IND1, IND2, TOL, IPRINT,
    XX, LDXX, CORR, LDCORR, COEF1, LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1,
    COEFR2, LDCFR2, STAT, LDSTAT, R, S, IND, WORK, WKA, WK)
```

The additional arguments are as follows:

R — Work vector of length NVAR1^2.

S — Work vector of length NVAR2^2.

IND — Work vector of length NVAR1 + NVAR2 + 2.

WORK — Work vector of length max(NOBS, 2 * (NVAR1 + NVAR2))

WKA — Work vector of length (max (NVAR1, NVAR2))^2.

WK — Work vector of length 3 * max(NVAR1, NVAR2) - 1.

2. Informational errors

Type	Code	Description
3	1	The standardized cross covariance matrix is not of full rank or is very ill-conditioned. Small canonical correlations may not be accurate.
3	2	One or more variables is linearly dependent upon the proceeding variables in its group.
4	3	The sum of the frequencies is equal to zero. The sum of the frequencies must be positive.
4	4	The sum of the weights is equal to zero. The sum of the weights must be positive.

Examples

Example 1

The following example is taken from Levin and Marascuilo (1983), pages 191–197. It is examining the relationship between the performance of individuals in a sociology course and predictor variables. The measures of performance in the sociology course are two midterms examinations, a final examination, and a course evaluation, the predictor variables are social class, sex, grade point average, college board test score, whether the student has previously taken a course in sociology, and the student's score on a pretest.

```
      USE WRRRL_INT
      USE CANCR_INT
      IMPLICIT NONE

      INTEGER    IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, LDCORR, LDSTAT,&
           LDX, LDXX, NCOL, NOBS, NV, NVAR1, NVAR2, I
      REAL       TOL
      PARAMETER  (IPRINT=1, LDSTAT=15, NCOL=10, NOBS=40, NVAR1=6, &
                  NVAR2=4, TOL=0.0001, LDCFR1=NVAR1, LDCFR2=NVAR2, &
                  LDCOF1=NVAR1, LDCOF2=NVAR2, LDX=NOBS, LDXX=NOBS, &
                  NV=NVAR2, LDCORR=NV)
!
      INTEGER    IND1(NVAR1), IND2(NVAR2)
      REAL       COEF1(LDCOF1,NVAR1), COEF2(LDCOF2,NVAR2), &
           COEFR1(LDCFR1,NV), COEFR2(LDCFR2,NV), &
           CORR(LDCORR,6),STAT(LDSTAT,NVAR1+NVAR2), &
           X(LDX,NCOL), XX(LDXX,NCOL)
      CHARACTER  FMT*35, NUMBER(1)*6, XLAB(11)*25
!
      DATA IND1/1, 2, 3, 4, 5, 6/, IND2/7, 8, 9, 10/
      DATA (X(I,1),I=1,NOBS)/3*2.0, 3.0, 2.0, 3.0, 1.0, 2.0, 3.0, &
      2*2.0, 3.0, 1.0, 4*2.0, 3.0, 3*2.0, 1.0, 3*2.0, 1.0, 2.0, &
      1.0, 2.0, 3.0, 2*2.0, 2*1.0, 2.0, 3.0, 1.0, 2.0, 3.0, 1.0/
      DATA (X(I,2),I=1,NOBS)/6*1.0, 0.0, 2*1.0, 3*0.0, 3*1.0, 3*0.0, &
      1.0, 0.0, 3*1.0, 3*0.0, 4*1.0, 0.0, 8*1.0, 0.0/
      DATA (X(I,3),I=1,NOBS)/3.55, 2.70, 3.50, 2.91, 3.10, 3.49, 3.17, &
      3.57, 3.76, 3.81, 3.60, 3.10, 3.08, 3.50, 3.43, 3.39, 3.76, &
      3.71, 3.00, 3.47, 3.69, 3.24, 3.46, 3.39, 3.90, 2.76, 2.70, &
```

```
     3.77, 4.00, 3.40, 3.09, 3.80, 3.28, 3.70, 3.42, 3.09, 3.70, &
     2.69, 3.40, 2.95/
     DATA (X(I,4),I=1,NOBS)/410.0, 390.0, 510.0, 430.0, 600.0, &
         2*610.0, 560.0, 700.0, 460.0, 590.0, 500.0, 410.0, 470.0, &
     210.0, 610.0, 510.0, 600.0, 470.0, 460.0, 800.0, 610.0, &
     490.0, 470.0, 610.0, 580.0, 410.0, 630.0, 790.0, 490.0, &
     400.0, 2*610.0, 500.0, 430.0, 540.0, 610.0, 400.0, 390.0, &
     490.0/
     DATA (X(I,5),I=1,NOBS)/8*0.0, 4*1.0, 0.0, 2*1.0, 0.0, 1.0, 0.0, &
     1.0, 0.0, 1.0, 3*0.0, 1.0, 2*0.0, 2*1.0, 2*0.0, 4*1.0, &
     5*0.0/
     DATA (X(I,6),I=1,NOBS)/17.0, 20.0, 22.0, 13.0, 16.0, 28.0, 14.0, &
     10.0, 28.0, 30.0, 28.0, 15.0, 24.0, 15.0, 26.0, 16.0, 25.0, &
     3.0, 5.0, 16.0, 28.0, 13.0, 9.0, 13.0, 30.0, 10.0, 13.0, &
     8.0, 29.0, 17.0, 15.0, 16.0, 13.0, 30.0, 2*17.0, 25.0, &
     10.0, 23.0, 18.0/
     DATA (X(I,7),I=1,NOBS)/43.0, 50.0, 47.0, 24.0, 47.0, 57.0, &
     2*42.0, 69.0, 48.0, 59.0, 21.0, 52.0, 2*35.0, 59.0, 68.0, &
     38.0, 45.0, 37.0, 54.0, 45.0, 31.0, 39.0, 67.0, 30.0, 19.0, &
     71.0, 80.0, 47.0, 46.0, 59.0, 48.0, 68.0, 43.0, 31.0, 64.0, &
     19.0, 43.0, 20.0/
     DATA (X(I,8),I=1,NOBS)/61.0, 47.0, 79.0, 40.0, 60.0, 59.0, 61.0, &
     79.0, 83.0, 67.0, 74.0, 40.0, 71.0, 40.0, 57.0, 58.0, 66.0, &
     58.0, 24.0, 48.0, 100.0, 83.0, 70.0, 48.0, 85.0, 14.0, &
     55.0, 100.0, 94.0, 45.0, 58.0, 90.0, 84.0, 81.0, 49.0, &
     54.0, 87.0, 36.0, 51.0, 59.0/
     DATA (X(I,9),I=1,NOBS)/129.0, 60.0, 119.0, 100.0, 79.0, 99.0, &
     92.0, 107.0, 156.0, 110.0, 116.0, 49.0, 107.0, 125.0, 64.0, &
     100.0, 138.0, 63.0, 82.0, 73.0, 132.0, 87.0, 89.0, 99.0, &
     119.0, 100.0, 84.0, 166.0, 111.0, 110.0, 93.0, 141.0, 99.0, &
     114.0, 96.0, 39.0, 149.0, 53.0, 39.0, 91.0/
     DATA (X(I,10),I=1,NOBS)/3.0, 3*1.0, 2.0, 1.0, 3.0, 2.0, 4*1.0, &
     5.0, 1.0, 5.0, 1.0, 2.0, 1.0, 2*3.0, 3*2.0, 1.0, 2.0, 1.0, &
     2.0, 3.0, 2.0, 2*1.0, 2*2.0, 5.0, 2*1.0, 4.0, 3.0, 2*1.0/
!
     DATA XLAB/' ','Social%/Class', '%/Sex', '%/GPA', &
     'College%/Boards', 'H.S.%/Soc.', 'Pretest%/Score', &
     '%/Exam 1', '%/Exam 2', 'Final%/Exam', 'Course%/Eval.'/
     DATA NUMBER/'NUMBER'/, FMT/'(2W3.1,W5.3,W4.1,W3.1,4W5.1,W3.1)'/
!
     CALL WRRRL ('First 10 Observations', X, NUMBER, XLAB, &
                 10, NCOL, LDX, FMT=FMT)
!
     CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, &
     COEF2, COEFR1, COEFR2, STAT, TOL=TOL, IPRINT=IPRINT)
!
     END
```

Output

```
                          First 10 Observations
    Social                College  H.S.  Pretest                Final Course
     Class    Sex   GPA   Boards  Soc.    Score  Exam 1  Exam 2  Exam   Eval
1        2    1   3.55      410    0       17      43      61    129      3
2        2    1   2.70      390    0       20      50      47     60      1
3        2    1   3.50      510    0       22      47      79    119      1
4        3    1   2.91      430    0       13      24      40    100      1
5        2    1   3.10      600    0       16      47      60     79      2
6        3    1   3.49      610    0       28      57      59     99      1
```

7	1	0	3.17	610	0	14	42	61	92	3
8	2	1	3.57	560	0	10	42	79	107	2
9	3	1	3.76	700	1	28	69	83	156	1
10	2	0	3.81	460	1	30	48	67	110	1

*** Canonical Correlations Statistics ***

	Canonical Correlations	Wilks Lambda	Raos F	Num. df	Denom. df	Prob. of Larger F
1	0.9242	0.0612	5.412	24	105.9	0.0000
2	0.7184	0.4201	2.116	15	86.0	0.0162
3	0.2893	0.8683	0.586	8	64.0	0.7861
4	0.2290	0.9476	0.609	3	33.0	0.6142

Group One Canonical Coefficients

	1	2	3	4	5	6
1	-0.622	1.158	-0.285	-0.179	0.601	-0.423
2	0.558	-0.739	0.231	-1.278	1.391	-0.024
3	1.796	-0.432	0.765	0.185	-0.643	-3.314
4	0.002	0.006	0.004	-0.002	0.000	0.006
5	-0.059	-0.043	-0.456	1.671	1.463	0.774
6	0.031	0.018	-0.121	-0.058	-0.042	0.056

Group Two Canonical Coefficients

	1	2	3	4
1	0.0233	-0.0365	0.0845	-0.0176
2	0.0257	-0.0057	-0.0352	0.0555
3	0.0073	0.0110	-0.0259	-0.0341
4	0.1034	0.8089	0.2828	0.0260

Correlations Between the Group One Variables and the Group One Canonical Scores

	1	2	3	4
1	-0.3685	0.6795	-0.2291	-0.1854
2	0.2157	-0.3252	0.0521	-0.5985
3	0.8153	0.2770	-0.0692	0.2123
4	0.6144	0.5681	0.4151	-0.0050
5	0.4661	0.0603	-0.3034	0.6530
6	0.5461	0.1768	-0.7915	-0.1375

Correlations Between the Group Two Variables and the Group Two Canonical Scores

	1	2	3	4
1	0.8713	-0.2406	0.3864	-0.1835
2	0.9174	-0.0557	-0.2068	0.3355
3	0.7707	0.0293	-0.3146	-0.5533
4	0.3490	0.8765	0.3077	0.1240

*** Statistics for Group One Variables ***
Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	1.9750	0.4353	0.6597	0.02476	-0.6452
2	0.6750	0.2250	0.4743	-0.74726	-1.4416
3	3.3758	0.1247	0.3532	-0.37911	-0.7521
4	524.2499	13148.1377	114.6653	0.09897	0.6494
5	0.4000	0.2462	0.4961	0.40825	-1.8333
6	18.1250	55.1378	7.4255	0.10633	-0.9358

Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	1.0000	3.0000	2.0000	0.3340	40.0000

2	0.0000	1.0000	1.0000	0.7027	40.0000
3	2.6900	4.0000	1.3100	0.1046	40.0000
4	210.0000	800.0000	590.0000	0.2187	40.0000
5	0.0000	1.0000	1.0000	1.2403	40.0000
6	3.0000	30.0000	27.0000	0.4097	40.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	1.7640	2.1860	0.29207	0.7176
2	0.5233	0.8267	0.15098	0.3710
3	3.2628	3.4887	0.08369	0.2056
4	487.5782	560.9217	8822.72168	21677.9590
5	0.2413	0.5587	0.16518	0.4058
6	15.7502	20.4998	36.99883	90.9083

*** Statistics for Group Two Variables ***
Univariate Statistics from UVSTA

Variable	Mean	Variance	Std. Dev.	Skewness	Kurtosis
1	46.0500	237.0231	15.3956	0.08762	-0.5505
2	62.8750	403.4967	20.0872	-0.10762	-0.3642
3	99.4750	919.4864	30.3230	-0.03483	-0.2533
4	1.9500	1.4333	1.1972	1.27704	0.8407

Variable	Minimum	Maximum	Range	Coef. Var.	Count
1	19.0000	80.0000	61.0000	0.3343	40.0000
2	14.0000	100.0000	86.0000	0.3195	40.0000
3	39.0000	166.0000	127.0000	0.3048	40.0000
4	1.0000	5.0000	4.0000	0.6140	40.0000

Variable	Lower CLM	Upper CLM	Lower CLV	Upper CLV
1	41.1263	50.9737	159.0483	390.7912
2	56.4508	69.2992	270.7562	665.2642
3	89.7772	109.1728	616.9979	1516.0009
4	1.5671	2.3329	0.9618	2.3632

Example 2

Correspondence analysis is an interesting application of canonical correlation in the analysis of contingency tables. The example is taken from Kendall and Stuart (1979, pages 595–599) and involves finding the optimal scores for the values of two categorical variables to maximize the correlation between the two variables. The contingency table is given below, along with the more traditional matrix X of "observations" for which canonical correlations are desired.

$$\begin{pmatrix} 821 & 112 & 85 & 35 \\ 116 & 494 & 145 & 27 \\ 72 & 151 & 583 & 87 \\ 43 & 34 & 106 & 331 \end{pmatrix}$$

The data matrix **X** is given as:

Group 1 Var.				Group 2 Var.				Frequencies
1	0	0	0	1	0	0	0	821
1	0	0	0	0	1	0	0	112
1	0	0	0	0	0	1	0	85
1	0	0	0	0	0	0	1	35
0	1	0	0	1	0	0	0	116
0	1	0	0	0	1	0	0	494
0	1	0	0	0	0	1	0	145
0	1	0	0	0	0	0	1	27
0	0	1	0	1	0	0	0	72
0	0	1	0	0	1	0	0	151
0	0	1	0	0	0	1	0	583
0	0	1	0	0	0	0	1	87
0	0	0	1	1	0	0	0	43
0	0	0	1	0	1	0	0	34
0	0	0	1	0	0	1	0	106
0	0	0	1	0	0	0	1	331

For this table, the optimal correlation turns out to be 0.70 when scores of 2.67, 1.34, 0.62, and 0.00 (see Column 1 of **COEF1**) are assigned to the variable 1 categories, and scores of 2.72, 1.37, 0.68, and 0.00 are assigned to the variable 2 categories. These scores are obtained as the canonical scores when canonical correlations are computed between the the row and column variable indicator variables (variables 1-4 and variables 5-8 in **X**, respectively). The warning error appears in the output because the covariance matrix is not of full rank (indeed, neither the group 1 or the group 2 covariance matrices are of full rank).

```
          USE CANCR_INT

          IMPLICIT   NONE
          INTEGER    IFRQ, IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, &
                     LDCORR, LDSTAT, LDX, LDXX, NCOL, NOBS, NV, NVAR1, &
                     NVAR2
          REAL       TOL
          PARAMETER  (IFRQ=9, IPRINT=2, LDCFR1=4, LDCFR2=4, &
                     LDCOF1=4, LDCOF2=4, LDCORR=4, LDSTAT=15, LDX=16, &
                     LDXX=16, NCOL=9, NOBS=16, NV=4, NVAR1=4, NVAR2=4, &
                     TOL=0.0001)
    !
          INTEGER    IND1(NVAR1), IND2(NVAR2)
          REAL       COEF1(LDCOF1,NVAR1), COEF2(LDCOF2,NVAR2), &
                     COEFR1(LDCFR1,NV), COEFR2(LDCFR2,NV), CORR(LDCORR,6), &
```

```
                  STAT(LDSTAT,8), X(LDX,NCOL), XX(LDXX,NCOL)
!
      DATA IND1/1, 2, 3, 4/, IND2/5, 6, 7, 8/
      DATA X/4*1.0, 16*0.0, 4*1.0, 16*0.0, 4*1.0, 16*0.0, 5*1.0, &
          3*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, &
          3*0.0, 1.0, 3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, &
          3*0.0, 1.0, 4*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, 3*0.0, 1.0, &
          821.0, 112.0, 85.0, 35.0, 116.0, 494.0, 145.0, 27.0, 72.0, &
          151.0, 583.0, 87.0, 43.0, 34.0, 106.0, 331.0/
!
      CALL CANCR (X, IND1, IND2, XX, CORR, COEF1, &
                  COEF2, COEFR1, COEFR2, STAT, IFRQ=IFRQ, &
                  TOL=TOL, IPRINT=IPRINT)
!
      END
```

Output

```
*** WARNING  ERROR 2 from C2NCR.  One or more Group 1 variables is linearly
***          dependent on the proceeding variables in Group 1.
Here is a traceback of subprogram calls in reverse order:
Routine name                    Error type  Error code
------------                    ----------  ----------
C2NCR                               6           2    (Called internally)
CANCR                               0           0
USER                                0           0
*** WARNING  ERROR 3 from C2NCR.  One or more Group 2 variables is linearly
***          dependent on the proceeding variables in Group 2.
Here is a traceback of subprogram calls in reverse order:
Routine name                    Error type  Error code
------------                    ----------  ----------
C2NCR                               6           3    (Called internally)
CANCR                               0           0
USER                                0           0

          *** Canonical Correlations Statistics ***
      Canonical                                                  Prob. of
   Correlations  Wilks Lambda      Raos F  Num. df  Denom. df  Larger F
1        0.6965        0.2734     615.925        9     7875.7    0.0000
2        0.5883        0.5310     602.598        4     6474.0    0.0000
3        0.4336        0.8120     749.823        1     3238.0    0.0000
4        0.0000        0.0000       0.000        0        0.0    0.0000

Group One Canonical Coefficients
         1       2       3       4
1   2.670   1.100   1.023   0.000
2   1.341   2.905  -0.460   0.000
3   0.624   2.222   2.147   0.000
4   0.000   0.000   0.000   0.000

Group Two Canonical Coefficients
         1       2       3       4
1   2.715   1.164   1.053   0.000
2   1.366   2.972  -0.393   0.000
3   0.676   2.250   2.182   0.000
4   0.000   0.000   0.000   0.000

Correlations Between the Group One Variables
     and the Group One Canonical Scores
```

```
          1        2        3        4
1   0.9068  -0.3954   0.1459   0.0000
2  -0.0121   0.6965  -0.7175   0.0000
3  -0.4555   0.3404   0.8226   0.0000
4   0.0000   0.0000   0.0000   0.0000

Correlations Between the Group Two Variables
     and the Group Two Canonical Scores
          1        2        3        4
1   0.9072  -0.3997   0.1310   0.0000
2  -0.0227   0.6995  -0.7143   0.0000
3  -0.4590   0.3205   0.8287   0.0000
4   0.0000   0.0000   0.0000   0.0000

                  *** Statistics for Group One Variables ***
                      Univariate Statistics from UVSTA

Variable          Mean      Variance    Std. Dev.     Skewness     Kurtosis
    1           0.3248        0.2194       0.4684       0.7482      -1.4401
    2           0.2412        0.1831       0.4279       1.2098      -0.5363
    3           0.2754        0.1996       0.4468       1.0053      -0.9894
    4           0.1585        0.0000       0.0000       1.8697       1.4958
Variable       Minimum       Maximum        Range   Coef. Var.        Count
    1           0.0000        1.0000       1.0000       1.4420    3242.0000
    2           0.0000        1.0000       1.0000       1.7739    3242.0000
    3           0.0000        1.0000       1.0000       1.6221    3242.0000
    4           0.0000        1.0000       1.0000       2.3041    3242.0000

Variable     Lower CLM     Upper CLM     Lower CLV     Upper CLV
    1           0.3087        0.3409        0.2091        0.2305
    2           0.2265        0.2559        0.1745        0.1923
    3           0.2601        0.2908        0.1903        0.2097
    4           0.1460        0.1711        0.1272        0.1402

Canonical Scores for Group One
         1       2       3       4
 1   1.307  -0.570   0.210   0.000
 2   1.307  -0.570   0.210   0.000
 3   1.307  -0.570   0.210   0.000
 4   1.307  -0.570   0.210   0.000
 5  -0.021   1.235  -1.272   0.000
 6  -0.021   1.235  -1.272   0.000
 7  -0.021   1.235  -1.272   0.000
 8  -0.021   1.235  -1.272   0.000
 9  -0.739   0.552   1.334   0.000
10  -0.739   0.552   1.334   0.000
11  -0.739   0.552   1.334   0.000
12  -0.739   0.552   1.334   0.000
13  -1.362  -1.670  -0.813   0.000
14  -1.362  -1.670  -0.813   0.000
15  -1.362  -1.670  -0.813   0.000
16  -1.362  -1.670  -0.813   0.000

                  *** Statistics for Group Two Variables ***
                      Univariate Statistics from UVSTA

Variable          Mean      Variance     Std. Dev.     Skewness     Kurtosis
    1           0.3245        0.2193       0.4683       0.7497      -1.4379
    2           0.2440        0.1845       0.4296       1.1922      -0.5787
    3           0.2835        0.2032       0.4508       0.9609      -1.0766
```

```
       4         0.1481         0.0000         0.0000         1.9819         1.9280

   Variable      Minimum        Maximum          Range     Coef. Var.         Count
       1          0.0000         1.0000         1.0000         1.4430     3242.0000
       2          0.0000         1.0000         1.0000         1.7606     3242.0000
       3          0.0000         1.0000         1.0000         1.5901     3242.0000
       4          0.0000         1.0000         1.0000         2.3992     3242.0000

   Variable     Lower CLM     Upper CLM     Lower CLV     Upper CLV
       1           0.3084        0.3406        0.2090        0.2303
       2           0.2292        0.2588        0.1758        0.1938
       3           0.2679        0.2990        0.1936        0.2134
       4           0.1358        0.1603        0.1203        0.1326

     Canonical Scores for Group Two
            1       2       3       4
    1   1.309  -0.577   0.189   0.000
    2  -0.040   1.231  -1.257   0.000
    3  -0.730   0.509   1.317   0.000
    4  -1.406  -1.740  -0.864   0.000
    5   1.309  -0.577   0.189   0.000
    6  -0.040   1.231  -1.257   0.000
    7  -0.730   0.509   1.317   0.000
    8  -1.406  -1.740  -0.864   0.000
    9   1.309  -0.577   0.189   0.000
   10  -0.040   1.231  -1.257   0.000
   11  -0.730   0.509   1.317   0.000
   12  -1.406  -1.740  -0.864   0.000
   13   1.309  -0.577   0.189   0.000
   14  -0.040   1.231  -1.257   0.000
   15  -0.730   0.509   1.317   0.000
   16  -1.406  -1.740  -0.864   0.000

   *** WARNING  ERROR 1 from CANCR.  The standardized cross covariance matrix
   ***          is not of full rank or is very ill-conditioned.  Small
   ***          canonical correlations may not be accurate.
```

CANVC

more...

Performs canonical correlation analysis from a variance-covariance matrix or a correlation matrix.

Required Arguments

NDF — Number of degrees of freedom in the covariance or correlation matrix. (Input)
If NDF is unknown, an estimate of NDF = 100 is suggested in which case the last four columns of CORR are meaningless.

COV — NVAR1 + NVAR2 by NVAR1 + NVAR2 matrix containing the covariance or correlation matrix. (Input)
Routines COVPL, RBCOV, or CORVC (see Chapter 3, "Correlation") may be used to calculate COV from a data matrix. COV must be nonnegative definite within a tolerance of 100.0 * AMACH(4). Only the upper triangle of COV is referenced.

IND1 — Vector of length NVAR1 containing the column and row numbers in COV for the group 1 variables. (Input)

IND2 — Vector of length NVAR2 containing the column and row numbers in COV for the group 2 variables. (Input)

CORR — NV by 6 matrix containing the output statistics. (Output)
NV is the minimum of NVAR1 and NVAR2.

Col	Statistic
1	Canonical correlations sorted from the largest to the smallest.
2	Wilks' lambda for testing that the current and all smaller canonical correlations are zero.
3	Rao's *F* corresponding to Wilks' lambda. If the canonical correlation is greater than 0.99999, *F* is set to 9999.99.
4	Numerator degrees of freedom for the *F*.

Col	Statistic
5	Denominator degrees of freedom for the *F*.
6	Probability of a larger *F* statistic.

If an *F* statistic is negative, then CORR(*i*, 6) is set to one. If either CORR(*i*, 4) or CORR(*i*, 5) is not positive, then CORR(*i*, 6) is set to the missing value code (NaN).

COEF1 — NVAR1 by NVAR1 matrix containing the group 1 canonical coefficients. (Output)
The columns of COEF1 contain the vectors of canonical coefficients for group 1.

COEF2 — NVAR2 by NVAR2 matrix containing the group 2 canonical coefficients. (Output)
The columns of COEF2 contain the vectors of canonical coefficients for group 2.

COEFR1 — NVAR1 by NV matrix containing the correlations between the group 1 variables and the group 1 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

COEFR2 — NVAR2 by NV matrix containing the correlations between the group 2 variables and the group 2 canonical scores. (Output)
NV is the minimum of NVAR1 and NVAR2.

Optional Arguments

NVAR1 — Number of variables in group 1. (Input)
Default: NVAR1 = size (IND1,1).

NVAR2 — Number of variables in group 2. (Input)
Default: NVAR2 = size (IND2,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing.
1	Printing of CORR, COEF1, COEF2, COEFR1, and COEFR2 is performed.

LDCORR — Leading dimension of CORR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCORR = size (CORR,1).

LDCOF1 — Leading dimension of COEF1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF1 = size (COEF1,1).

LDCOF2 — Leading dimension of COEF2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOF2 = size (COEF2,1).

LDCFR1 — Leading dimension of COEFR1 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR1 = size (COEFR1,1).

LDCFR2 — Leading dimension of COEFR2 exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFR2 = size (COEFR2,1).

FORTRAN 90 Interface

Generic: CALL CANVC (NDF, COV, IND1, IND2, CORR, COEF1, COEF2, COEFR1, COEFR2 [, ...])

Specific: The specific interface names are S_CANVC and D_CANVC.

FORTRAN 77 Interface

Single: CALL CANVC (NDF, NVAR1, NVAR2, COV, LDCOV, IND1, IND2, IPRINT, CORR, LDCORR, COEF1, LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1, COEFR2, LDCFR2)

Double: The double precision name is DCANVC.

Description

Routine CANVC computes the canonical correlations, the canonical coefficients, Wilks' lambda (for testing the independence of two sets of variates), and a series of tests due to Bartlett for testing that all canonical correlations greater than or equal to the *k*-th largest are simultaneously zero. The covariance matrix is used in these computations.

The group 1 variables covariance matrix is first extracted from COV and placed in the matrix S_{11}. Similarly, the group 2 variables covariance matrix is placed in S_{22}. The "standardized" cross covariance matrix is then computed as:

$$C = \left(S_{11}^{-\frac{1}{2}} \right)^T S_{12} \left(S_{22}^{-\frac{1}{2}} \right)$$

where S_{12} is the **NVAR1** × **NVAR2** matrix of covariances between the group 1 and group 2 variables, and $S^{1/2}$denotes the upper triangular Cholesky ($R^T R$) factorization of S. In the computation of C and in the following, it is assumed that **NVAR1** is greater than **NVAR2**. The group 1 and group 2 variables should be interchanged in the following if this is not the case.

The canonical correlations are computed as the singular values of the matrix C. The canonical coefficients are obtained from the left and right orthogonal matrices resulting from the singular value decomposition of C. In particular, for Γ_1 = **COEF1**.

$$\Gamma_1 = \left(S_{11}^{-\frac{1}{2}} \right) L$$

where L is the left orthogonal matrix from the singular value decomposition.

Similarly, the correlations between the original variables and the canonical variables, R_1 = **COEFR1**, are obtained for the group 1 variables as:

$$R_1 = \Delta_{11}^{-\frac{1}{2}} \left(S_{11}^{\frac{1}{2}} \right)^T L$$

where Δ_{11} is a diagonal matrix containing the diagonal of S_{11} along its diagonal.

Wilks' lambda, the Bartlett's tests, Rao's F corresponding to these tests, the numerator and denominator degrees of freedom of F , and the significance level of F are computed as in Rao (1973, page 556). Bartlett's tests are computed as

$$\Lambda_i = \prod_{j=i}^{q} \left(1 - \rho_j^2 \right)$$

where q = **NVAR2** is the number of canonical correlations, the canonical correlations are ordered from largest to smallest, and ρ_j denotes the j-th largest canonical correlation. Wilks' lambda is given as Λ_1. The degrees of freedom in the numerator of the corresponding Rao's F statistic is given as

$$d_1 = pu$$

where $p = v_1 - i + 1$, $u = v_2 - i + 1$, v_1 = **NVAR2**, and v_2 = **NVAR1**. Let

$$m = t - \frac{p + u + 1}{2}$$

where *t* is the degrees of freedom in **COV**, and let

$$s = \sqrt{\frac{p^2u^2 - 4}{p^2 + u^2 - 5}}$$

if $p^2 + u^2 - 5 \neq 0$, and let $s = 2$ otherwise. Then, Rao's *F* corresponding to Bartlett's test is computed as

$$F_i = \frac{1 - \Lambda_i^{\frac{1}{s}}}{\Lambda_i^{\frac{1}{s}}} \frac{ms - pu/2 + 1}{pu}$$

Rao's *F* has numerator degrees of freedom $d_2 = ms - pu/2 + 1$. The significance level of *F* is obtained from the standard *F* distribution

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2NVC/DC2NVC**. The reference is:

   ```
   CALL C2NVC (NDF, NVAR1, NVAR2, COV, LDCOV, IND1, IND2, IPRINT, CORR, LDCORR,
      COEF1, LDCOF1, COEF2, LDCOF2, COEFR1, LDCFR1, COEFR2, LDCFR2, R, S, STD1,
      STD2, WKA, WK)
   ```

   The additional arguments are as follows:

   R — Work vector of length **NVAR1**^{2}.

   S — Work vector of length **NVAR2**^{2}.

   STD1 — Work vector of length **NVAR1**.

   STD2 — Work vector of length **NVAR2**.

   WKA — Work vector of length (**NVAR1** + **NVAR2**)^2.

   WK — Work vector of length 3 * max(**NVAR1**, **NVAR2**).

2. Informational errors

Type	Code	Description
3	1	The standardized cross covariance matrix is not of full rank or is very ill-conditioned. Small canonical correlations may not be accurate.
4	2	COV is not nonnegative definite.

Example

The following example is taken from Van de Geer (1971). There are six group 1 variables and two group 2 variables. The maximum correlation turns out to be 0.609.

```
      USE CANVC_INT

      IMPLICIT   NONE
      INTEGER    IPRINT, LDCFR1, LDCFR2, LDCOF1, LDCOF2, LDCORR, &
                 LDCOV, NDF, NV, NVAR1, NVAR2
      PARAMETER  (IPRINT=1, LDCFR1=6, LDCFR2=2, LDCOF1=6, LDCOF2=2, &
                 LDCORR=2, LDCOV=8, NDF=100, NV=2, NVAR1=6, NVAR2=2)
!
      INTEGER    IND1(NVAR1), IND2(NVAR2)
      REAL       COEF1(NVAR1,NVAR1), COEF2(NVAR2,NVAR2), &
                 COEFR1(NVAR1,NVAR2), COEFR2(NVAR2,NVAR2), &
                 CORR(NVAR2,NVAR1), COV(LDCOV,NVAR1+NVAR2)
!
      DATA COV/1.0000, 0.1839, 0.0489, 0.0186, 0.0782, 0.1147, 0.2137, &
          0.2742, 0.1839, 1.0000, 0.2220, 0.1861, 0.3355, 0.1021, &
          0.4105, 0.4043, 0.0489, 0.2220, 1.0000, 0.2707, 0.2302, &
          0.0931, 0.3240, 0.4047, 0.0186, 0.1861, 0.2707, 1.0000, &
          0.2950, -0.0438, 0.2930, 0.2407, 0.0782, 0.3355, 0.2302, &
          0.2950, 1.0000, 0.2087, 0.2995, 0.2863, 0.1147, 0.1021, &
          0.0931, -0.0438, 0.2087, 1.0000, 0.0760, 0.0702, 0.2137, &
          0.4105, 0.3240, 0.2930, 0.2995, 0.0760, 1.0000, 0.6247, &
          0.2742, 0.4043, 0.4047, 0.2407, 0.2863, 0.0702, 0.6247, &
          1.0000/
!
      DATA IND1/1, 2, 3, 4, 5, 6/, IND2/7, 8/
!
      CALL CANVC (NDF, COV, IND1, IND2, CORR, &
                 COEF1, COEF2, COEFR1, COEFR2, IPRINT=IPRINT)
!
!
      END
```

Output

```
                 *** Canonical Correlations Statistics ***
      Canonical                                              Prob. of
    Correlations  Wilks Lambda      Raos F  Num. df  Denom. df  Larger F
1       0.6093        0.6159       4.250       12        186    0.0000
2       0.1431        0.9795       0.393        5         94    0.8524

        Group One Canonical Coefficients
        1       2       3       4       5       6
```

```
    1   0.326   0.411  -0.799   0.358  -0.032   0.053
    2   0.481  -0.340  -0.083  -0.766  -0.484  -0.139
    3   0.456   0.718   0.625   0.134  -0.056   0.038
    4   0.202  -0.689   0.060   0.732  -0.335   0.080
    5   0.184  -0.125  -0.064  -0.045   1.079  -0.225
    6  -0.027  -0.174   0.054  -0.086  -0.021   1.017

    Group Two Canonical Coefficients
            1       2
    1   0.464   1.194
    2   0.642  -1.108

    Correlations Between the Group One Variables
         and the Group One Canonical Scores
                          1       2
                  1   0.4517   0.3408
                  2   0.7388  -0.2932
                  3   0.6733   0.4313
                  4   0.4769  -0.5799
                  5   0.5299  -0.2811
                  6   0.1319  -0.0903

    Correlations Between the Group Two Variables
         and the Group Two Canonical Scores
                          1       2
                  1   0.8653   0.5013
                  2   0.9320  -0.3625
```

Discriminant Analysis

Routines

Usage Notes

The routine DSCRM allows linear or quadratic discrimination and the use of either reclassification, split sample, or the leaving-out-one methods in order to evaluate the rule. Moreover, DSCRM can be executed in an online mode, that is, one or more observations can be added to the rule during each invocation of DSCRM.

The mean vectors for each group of observations and an estimate of the common covariance matrix for all groups are input to DMSCR. These estimates can be computed via routine DSCRM. Output from DMSCR are linear combinations of the observations, which at most separate the groups. These linear combinations may subsequently be used for discriminating between the groups. Their use in graphically displaying differences between the groups is possibly more important, however.

Nearest neighbor discrimination is performed in routine NNBRD. In this routine, the user can set the number of nearest neighbors to be used in the discrimination and the threshold for classification. Split samples can also be used.

DSCRM

Performs a linear or a quadratic discriminant function analysis among several known groups.

Required Arguments

NROW — The absolute value of **NROW** is the number of rows of **X** that contain an observation. (Input)
If **NROW** is negative, the observations are deleted from the discriminant statistics. If **NROW** is positive, they are added.

NVAR — Number of variables to be used in the discrimination. (Input)

X — |**NROW**| by **NVAR** + *m* matrix containing the data to be used on this call. (Input, if |**NROW**| > 0; **X** is not referenced otherwise)
m is 1, 2, or 3 depending upon whether any columns in **X** contain frequencies or weights. One column in **X** must contain the group number for each observation. Group numbers must be 1.0, 2. 0, ..., **NGROUP**. If present, **IFRQ** gives the column containing the frequencies, while **IWT** gives the column in **X** containing the weights.

NGROUP — Number of groups in the data. (Input)

COV — **NVAR** by **NVAR** by *g* matrix of covariances. (Output, for **IDO** = 0 or 1; input/output, for **IDO** = 2, 3, or 5; input, for **IDO** = 4; not referenced if **IDO** = 6)
g = **NGROUP** + 1 when **IMTH** = 1, 2, 4, or 5, and *g* = 1 otherwise. When **IMTH** = 3 or 6, the within-group covariance matrices are not computed. Regardless of the value of **IMTH**, the pooled covariance matrix is always computed and saved as the *g*-th covariance matrix in **COV**.

COEF — **NGROUP** by **NVAR** + 1 matrix containing the linear discriminant function coefficients. (Output, if **IDO** = 0 or 3; input, if **IDO** = 4; not referenced if **IDO** = 1, 2, 5, or 6)
The first column of **COEF** contains the constant term, and the remaining columns contain the variable coefficients. Row *i* of **COEF** corresponds to group *i*, for *i* = 1,...,**NGROUP**. Matrix **COEF** is always computed as the linear discriminant function coefficients even when quadratic discrimination is specified. Specifically, given the linear discriminant function $z_i = \ln\left(p_i\right) - 0.5\bar{x}_i^T S_p^{-1}\bar{x}_i + x^T S_p^{-1}\bar{x}_i$, the intercept $\ln\left(p_i\right) - 0.5\bar{x}_i^T S_p^{-1}\bar{x}_i$ is assigned to **COEF(i,1)** and the coefficient vector $S_p^{-1}\bar{x}_i$ is assigned to **COEF(i,2:NVAR+1)** for *i* = 1,...,**NGROUP**. Matrix **COEF** is updated if **IDO** is equal to 0 or 3.

ICLASS — Vector of length |NROW| containing the group to which the observation was classified. (Output, if IDO = 0 or 4; not referenced otherwise)
If an observation has an invalid group number, frequency, or weight when the leaving-out-one method has been specified, then the observation is not classified and the corresponding elements of ICLASS and PROB are set to zero.

PROB — |NROW| by NGROUP matrix containing the posterior probabilities for each observation. (Output, if IDO = 0 or 4; not referenced otherwise)

CLASS — NGROUP by NGROUP matrix containing the classification table. (Output, if IDO = 0 or 1, input/output, if IDO = 4; not referenced otherwise)
Each observation that is classified and has a group number equal to 1.0, 2.0, ..., NGROUP is entered into the table. The rows of the table correspond to the known group membership. The columns refer to the group to which the observation was classified. Classification results accumulate with each call to DSCRM with IDO = 4. For example, if 2 calls with IDO = 4 are made, then the elements in CLASS sum to the total number of valid observations in the 2 calls.

D2 — NGROUP by NGROUP matrix containing the Mahalanobis distances

$$D_{ij}^2$$

between the group means. (Output, when IDO= 0 or 3; not referenced otherwise)
For linear discrimination, the Mahalanobis distance is computed using the pooled covariance matrix. Otherwise, the Mahalanobis distance

$$D_{ij}^2$$

between group means *i* and *j* is computed using the within covariance matrix for group *i* in place of the pooled covariance matrix.

STAT — Vector of length 4 + 2 * (NGROUP + 1) containing statistics of interest. (Input/ Output, if IDO = 3 or 5; output, if IDO = 0 or 1; not referenced otherwise)
The first element of STAT is the sum of the degrees of freedom for the within-covariance matrices. The second, third and fourth elements of STAT correspond to the chi-squared statistic, its degrees of freedom, and the probability of a greater chi-squared, respectively, of a test of the homogeneity of the within-covariance matrices (not computed if IMTH = 3 or 6). The 5-th through 5 + NGROUP elements of STAT contain the log of the determinants of each group's covariance matrix (not computed if IMTH = 3 or 6) and of the pooled covariance matrix (element 5 + NGROUP). Finally, the last NGROUP + 1 elements of STAT contain the sum of the weights within each group and, in the last position, the sum of the weights in all groups.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of DSCRM; all the data are input at once.
1	This is the first invocation of DSCRM with this data, additional calls will be made. Initialization and updating for the NROW observations are performed.
2	This is an intermediate invocation of DSCRM; updating for the NROW observations is performed.
3	All statistics are updated for the NROW observations. The discriminant functions and other statistics are computed.
4	The discriminant functions are used to classify each of the NROW observations in X.
5	The covariance matrices are computed, and workspace is released. No further calls to DSCRM with IDO greater than 1 should be made without first calling DSCRM with IDO = 1.
6	Workspace is released. No further calls to DSCRM with IDO greater than 1 should be made without first calling DSCRM with IDO = 1. This option is not required if a call has been made with IDO = 5 or if workspace is explicitly provided by use of D2CRM.

See Comments 5 and 6 for further information.

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IND — Vector of length NVAR containing the column numbers in X to be used in the discrimination. (Input)
By default, IND(I)=I.

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. Positive IFRQ indicates that column number IFRQ of X contains the frequencies. All frequencies should be integer values. If this is not the case, the NINT (nearest integer) function is used to obtain integer frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)

IWT = 0 means that all weights are 1.0. Positive IWT means that column IWT of X contains the weights. Negative weights are not allowed.

Default: IWT = 0.

IGRP — Column number in X containing the group numbers. (Input)

The group numbers must be 1.0, 2,0, ..., NGROUP for an observation to be used in the discriminant functions. An observation will be classified regardless of its group number when the reclassification method is specified.

Default: IGRP = NVAR + 1.

IMTH — Option parameter giving the method of discrimination. (Input)

IMTH determines whether linear or quadratic discrimination is used whether the group covariance matrices are computed (the pooled covariance matrix is always computed) and whether the leaving-out-one or the reclassification method is used to classify each observation.

Default: IMTH = 1.

IMTH	Discrim.	Coveriance	Classification
1	Linear	All	Reclassification
2	Quadratic	All	Reclassification
3	Linear	Pooled only	Reclassification
4	Linear	All	Leaving-out-one
5	Quadratic	All	Leaving-out-one
6	Linear	Pooled only	Leaving-out-one

In the leaving-out-one method of classification, the posterior probabilities are adjusted so as to eliminate the effect of the observation from the sample statistics prior to its classification. In the reclassification method, the effect of the observation is not eliminated from the classification function. Calls to DSCRM with IMTH = 1, 2, 4, or 5 can be intermixed, as can calls to DSCRM with IMTH = 3 or 6. Calls to DSCRM with IMTH = 1, 2, 4, or 5 cannot be intermixed with calls to DSCRM with IMTH = 3 or 6 without first calling DSCRM with IDO = 1 (or 0).

IPRINT — Printing option. (Input)

Default: IPRINT = 0.

For the given combination of IDO and IPRINT, the following arrays are printed.

IPRINT	IDO	Printing
0	Any	None
1 or 2	0	PRIOR, NI, XMEAN, COV, COEF, ICLASS, PROB, CLASS, D2, STAT, NRMISS
1 or 2	1 or 2	None

IPRINT	IDO	Printing
1 or 2	3	PRIOR, NI, XMEAN, COEF, D2, STAT, NRMISS
1	4	None
2	4	ICLASS, PROB
1 or 2	5	COV, CLASS
1 or 2	6	None

Note that the only change from IPRINT = 1 to IPRINT = 2 is the printing when IDO = 4. Also, note that PRIOR is printed even though it may be input only.

PRIOR — Vector of length NGROUP containing the prior probabilities for each group. (Input, if PRIOR(1) is not -1.0 and IDO is 0, 3 or 4; input/output, if PRIOR(1) is -1.0 and IDO is 0 or 3; not referenced if IDO is 1, 2, 5 or 6)
If PRIOR(1) is not -1.0, then the elements of PRIOR should sum to 1.0. Proportional priors can be selected by setting PRIOR(1) = -1.0. In this case, the prior probabilities will be proportional to the sample size in each group, and the elements of PRIOR will contain the proportional prior probabilities after the first call with IDO = 0 or 3. Use of this optional argument is mandatory if IDO = 4.
Default: PRIOR(1) = -1.0.

NI — Vector of length NGROUP. (Input, for IDO = 4 or 5; input/output, for IDO = 1, 2 or 3; output, for IDO = 0; not referenced if IDO = 6)
The *i*-th element of NI contains the number of observations in group *i*.
Use of this optional argument is mandatory if IDO = 1, 2, 3, 4, 5.

XMEAN — NGROUP by NVAR matrix. (Input, for IDO = 3, 4; input/output, for IDO = 2, 5; output, for IDO = 0 or 1; not referenced if IDO = 6)
The *i*-th row of XMEAN contains the group *i* variable means.

Note that for IDO = 1, 2, 3 ,4 the input and/or output values are just the weighted sums of the observations in each group, i.e. the group *i* variable means $\overline{x}_i$ is calculated as

$$\overline{x}_i = \sum_{k=1}^{M_i} w_{ik} f_{ik} x_{ik}$$

where M_i denotes the number of elements in group *i* and x_{ik} is the *k*-th observation in group *i* with associated weight w_{ik} and frequency f_{ik}.

For IDO = 0 or 5, the means are output and group-wise calculated as weighted arithmetic means,

$$\bar{x}_i = \frac{\sum_{k=1}^{M_i} w_{ik} f_{ik} x_{ik}}{\sum_{k=1}^{M_i} w_{ik} f_{ik}}$$

Use of this optional argument is mandatory if **IDO** = 1,2,3,4,5.

LDXMEA — Leading dimension of **XMEAN** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDXMEA** = size(**XMEAN**,1).

LDCOV — Leading and second dimensions of **COV** exactly as specified in the dimension statement of the calling program. (Input)
The first two dimensions of **COV** must be equal.
Default: **LDCOV** = size (**COV**,1).

LDCOEF — Leading dimension of **COEF** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCOEF** = size (**COEF**,1).

LDPROB — Leading dimension of **PROB** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDPROB** = size (**PROB**,1).

LDCLAS — Leading dimension of **CLASS** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDCLAS** = size (**CLASS**,1).

LDD2 — Leading dimension of **D2** exactly as specified in the dimension statement of the calling program. (Input)
Default: **LDD2** = size (**D2**,1).

NRMISS — Number of rows of data encountered in calls to **DSCRM** containing missing values (NaN) for the classification, group, weight, and/or frequency variables. (Output, if **IDO** = 0 or 1; input/output, if **IDO** = 2 or 3, not referenced otherwise)
If a row of data contains a missing value (NaN) for any of these variables, that row is excluded from the computations.

FORTRAN 90 Interface

Generic: **CALL DSCRM** (**NROW**, **NVAR**, **X**, **NGROUP**, **COV**, **COEF**, **ICLASS**, **PROB**, **CLASS**, **D2**, **STAT** [, ...])

Specific: The specific interface names are **S_DSCRM** and **D_DSCRM**.

FORTRAN 77 Interface

Single: CALL DSCRM (IDO, NROW, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, IGRP, NGROUP, IMTH, IPRINT, PRIOR, NI, XMEAN, LDXMEA, COV, LDCOV, COEF, LDCOEF, ICLASS, PROB, LDPROB, CLASS, LDCLAS, D2, LDD2, STAT, NRMISS)

Double: The double precision name is DDSCRM.

Description

Routine DSCRM performs discriminant function analysis using either linear or quadratic discrimination. The output from DSCRM includes a measure of distance between the groups, a table summarizing the classification results, a matrix containing the posterior probabilities of group membership for each observation, and the within-sample means and covariance matrices. The linear discriminant function coefficients are also computed.

All observations can be input during one call to DSCRM, a method of operation that has the advantage of simplicity. Alternatively, one or more rows of observations can be input during separate calls. This method does not require that all observations be memory resident, a significant advantage with large data sets. Note, however, that DSCRM requires two passes of the data. During the first pass the discriminant functions are computed while in the second pass, the observations are classified. Thus, with the second method of operation, the data will usually need to be input into DSCRM twice.

Because both methods result in the same operations being performed, the algorithm for DSCRM is discussed as if only a few observations are input during each call. The operations performed during each call to DSCRM depend upon the IDO parameter. IDO = 0 should be used if all observations are to be input at one time.

The IDO = 1 step is the initialization step. The variables XMEAN, CLASS, and COV are initialized to zero, and other program parameters are set. After this call, all subroutine arguments except IDO, NROW, X, LDX and IMTH should not be changed by the user except via another call to DSCRM with IDO = 0 or IDO = 1. IMTH can be changed from one call to the next within the two sets {1, 2, 4, 5} or {3, 6} but not between these sets when IDO > 1. That is, do not call DSCRM with IMTH = 1 in one call and IMTH = 3 in another call without first calling DSCRM with IDO = 1.

After initialization has been performed in the IDO = 1 step, the within-group means are updated for all valid observations in X. Observations with invalid group numbers are ignored, as are observations with missing values. The *LU* factorization of the covariance matrices are updated by adding (or deleting) observations via Givens rotations.

The IDO = 2 step is used solely for adding or deleting observations from the model as in the above paragraph.

The IDO = 3 step begins by adding all observations in X to the means and the factorizations of the covariance matrices. It continues by computing some statistics of interest: the linear discriminant functions, the prior probabilities (if PRIOR(1) = -1.0), the log of the determinant of each of the covariance matrices, a test statistic for

testing that all of the within-group covariance matrices are equal, and a matrix of Mahalanobis distances between the groups. The matrix of Mahalanobis distances is computed via the pooled covariance matrix when linear discrimination is specified, the row covariance matrix is used when the discrimination is quadratic.

Covariance matrices are defined as follows. Let N_i denote the sum of the frequencies of the observations in group *i*, and let M_i denote the number of observations in group *i*. Then, if S_i denotes the within-group *i* covariance matrix,

$$S_i = \frac{1}{N_i - 1} \sum_{j=1}^{M_i} w_j f_j \left(x_j - \overline{x} \right) \left(x_j - \overline{x} \right)^T$$

where w_j is the weight of the *j*-th observation in group *i*, f_j is its frequency, x_j is the *j*-th observation column vector (in group *i*), and $\overline{x}$ denotes the mean vector of the observations in group *i*. The mean vectors are computed as

$$\overline{x} = \frac{1}{W_i} \sum_{j=1}^{M_i} w_j f_j x_j$$

where

$$W_i = \sum_{j=1}^{M_i} w_j f_j$$

Given the means and the covariance matrices, the linear discriminant function for group *i* is computed as:

$$z_i = \ln \left(p_i \right) - 0.5 \overline{x}_i^T S_p^{-1} \overline{x}_i + x^T S_p^{-1} \overline{x}_i$$

where $\ln(p_i)$ is the natural log of the prior probability for the *i*-th group, *x* is the observation to be classified, and S_p denotes the pooled covariance matrix.

Let *S* denote either the pooled covariance matrix or one of the within-group covariance matrices S_i. (*S* will be the pooled covariance matrix in linear discrimination, and S_i otherwise.) The Mahalanobis distance between group *i* and group *j* is computed as:

$$D_{ij}^2 = \left(\overline{x}_i - \overline{x}_j \right)^T S^{-1} \left(\overline{x}_i - \overline{x}_j \right)$$

Finally, the asymptotic chi-squared test for the equality of covariance matrices is computed as follows (Morrison 1976, page 252):

$$\gamma = C^{-1}\sum_{i=1}^{k} n_i \left\{ \ln\left(|S_p| \right) - \ln\left(|S_i| \right) \right\}$$

where n_i is the number of degrees of freedom in the *i*-th sample covariance matrix, *k* is the number of groups, and

$$C^{-1} = 1 - \frac{2p^2 + 3p - 1}{6\left(p+1\right)\left(k-1\right)} \left(\sum_{i=1}^{k} \frac{1}{n_i} - \frac{1}{\Sigma_j n_j} \right)$$

where *p* is the number of variables.

When **IDO** = 4, the estimated posterior probability of each observation *x* belonging to group *i* is computed using the prior probabilities and the sample mean vectors and estimated covariance matrices under a multivariate normal assumption. Under quadratic discrimination, the within-group covariance matrices are used to compute the estimated posterior probabilities. The estimated posterior probability of an observation *x* belonging to group *i* is

$$\hat{q}_i\left(x\right) = \frac{e^{-\frac{1}{2}D_i^2(x)}}{\sum_{j=1}^{k} e^{-\frac{1}{2}D_j^2(x)}}$$

where

$$D_{\mathrm{i}}^2\left(x\right) = \begin{cases} \left(x - \overline{x}_{\mathrm{i}}\right)^{\mathrm{T}} S_{\mathrm{i}}^{-1} \left(x - \overline{x}_{\mathrm{i}}\right) + \ln \mid S_{\mathrm{i}} \mid - 2\ln\left(p_{\mathrm{i}}\right) & \mathrm{IMTH} = 1 \text{ or } 2 \\ \left(x - \overline{x}_{\mathrm{i}}\right)^{\mathrm{T}} S_{\mathrm{p}}^{-1} \left(x - \overline{x}_{\mathrm{i}}\right) - 2\ln\left(p_{\mathrm{i}}\right) & \mathrm{IMTH} = 3 \end{cases}$$

For the leaving-out-one method of classification (**IMTH** = 4, 5, and 6), the sample mean vector and sample covariance matrices in the formula for

$$D_i^2\left(x\right)$$

are adjusted so as to remove the observation *x* from their computation. For linear discrimination (**IMTH** = 1, 2, 4, and 6), the linear discriminant function coefficients are actually used to compute the same posterior probabilities.

Using the posterior probabilities, each observations in **X** is classified into a group; the result is tabulated in the matrix **CLASS** and saved in the vector **ICLASS**. **CLASS** is not altered at this stage if **X**(*i*, **IGRP**) contains a group number that is out of range. If the reclassification method is specified, then all observations with no missing values in the **NVAR** classification variables are classified. When the leaving-out-one method is used, observations with invalid group numbers, weights, frequencies or classification variables are not classified. Regardless of the frequency, a 1 is added (or subtracted) from **CLASS** for each row of **X** that is classified and contains a valid group number.

When **IMTH** > 3, adjustment is made to the posterior probabilities to remove the effect of the observation in the classification rule. In this adjustment, each observation is presumed to have a weight of **X**(*i*, **IWT**), if **IWT** > 0 and a frequency of 1.0. See Lachenbruch (1975, page 36) for the required adjustment.

Note that the **X** data in an **IDO** = 3 call do not have to be identical with the X data used in a subsequent IDO = 4 call if the reclassification method (**IMTH**=1,2,3) is used: The observations in the **IDO** = 3 call serve as a training sample to determine the discriminant functions that can be used in the following **IDO** = 4 call to classify new **X** data. Since the reclassification method classifies each observation regardless of its group number, it can also be used for the classification of new observations whose actual group assignment is not known (i.e. group column (**IGRP**) of **X** is not used when **IDO** = 4). See Example 3.

Finally, when **IDO** = 5, the covariance matrices are computed from their *LU* factorizations, and the mean values (**XMEAN**) are scaled.

Comments

1. Workspace may be explicitly provided, if desired, by use of **D2CRM/DD2CRM**. The reference is:

   ```
   CALL D2CRM (IDO, NROW, NVAR, NCOL, X, LDX, IND, IFRQ, IWT, IGRP, NGROUP, IMTH,
       IPRINT, RIOR, NI, XMEAN, LDXMEA, COV, LDCOV, COEF, LDCOEF, ICLASS, PROB,
       LDPROB, CLASS, LDCLAS, D2, LDD2, STAT, NRMISS, D, OB, OB1)
   ```

   The additional arguments are as follows:

   D — Work vector of length equal to (**NGROUP** + 1) * **NVAR** if **IMTH** is not 3 or 6, and of length **NVAR** otherwise.

   OB — Work vector of length equal to **NVAR**.

   OB1 — Work vector of length equal to **NVAR**.

2. Informational errors

Type	Code	Description
3	1	A row of the data matrix X has an invalid group number.
4	2	The variance-covariance matrix for a group is singular.
4	3	The pooled variance-covariance matrix is singular.
3	4	The variance-covariance matrix for a group is singular. STAT(2) cannot be computed. STAT(2) and STAT(4) are set to the missing value code (NaN).
3	5	An element of PRIOR is less than or equal to 10^{-20}.
3	6	The leaving-out-one method is specified, but this observation does not have a valid weight, or it does not have a valid frequency. This observation is ignored.
3	7	The leaving-out-one method is specified, but this observation does not have a valid group number. This observation is ignored.

3. Common choices for the Bayesian prior probabilities are given by:

   PRIOR(*i*) = 1.0/**NGROUP** (equal prior probabilities)

   PRIOR(*i*) = **NI**(*i*)/**NOBS** (proportional prior probabilities)

   PRIOR(*i*) = Past history or subjective judgement

   In all cases, the prior probabilities should sum to 1.0.

4. Two passes of the data are made. In the first pass, the statistics required to compute the discriminant functions are obtained (**IDO** = 1, 2, and 3). In the second pass, the discriminant functions are used to classify the observations. When **IDO** = 0, all of the data are memory resident, and both passes are made in one call to **DSCRM**. When **IDO** > 0 and workspace is not explicitly provided by use of **D2CRM**, a third call to **DSCRM** involving no data is required with **IDO** = 5 or 6.

5. Here are a few rules and guidelines for the correct value of **IDO** in a series of calls.

   (1) Calls with IDO = 0 or 1 may be made at any time. These calls destroy all statistics from previous calls.

   (2) IDO may not be 2, 3, 4, 5, or 6
   (a) immediately after a call where IDO was 0,
   (b) before a call with IDO = 1 has been made, or
   (c) immediately after a call with IDO = 5 or 6 has been made.

   (3) IDO may not be 4 or 5 before a call with IDO = 3 has been made.

   (4) Each series of calls to DSCRM which begins with IDO = 1 should end with IDO = 5 or 6 to ensure the proper release of workspace.This is a valid sequence of IDOs:
   0, 1, 2, 3, 4, 5, 1, 3, 4, 3, 5, 1, 6, 1, 2, 6, 0, 0, 1, 3, 5.

6. Unlike many routines using the parameter **IDO**, because of the workspace allocation and saved variables, neither **DSCRM** or **D2CRM** can be called with **IDO** greater than 1 in consecutive invocations with more than one dataset.

Examples

Example 1

The following example uses linear discrimination with equal prior probabilities on Fisher's (1936) iris data. This example illustrates the execution of **DSCRM** when one call is made.

```
      USE GDATA_INT
      USE DSCRM_INT

      IMPLICIT  NONE
      INTEGER   IGRP, IMTH, IPRINT, LDCLAS, LDCOEF, LDCOV, LDD2, &
                LDPROB, LDX, LDXMEA, NCOL, NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=3, IPRINT=1, LDCOV=4, NCOL=5, &
                  NGROUP=3, NROW=150, NVAR=4, LDCLAS=NGROUP, &
                  LDCOEF=NGROUP, LDD2=NGROUP, LDPROB=NROW, &
                  LDX=NROW, LDXMEA=NGROUP)
!
      INTEGER    ICLASS(NROW), IND(4), NI(NGROUP), NOBS, NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), COEF(LDCOEF,NVAR+1), &
                 COV(LDCOV,LDCOV,1), D2(LDD2,NGROUP), PRIOR(3), &
                 PROB(LDPROB,NGROUP), STAT(6+2*NGROUP), X(LDX,5), &
                 XMEAN(LDXMEA,NVAR)
!
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      CALL DSCRM (NROW, NVAR, X, NGROUP, COV, COEF, ICLASS, PROB, &
                  CLASS, D2, STAT, IND=IND, IGRP=IGRP, IMTH=IMTH, &
                  IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, XMEAN=XMEAN, &
                  NRMISS=NRMISS)
!
      END
```

Output

```
 PRIOR, the prior probabilities
      1        2        3
 0.3333   0.3333   0.3333

 NI, the number in each group
  1    2    3
 50   50   50

      XMEAN, the group means
         1       2       3       4
 1   5.006   3.428   1.462   0.246
 2   5.936   2.770   4.260   1.326
```

```
3   6.588   2.974   5.552   2.026

The pooled within-groups covariance matrix
          1       2       3       4
1   0.2650   0.0927   0.1675   0.0384
2   0.0927   0.1154   0.0552   0.0327
3   0.1675   0.0552   0.1852   0.0427
4   0.0384   0.0327   0.0427   0.0419

COEF, the discriminant function coefficients
        1       2       3       4       5
1   -86.3    23.5    23.6   -16.4   -17.4
2   -72.9    15.7     7.1     5.2     6.4
3  -104.4    12.4     3.7    12.8    21.1

ICLASS, the classifications

 Obs.  Class
  1     1
  2     1
  3     1
  4     1
  5     1
  6     1

      .
      .
      .
145     3
146     3
147     3
148     3
149     3
150     3

PROB, the posterior probabilities
          1       2       3
  1   1.000   0.000   0.000
  2   1.000   0.000   0.000
  3   1.000   0.000   0.000
  4   1.000   0.000   0.000
  5   1.000   0.000   0.000
  6   1.000   0.000   0.000
          .
          .
          .
145   0.000   0.000   1.000
146   0.000   0.000   1.000
147   0.000   0.006   0.994
148   0.000   0.003   0.997
149   0.000   0.000   1.000
150   0.000   0.018   0.982

CLASS, the classification table
          1       2       3
  1   50.00    0.00    0.00
  2    0.00   48.00    2.00
  3    0.00    1.00   49.00

D2, the distances between group means
```

```
            1       2       3
  1       0.0    89.9   179.4
  2      89.9     0.0    17.2
  3     179.4    17.2     0.0

                                        STAT
     1      2      3      4      5      6      7      8      9     10
 147.0    NaN    NaN    NaN    NaN    NaN    NaN  -10.0   50.0   50.0

 11      12
 50.0   150.0

 NRMISS, number of missing observations =   0
```

Example 2

Continuing with Fisher's iris data, the following example computes the quadratic discriminant functions using values of `IDO` > 0. In the first loop, all observations are added to the functions, two observations at a time. In the second loop, each of three observations is classified, one by one, using the leaving-out-one method. Output for statistics that are identical to those reported in the first example are not printed here.

```
      USE GDATA_INT
      USE DSCRM_INT

      IMPLICIT   NONE
      INTEGER    IGRP, IMTH, LDCLAS, LDCOEF, LDCOV, LDD2, LDPROB, &
                 LDX, LDXMEA, NCOL, NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=2, LDPROB=10, LDX=150, &
                NCOL=5, NGROUP=3, NROW=1, NVAR=4, LDCLAS=NGROUP, &
                LDCOEF=NGROUP, LDCOV=NVAR, LDD2=NGROUP, LDXMEA=NGROUP)
!
      INTEGER    I, ICLASS(LDPROB), IDO, IND(4), IPRINT, NI(NGROUP), &
                 NOBS, NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), COEF(LDCOEF,NVAR+1), &
                COV(LDCOV,LDCOV,NGROUP+1), D2(LDD2,NGROUP), PRIOR(3), &
                PROB(LDPROB,NGROUP), STAT(6+2*NGROUP), X(LDX,5), &
                XMEAN(LDXMEA,NVAR)
!
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      IPRINT = 0
      IDO    = 1
      CALL DSCRM (0, NVAR, X, NGROUP, COV, COEF, ICLASS, &
                 PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                 IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                 XMEAN=XMEAN, NRMISS=NRMISS)
!                                Add the observations
      IDO = 2
      DO 10  I=1, NOBS
      CALL DSCRM (NROW, NVAR, X(I:,1:), NGROUP, COV, COEF, ICLASS, &
                  PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                  IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
   10 CONTINUE
!                                Summarize the statistics
```

```
      IDO = 3
      CALL DSCRM (0, NVAR, X, NGROUP, COV, COEF, ICLASS, &
                  PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, IGRP=IGRP, &
                  IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
!                                 Classify the first three observations
      IPRINT = 2
      IDO    = 4
      DO 20  I=1, 3
      CALL DSCRM (NROW, NVAR, X(I:,1:), NGROUP, COV, COEF, ICLASS(I:),&
      PROB(I:,1:), CLASS, D2, STAT, IDO=IDO, IND=IND, &
      IGRP=IGRP, IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, &
      NI=NI, XMEAN=XMEAN, NRMISS=NRMISS)
   20 CONTINUE
!                                 Release Workspace
      IDO = 6
      CALL DSCRM (0, NVAR, X, NGROUP, COV, &
                  COEF, ICLASS, PROB, CLASS, D2, STAT, IDO=IDO, IND=IND, &
                  IGRP=IGRP, IMTH=IMTH, IPRINT=IPRINT, PRIOR=PRIOR, NI=NI, &
                  XMEAN=XMEAN, NRMISS=NRMISS)
!
      END
```

Output

```
 ICLASS, the classifications

 Obs.  Class
 1     1

 PROB, the posterior probabilities
     1       2       3
 1.000   0.000   0.000

 ICLASS, the classifications

 Obs.  Class
 1     1

 PROB, the posterior probabilities
     1       2       3
 1.000   0.000   0.000

 ICLASS, the classifications
 Obs.  Class
 1     1

 PROB, the posterior probabilities
     1       2       3
 1.000   0.000   0.000
```

Example 3

Fisher's iris data consists of 50 samples from each of three species of the Iris flower. The first 50 data sets belong to species 1, the second 50 data sets to species 2 and the last 50 data sets to species 3. In this example, 30 data sets from each of the 3 species are used to train the quadratic discrimination functions. In a second step, the

computed discriminant functions are used to classify the remaining data sets. The classification results show that two data sets are misclassified by the discriminant functions: One data set belonging to species 2 is erroneously assigned to species 3 and one data set belonging to species 3 is erroneously assigned to species 2.

```
      ?USE GDATA_INT
      USE DSCRM_INT
!
      IMPLICIT NONE
      INTEGER, PARAMETER :: IGRP = 1, IMTH = 2, NGROUP = 3, NVAR = 4,&
                            NTRAIN = 90, NCLASS = 60, NTOTAL = 150
      INTEGER :: IDO, NROW, NOBS, NV
      INTEGER :: NI(NGROUP), IND(NVAR), ICLASS(NCLASS)
      REAL :: RPRIOR(NGROUP), STAT(6+2*(NGROUP)), X(NTOTAL,NVAR+1),&
              COEF(NGROUP,NVAR+1), COV(NVAR,NVAR,NGROUP+1),&
              PROB(NCLASS,NGROUP), CLASS(NGROUP,NGROUP),&
              D2(NGROUP,NGROUP), XMEAN(NGROUP,NVAR),&
              TRAINDATA(NTRAIN,NVAR+1), CLASSDATA(NCLASS,NVAR+1)
!
      IND = (/ 2, 3, 4, 5 /)
!
! Read Fisher's Iris data into array x
!
      CALL GDATA (3, X, NOBS, NV)
!
! Prepare training data.
! Use 90 data sets of Fisher's Iris data for training the
! discriminant functions. Fisher's Iris data consist of 50
! samples from each of three species of Iris. The first 50
! sets belong to species 1, the second 50 sets belong to
! species 2, and the last 50 sets belong to species 3. Add
! the first 30 data sets of each species to the training
! data.
!
      TRAINDATA(1:30,1:5) = X(1:30,1:5)
      TRAINDATA(31:60,1:5) = X(51:80,1:5)
      TRAINDATA(61:90,1:5) = X(101:130,1:5)
!
! Prepare classification data, the last 20 data of each
! species of Fisher's Iris data.
!
      CLASSDATA(1:20,1:5) = X(31:50,1:5)
      CLASSDATA(21:40,1:5) = X(81:100,1:5)
      CLASSDATA(41:60,1:5) = X(131:150,1:5)
!
! Initialize and add the training data
!
      NROW = 90
      RPRIOR(1) = -1.0e0
      CALL DSCRM (NROW, NVAR, TRAINDATA, NGROUP, COV, COEF, ICLASS,&
                  PROB, CLASS, D2, STAT, IDO=1, IND=IND, IGRP=IGRP,&
                  IMTH=IMTH, PRIOR=RPRIOR, NI=NI, XMEAN=XMEAN)
!
! Compute quadratic discriminant functions (IMTH=2) using the
! training data
!
      CALL DSCRM (0, NVAR, TRAINDATA, NGROUP, COV, COEF, ICLASS,&
                  PROB, CLASS, D2, STAT, IDO=3, IND=IND, IGRP=IGRP,&
                  IMTH=IMTH, PRIOR=RPRIOR, NI=NI, XMEAN=XMEAN)
```

```
!
! Apply discriminant functions to the remaining data sets of
! Fishers's Iris data for group classification and print ICLASS
! and PROB.
!
   NROW = 60
   CALL DSCRM (NROW, NVAR, CLASSDATA, NGROUP, COV, COEF, ICLASS,&
               PROB, CLASS, D2, STAT, IDO=4, IND=IND, IGRP=IGRP,&
               IMTH=IMTH, PRIOR=RPRIOR, NI=NI, XMEAN=XMEAN,&
               IPRINT=2)
!
! Compute and print posterior probabilities, covariance matrices and
! classification table. Release Workspace.
! The entries in ICLASS and the classification table show that two
! sets in the classification data - one from species 2 and one from
! species 3 - are misclassified by the discriminant functions.
!
   CALL DSCRM (NROW, NVAR, CLASSDATA, NGROUP, COV, COEF, ICLASS,&
               PROB, CLASS, D2, STAT, IDO=5, IND=IND, IGRP=IGRP,&
               IMTH=IMTH, PRIOR=RPRIOR, NI=NI, XMEAN=XMEAN,&
               IPRINT=2)
!
   END

Output

 ICLASS, the classifications

   Obs.  Class

     1     1
     2     1
     3     1
     4     1
     5     1
     6     1
     7     1
     8     1
     9     1
    10     1
    11     1
    12     1
    13     1
    14     1
    15     1
    16     1
    17     1
    18     1
    19     1
    20     1
    21     2
    22     2
    23     2
    24     3
    25     2
    26     2
    27     2
    28     2
    29     2
    30     2
```

```
        31     2
        32     2
        33     2
        34     2
        35     2
        36     2
        37     2
        38     2
        39     2
        40     2
        41     3
        42     3
        43     3
        44     2
        45     3
        46     3
        47     3
        48     3
        49     3
        50     3
        51     3
        52     3
        53     3
        54     3
        55     3
        56     3
        57     3
        58     3
        59     3
        60     3

    PROB, the posterior probabilities
              1       2       3
         1   1.000   0.000   0.000
         2   1.000   0.000   0.000
         3   1.000   0.000   0.000
         4   1.000   0.000   0.000
         5   1.000   0.000   0.000
         6   1.000   0.000   0.000
         7   1.000   0.000   0.000
         8   1.000   0.000   0.000
         9   1.000   0.000   0.000
        10   1.000   0.000   0.000
        11   1.000   0.000   0.000
        12   1.000   0.000   0.000
        13   1.000   0.000   0.000
        14   1.000   0.000   0.000
        15   1.000   0.000   0.000
        16   1.000   0.000   0.000
        17   1.000   0.000   0.000
        18   1.000   0.000   0.000
        19   1.000   0.000   0.000
        20   1.000   0.000   0.000
        21   0.000   1.000   0.000
        22   0.000   1.000   0.000
        23   0.000   1.000   0.000
        24   0.000   0.131   0.869
        25   0.000   0.943   0.057
        26   0.000   0.993   0.007
        27   0.000   1.000   0.000
```

```
        28   0.000   0.999   0.001
        29   0.000   0.999   0.001
        30   0.000   0.997   0.003
        31   0.000   0.979   0.021
        32   0.000   0.994   0.006
        33   0.000   1.000   0.000
        34   0.000   1.000   0.000
        35   0.000   0.997   0.003
        36   0.000   0.999   0.001
        37   0.000   0.999   0.001
        38   0.000   1.000   0.000
        39   0.000   1.000   0.000
        40   0.000   0.999   0.001
        41   0.000   0.000   1.000
        42   0.000   0.007   0.993
        43   0.000   0.000   1.000
        44   0.000   0.506   0.494
        45   0.000   0.001   0.999
        46   0.000   0.000   1.000
        47   0.000   0.000   1.000
        48   0.000   0.040   0.960
        49   0.000   0.254   0.746
        50   0.000   0.007   0.993
        51   0.000   0.000   1.000
        52   0.000   0.000   1.000
        53   0.000   0.002   0.998
        54   0.000   0.000   1.000
        55   0.000   0.000   1.000
        56   0.000   0.000   1.000
        57   0.000   0.004   0.996
        58   0.000   0.010   0.990
        59   0.000   0.000   1.000
        60   0.000   0.086   0.914

         The group 1 covariance matrix
              1        2        3        4
    1   0.1386   0.1010   0.0180   0.0160
    2   0.1010   0.1226   0.0017   0.0193
    3   0.0180   0.0017   0.0344   0.0058
    4   0.0160   0.0193   0.0058   0.0102

         The group 2 covariance matrix
              1        2        3        4
    1   0.2980   0.1021   0.1931   0.0572
    2   0.1021   0.1078   0.0852   0.0443
    3   0.1931   0.0852   0.2113   0.0730
    4   0.0572   0.0443   0.0730   0.0446

         The group 3 covariance matrix
              1        2        3        4
    1   0.4745   0.1092   0.3925   0.0436
    2   0.1092   0.1120   0.0878   0.0470
    3   0.3925   0.0878   0.3927   0.0620
    4   0.0436   0.0470   0.0620   0.0655

    The pooled within-groups covariance matrix
                1        2        3        4
       1   0.3037   0.1041   0.2012   0.0389
       2   0.1041   0.1141   0.0582   0.0369
       3   0.2012   0.0582   0.2128   0.0469
```

```
        4   0.0389   0.0369   0.0469   0.0401

     CLASS, the classification table
                1       2       3
        1   20.00    0.00    0.00
        2    0.00   19.00    1.00
        3    0.00    1.00   19.00
```

DMSCR

more...

Uses Fisher's linear discriminant analysis method to reduce the number of variables.

Required Arguments

XMEAN — NGROUP by NVAR matrix containing the means of the variables in each group. (Input)

SUMWT — Vector of length NGROUP containing the sum of the weights of the observations in each group. (Input)

COV — NVAR by NVAR matrix containing the pooled within-groups variance-covariance matrix S_p. (Input)

NNV — Number of eigenvectors extracted from

$$S_p^{-1} S_b$$

the standardized between-groups variance-covariance matrix. (Output)
S_p is the pooled within-groups variance-covariance matrix, and S_b is the between-groups variance-covariance matrix. NNV is usually the minimum of NVAR and NGROUP - 1, but it may be smaller if any row of XMEAN or COV is a linear combination of the other rows.

EVAL — Vector of length NNV containing the eigenvalues extracted from the standardized between-means variancecovariance matrix, in descending order. (Output)
NNV is less than or equal to the minimum of NVAR and (NGROUP - 1).

COEF — NVAR by NNV matrix of eigenvectors from the standardized between-means variance-covariance matrix. (Output)
The eigenvector coefficients have been standardized such that the canonical scores can be obtained directly by multiplication of the original data by COEF.

CMEAN — NGROUP by NNV matrix of group means of the canonical variables. (Output)

Optional Arguments

NGROUP — Number of groups. (Input)
Default: NGROUP = size (XMEAN,1).

NVAR — Number of variables. (Input)
Default: NVAR = size (XMEAN,2).

LDXMEA — Leading dimension of XMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDXMEA = size (XMEAN,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCMEA — Leading dimension of CMEAN exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCMEA = size (CMEAN,1).

FORTRAN 90 Interface

Generic: CALL DMSCR (XMEAN, SUMWT, COV, NNV, EVAL, COEF, CMEAN [, ...])
Specific: The specific interface names are S_DMSCR and D_DMSCR.

FORTRAN 77 Interface

Single: CALL DMSCR (NGROUP, NVAR, XMEAN, LDXMEA, SUMWT, COV, LDCOV, NNV, EVAL, COEF, LDCOEF, CMEAN, LDCMEA)
Double: The double precision name is DDMSCR.

Description

Routine DMSCR is a natural generalization of R.A. Fisher's linear discrimination procedure for two groups. This method of discrimination obtains those linear combinations of the observed random variables that maximize the between-groups variation relative to the within groups variation. Denote the first of these linear combinations by

$$z_1 = \beta_1^T x$$

where β_1 is a column vector of coefficients of length **NVAR** and *x* is an observation to be classified. On the basis of one linear combination, the discriminant rule assigns the observation, *z*, to a group (characterized by the group mean) by minimizing the Euclidean distance between *z* and the group mean.

To obtain β_1 (see, e.g., Tatsuoka 1971, page 158), let S_p denote the pooled within-groups covariance matrix (S_p is defined and can be computed via routine **DSCRM**) and let S_b denote the between-groups covariance matrix defined by

$$S_b = \sum_{i=1}^{g} w_i \left(\overline{x}_i - \overline{x}\right)\left(\overline{x}_i - \overline{x}\right)^T / \left(N - g\right)$$

where *g* is the number of groups,

$$\overline{x}_i$$

is the mean vector for the *i*-th group of observations, $\overline{x}$ denotes the vector of means over all observations, w_i is the sum of the weights times the frequencies as input in **SUMWT** and as used in the computation of

$$\overline{x}_i$$

and *N* is the total number of observations used in computing **COV**. Then, β_1, such that

$$\beta_1^T S_p \beta_1 = 1$$

can be computed as the maximum of

$$\psi = \beta_1^T S_b \beta_1$$

This yields β_1 as the eigenvector associated with the largest eigenvalue from

$$S_p^{-1} S_b$$

Generally,

$$S_p^{-1} S_b$$

has rank *m*, where *m* = min(*g* - 1, *p*) and *p* = **NVAR**.

$$S_p^{-1} S_b$$

has *m* such eigenvectors, and the matrix **COEF** is obtained as (β_1, β_2, ..., β_m), where each β_i is an eigenvector.

The matrix **CMEAN** is taken as the within-group means vector of the linear combinations z_i defined by the **β**'s. For each observation *x*, scores

$$z_i = \beta_i^T x$$

can be computed, because of the restriction on β_i, the sample variance of the z_i is 1.0. The observation is classified into the group (as specified by the group mean of the z_i's) to which, on the basis of the z_i, the Euclidean distance is the least.

Note that the linear combinations z_i have meaning even when discrimination is not desired. The linear combination of the observed variables that most separates the *g* groups is z_1; z_2, giving the second highest such separation orthogonal to the first, and so on. Thus, a plot of the mean vectors of the first two variables gives a good two-dimensional summarization of the relationships between the groups.

Comments

1. Workspace may be explicitly provided, if desired, by use of **D2SCR/DD2SCR**. The reference is:

   ```
   CALL D2SCR (NGROUP, NVAR, XMEAN, LDXMEA, SUMWT, COV, LDCOV, NNV, EVAL, COEF,
       LDCOEF, CMEAN, LDCMEA, BCOV, EVAL2, EVEC, WKR, WK)
   ```

   The additional arguments are as follows:

   BCOV — Work array of length **NVAR * NVAR**.

   EVAL2 — Work array of length **NVAR**.

   EVEC — Work array of length **NVAR * NVAR**.

   WKR — Work array of length **NVAR * NVAR**.

   WK — Work array of length 2 *** NVAR**.

2. IMSL routine **DSCRM** may be used to calculate the input arrays for this routine from the original data.

Example

The following example illustrates a typical sequence. Fisher's iris data is used. (See routine **GDATA**, Chapter 19, "Utilities"). Routine **DSCRM** is first used to perform a discriminant analysis based on all the variables. **COV**, **XMEAN**, and **NI** are obtained from **DSCRM**. Function **DMSCR**, which uses these arrays, is then called.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    IGRP, IMTH, IPRINT, LDCLAS, LDCMEA, LDCO, LDCOEF, &
                 LDCOV, LDD2, LDPROB, LDX, LDXMEA, NCOL, &
                 NGROUP, NROW, NVAR
      PARAMETER  (IGRP=1, IMTH=3, IPRINT=0, LDCOV=4, NCOL=5, NGROUP=3, &
```

```
                    NROW=150, NVAR=4, LDCLAS=NGROUP, LDCMEA=NGROUP, &
                    LDCO=NGROUP, LDCOEF=NVAR, LDD2=NGROUP, LDPROB=NROW, &
                    LDX=NROW, LDXMEA=NGROUP)
!
      INTEGER    ICLASS(NROW), IND(4), NI(NGROUP), NNV, NOBS, NOUT, &
                 NRMISS, NV
      REAL       CLASS(LDCLAS,NGROUP), CMEAN(LDCMEA,NGROUP-1), &
                 CO(LDCO,NVAR+1), COEF(LDCOEF,NGROUP-1), &
                 COV(LDCOV,LDCOV,1), D2(LDD2,NGROUP), EVAL(NGROUP-1), &
                 PRIOR(3), PROB(LDPROB,NGROUP), REAL, &
                 STAT(6+2*NGROUP), SUMWT(NGROUP), X(LDX,5), &
                 XMEAN(LDXMEA,NVAR)
      INTRINSIC  REAL
!
      DATA IND/2, 3, 4, 5/, PRIOR/0.3333333, 0.3333333, 0.3333333/
!
      CALL GDATA (3, X, NOBS, NV)
!
      CALL DSCRM (NROW, NVAR, X, NGROUP, COV(1:,1:,1), CO, ICLASS, &
                  PROB, CLASS, D2, STAT, IND=IND, IGRP=IGRP, IMTH=IMTH, &
                  PRIOR=PRIOR, XMEAN=XMEAN)
!
      SUMWT(1) = STAT(6+NGROUP)
      SUMWT(2) = STAT(7+NGROUP)
      SUMWT(3) = STAT(8+NGROUP)
!
      CALL DMSCR (XMEAN, SUMWT, COV(1:,1:,1), NNV, EVAL, COEF, CMEAN)

      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' NNV = '',I1)') NNV
      CALL WRRRN ('EVAL', EVAL, 1, NNV, 1)
      CALL WRRRN ('COEF', COEF)
      CALL WRRRN ('CMEAN', CMEAN)
      END
```

Output

```
 NNV = 2

      EVAL
     1       2
 32.19    0.29

         COEF
         1       2
1  -0.829   0.024
2  -1.534   2.165
3   2.201  -0.932
4   2.810   2.839

        CMEAN
         1       2
1  -5.502   6.877
2   3.930   5.934
3   7.888   7.174
```

NNBRD

Performs *k* nearest neighbor discrimination.

Required Arguments

X — NROW by NVAR + 1 matrix containing the data to be used on this call. (Input/Output)
One column in X must contain the group number for each observation. On output, X is sorted into a *k-d* tree. The first NRULE + NCLASS rows of X must not contain missing values in the columns specified by IND and IGRP.

K — Number of nearest neighbors to be used in the discriminant rule. (Input)

NGROUP — Number of groups in the data. (Input)

NRULE — Number of observations in X to be used in the discriminant rule. (Input)
The first |NRULE| observations in X are used as the set defining the rule. If NRULE is positive, then the NRULE observations defining the rule are classified. If NRULE is negative, the NRULE observations defining the rule are not classified.

NCLASS — Number of observations in X to classify. (Input)
NCLASS is the number of observations in a second sample that may be used to test the rule formed from the first NRULE observations. If present, this sample is in rows
NRULE + 1 through NRULE + NCLASS of X.

THRESH — Threshold for the posterior probabilities. (Input)
If the maximum posterior probability is less than THRESH, the observation is classified into group NGROUP + 1 (the group "other").

PART — Vector of length NRULE containing the values to be used in the partition of X for the *k-d* tree. (Output)

IDISCR — Vector of length NRULE containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree. (Output)
IDISCR(*i*) = 0 if the observation is a terminal node. IND(IDISCR(*i*)) is the column number in X to be used as the discriminator.

NI — Vector of length NGROUP containing the number of observations in each group. (Output)

ICLASS — Vector of length *m* containing the group to which the observation was classified. (Output)
If NRULE > 0, *m* = NRULE + NCLASS; otherwise, *m* = NCLASS. The *i*-th element in ICLASS corresponds to to *i*-th row in the sorted matrix X.

PROB — *m* by NGROUP matrix containing the posterior probabilities for each observation. (Output)
The *i*-th row in PROB corresponds to the *i*-th row in the in the sorted matrix X.

CLASS — NGROUP by NGROUP + 1 matrix containing the classification table. (Output)
Each observation that is classified and has a group number equal to 1.0, 2.0, ..., NGROUP is entered into the table. The rows of the table correspond to the known group membership. The columns refer to the group to which the observation was classified. Column NGROUP + 1 refers to the column "other" (see THRESH).

Optional Arguments

NROW — Number of rows of X that contain an observation. (Input)
Default: NROW = size (X,1).

NVAR — Number of variables to be used in the discrimination. (Input)
Default: NVAR = size (X,2) – 1.

NCOL — Number of columns in matrix X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IND — Vector of length NVAR containing the column numbers in X to be used in the discrimination. (Input)
By default, IND(I)=1.

IGRP — Column number in X containing the group numbers. (Input)
The group numbers must be 1.0, 2.0, ..., NGROUP for an observation to be used in the discriminant functions. (Note, however, that the nearest integer (NINT) function is used to obtain the group numbers.)
Default: IGRP = NVAR + 1.

METRIC — Metric to be used in computing the *k* nearest neighbors. (Input)
Default: METRIC = 0.

METRIC	Metric used
0	Euclidean distance
1	L_1 norm
2	L_∞ norm

PRIOR — Vector of length NGROUP containing the prior probabilities for each group. (Input, if PRIOR(1) is not -1.0; input/output, if PRIOR(1) is -1.0)
If PRIOR(1) is not -1.0, then the elements of PRIOR should sum to 1.0. Proportional priors can be selected by setting PRIOR(1) = -1.0. In this case, the prior probabilities will be proportional to the sample size in each group based upon the first NRULE observations, and the elements of PRIOR will contain the proportional prior probabilities on return from NNBRD.
Default: PRIOR(1) = -1.0.

LDPROB — Leading dimension of PROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPROB = size (PROB,1).

LDCLAS — Leading dimension of CLASS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCLAS = size (CLASS,1).

FORTRAN 90 Interface

Generic: CALL NNBRD (X, K, NGROUP, NRULE, NCLASS, THRESH, PART, IDISCR, NI, ICLASS, PROB, CLASS [, ...])

Specific: The specific interface names are S_NNBRD and D_NNBRD.

FORTRAN 77 Interface

Single: CALL NNBRD (NROW, NVAR, NCOL, X, LDX, K, IND, IGRP, NGROUP, NRULE, NCLASS, METRIC, PRIOR, THRESH, PART, IDISCR, NI, ICLASS, PROB, LDPROB, CLASS, LDCLAS)

Double: The double precision name is DNNBRD.

Description

Routine NNBRD performs *k*-th nearest neighbor discriminant function analysis. The *k-d* tree algorithm of Friedman, Bentley, and Finkel (1977) is used to find the nearest neighbors. Consult this reference for a discussion of *k-d* trees and how one goes about finding nearest neighbors in them.

In **NNBRD**, the *k* nearest neighbors of any observation used in forming the rule (i.e., one of the first **NRULE** observations in **X**), do not include the observation. Let k_i(*i* = 1, ..., **NGROUP**) denote the number of nearest neighbors found from each of the groups for a given observation ($\Sigma_i k_i = k$); let p_i = **PRIOR**(*i*)($\Sigma_i p_i = 1$); and let

$$\hat{\theta}_i$$

denote the estimated posterior probability of membership in group *i*. Compute

$$\hat{\theta}_i \text{ as } \hat{\theta}_i = \frac{k_i p_i / n_i}{\sum_{j=1}^{g} k_j p_j / n_j}$$

where *g* = **NGROUP**. (If $n_j = 0$ for some *j*, the associated term in the denominator is excluded and

$$\hat{\theta}_j$$

is set to 0.0.)

Let *m* denote the index of the maximum

$$\hat{\theta}_i$$

and ϕ = **THRESH**. Then if

$$\hat{\theta}_m > \phi$$

the observation is classified into group *m*. If

$$\hat{\theta}_m \leq \phi$$

or if the maximum $\hat{\theta}$ is not unique, then the observation is not classified into any group and **ICLASS** is set to zero.

Three metrics are available in **NNBRD** for finding the nearest neighbors. These are Euclidean (L_2) distance, L_1 norm, and L_∞ norm. In order to use Mahalanobis distance, $x^T \Sigma^{-1} x$, a transformation $y = \Sigma^{-1/2} x$ is first needed so that Var(*y*) = *I*. These transformations can be accomplished by use of the mathematical routines. The L_2 norm would then be used with *y* as input to obtain the Mahalanobis metric.

Comments

Workspace may be explicitly provided, if desired, by use of N2BRD/DN2BRD. The reference is:

```
CALL N2BRD (NROW, NVAR, NCOL, X, LDX, K, IND, IGRP, NGROUP, NRULE, NCLASS,
    METRIC, PRIOR, THRESH, PART, IDISCR, NI, ICLASS, PROB, LDPROB, CLASS,
    LDCLAS, WK, IWK, ILOW, IHIGH, ISIDE, BNDL, BNDH, XKEY, IPQR, PQD)
```

The additional arguments are as follows:

WK — Work vector of length NROW.

IWK — Work vector of length NROW.

ILOW — Work vector of length $\log_2$(NROW) + 3.

IHIGH — Work vector of length $\log_2$(NROW) + 3.

ISIDE — Work vector of length $\log_2$(NROW) + 3.

BNDL — Work vector of length NVAR * ($\log_2$(NROW) + 3).

BNDH — Work vector of length NVAR * ($\log_2$(NROW) + 3).

XKEY — Work vector of length NVAR.

IPQR — Work vector of length K + 1.

PQD — Work vector of length K + 1.

Example

Fisher's iris data are used to illustrate routine NNBRD. The data consist of three types of iris. NNBRD is called with *k* = 5 and Euclidean distance as the metric. The results show a clear separation of the groups.

```
      USE GDATA_INT
      USE NNBRD_INT
      USE WRRRN_INT
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    IGRP, K, LDCLAS, LDPROB, LDX, NCLASS, NCOL, &
                 NGROUP, NROW, NRULE, NVAR
      REAL       THRESH
      PARAMETER  (IGRP=1, K=5, LDCLAS=3, LDPROB=150, LDX=150, &
                 NCLASS=0, NCOL=5, NGROUP=3, NROW=150, &
                 NRULE=150, NVAR=4, THRESH=0.10)
!
      INTEGER    ICLASS(NROW), IDISCR(NROW), IND(NVAR), NI(NGROUP), &
                 NRA, NRB
      REAL       CLASS(LDCLAS,NGROUP+1), PART(NRULE), PRIOR(NGROUP), &
                 PROB(LDPROB,NGROUP), X(LDX,NCOL)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (3, X, NRA, NRB)
!
      PRIOR(1) = -1.0
```

```
      CALL NNBRD (X, K, NGROUP, NRULE, NCLASS, THRESH, PART, IDISCR, &
                  NI, ICLASS, PROB, CLASS, IND=IND, IGRP=IGRP, PRIOR=PRIOR)
      CALL WRRRN ('The first 10 rows of X', X, 10, NCOL, LDX)
      CALL WRRRN ('PRIOR', PRIOR, 1, NGROUP, 1)
      CALL WRRRN ('The first 10 elements of PART', PART, 1, 10, 1)
      CALL WRIRN ('The first 10 elements of IDISCR', IDISCR, 1, 10, 1)
      CALL WRIRN ('NI', NI, 1, NGROUP, 1)
      CALL WRIRN ('The first 10 elements of ICLASS', ICLASS, 1, 10, 1)
      CALL WRRRN ('The first 10 rows of PROB', PROB, 10, NGROUP, LDPROB)
      CALL WRRRN ('CLASS', CLASS, NGROUP, NGROUP, LDCLAS)
!
      END
```

Output

```
           The first 10 rows of X
          1       2       3       4       5
 1   1.000   4.500   2.300   1.300   0.300
 2   1.000   4.400   2.900   1.400   0.200
 3   1.000   4.800   3.000   1.400   0.300
 4   1.000   4.400   3.000   1.300   0.200
 5   1.000   4.800   3.000   1.400   0.100
 6   1.000   4.300   3.000   1.100   0.100
 7   1.000   4.600   3.100   1.500   0.200
 8   1.000   4.900   3.100   1.500   0.100
 9   1.000   4.900   3.000   1.400   0.200
10   1.000   4.900   3.100   1.500   0.200


           PRIOR
       1       2       3
  0.3333  0.3333  0.3333

                     The first 10 elements of PART
     1      2      3      4      5      6      7      8      9     10
 0.000  0.000  3.000  0.000  3.000  0.000  0.000  4.900  0.000  3.100

     The first 10 elements of IDISCR
 1   2   3   4   5   6   7   8   9  10
 0   0   2   0   2   0   0   1   0   2

      NI
  1    2    3
 50   50   50

     The first 10 elements of ICLASS
 1   2   3   4   5   6   7   8   9  10
 1   1   1   1   1   1   1   1   1   1

 The first 10 rows of PROB
          1       2       3
 1   1.000   0.000   0.000
 2   1.000   0.000   0.000
 3   1.000   0.000   0.000
 4   1.000   0.000   0.000
 5   1.000   0.000   0.000
 6   1.000   0.000   0.000
 7   1.000   0.000   0.000
```

```
      8   1.000   0.000   0.000
      9   1.000   0.000   0.000
     10   1.000   0.000   0.000

                CLASS
            1       2       3
 1   50.00    0.00    0.00
 2    0.00   47.00    3.00
 3    0.00    2.00   48.00
```

Cluster Analysis

Routines

Usage Notes

The routines described in this chapter perform various forms of hierarchical or *K*-means cluster analysis. By appropriate manipulation of the input data, either variables or cases may be clustered. Additionally, for hierarchical clustering, similarity or dissimilarity (distance) matrices created by routines not included in this chapter can be clustered. Hartigan (1975) and Anderberg (1973) are general references that may be used in this chapter.

The first step in agglomerative hierarchical cluster analysis is to compute the distance between each observation (or variable). Initially, each observation (variable) is treated as a cluster. The two clusters that are closest to one another in distance are merged, and the distance of the new cluster from all other clusters is computed. This process continues until only one cluster remains. No attempt at finding an optimal clustering (in the sense of minimizing some criterion) is made.

The usual steps in a hierarchical cluster analysis might proceed as follows:

1. Routine **CDIST** is used to compute a distance (or possibly a similarity) matrix from the input data matrix. A scaled matrix of Euclidean distances is a common choice for a distance matrix, while a correlation matrix is a common choice for a similarity matrix. If a correlation matrix is to be used, many of the routines described in Chapter 3, "Correlation", may also be used to compute the correlation measures for the matrix. In particular, routine **CORVC** (see Chapter 3, "Correlation") from this chapter may be used.

2. Once the distance matrix has been computed, routine **CLINK** is used to perform the agglomerative hierarchical cluster analysis using either single, complete, average, or Ward's linkage.

3. The results obtained from **CLINK** are examined, and if desired, the number of clusters is selected. Routine **TREEP** in Chapter 16, "Line Printer Graphics" may be used to print the cluster tree. This tree may aid in selecting the number of clusters, assuming that such a number is desired. Based upon the number of clusters selected, routine **CNUMB** is used to obtain the cluster number of each of the clustered observations (or variables).

4. Routines described in Chapter 1, "Basic Statistics" and other chapters in the IMSL STAT/LIBRARY are used to obtain descriptive and other statistics to evaluate the clustering.

Because routine **CDIST** produces similarity and distance matrices for either rows or columns, it is easy to cluster either observations or variables. Optionally, the user may wish to cluster a correlation matrix obtained from one of the routines in the correlation chapter or to input a matrix of similarities (or dissimilarities) obtained via experimentation. The objects within such matrices may be clustered directly in routine **CLINK**.

Basic *K*-means clustering attempts to find a clustering that minimizes the within-cluster sums of squares. In this method of clustering the data, matrix **X** is grouped so that each observation (row in **X**) is assigned to one of a fixed number, *K*, of clusters. The sum of the squared difference of each observation about its assigned clusters

mean is used as the criterion for assignment. In the basic algorithm, observations are transferred from one cluster to another when doing so decreases the within-cluster sums of squared differences. When, in a pass through the entire data set, no transfer occurs, the algorithm stops. Routine **KMEAN** is one implementation of the basic algorithm.

The usual course of events in *K*-means cluster analysis might be to use routine **KMEAN** to obtain the optimal clustering. The clustering is then evaluated via routines described in Chapter 1, "Basic Statistics" and/or other chapters in the IMSL STAT/LIBRARY. Often, *K*-means clustering with more than one value for *K* is performed, and the value of *K* that best fits the data is used.

Clustering can be performed either on observations or on variables. The discussion of the routine **KMEAN** assumes the clustering is to be performed on the observations, which correspond to the rows of the input data matrix. If variables, rather than observations, are to be clustered using **KMEAN**, the data matrix should first be transposed (possibly using routine **TRNRR** (IMSL MATH/LIBRARY)). In the documentation for **KMEAN**, the words "observation" and "variable" would then be exchanged.

CDIST

Computes a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.

Required Arguments

X — NROW by NCOL matrix containing the data. (Input)

DIST — *m* by *m* matrix containing the computed dissimilarities or similarities, where *m* = NROW if IROW = 1 and *m* = NCOL otherwise. (Output)

Optional Arguments

NROW — Number of rows in the matrix. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns in the matrix. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NDSTM — Number of rows (columns, if IROW = 1) to be used in computing the distance measure between the columns (rows). (Input)
Default: NDSTM = size (IND,1) if IND is present. Otherwise, a default value of 2 is used.

IND — Vector of length NDSTM containing the indices of the rows (columns, if IROW = 1) to be used in computing the distance measure. (Input)
If IND(1) = 0; the first NDSTM rows (columns) are used.
By default, the first NDSTM rows (columns) are used.

IMETH — Method to be used in computing the dissimilarities or similarities. (Input)
Default: IMETH = 0.

IMETH	Method
0	Euclidean distance (L_2 norm)
1	Sum of the absolute differences (L_1 norm)
2	Maximum difference ($L\infty$ norm)

IMETH	Method
3	Mahalanobis distance
4	Absolute value of the cosine of the angle between the vectors
5	Angle in radians (0, π) between the lines through the origin defined by the vectors
6	Correlation coefficient
7	Absolute value of the correlation coefficient
8	Number of exact matches

The algorithm section of the manual document has a more detailed description of each measure.

IROW — Row or columns option. (Input)
If IROW = 1, distances are computed between the NROW rows of X. Otherwise, distances between the NCOL columns of X are computed.
Default: IROW = 1.

ISCALE — Scaling option. (Input)
ISCALE is not used for methods 3 through 8.
Default: ISCALE = 0.

ISCALE	Scaling Performed
0	No scaling is performed.
1	Scale each column (row, if IROW = 1) by the standard deviation of the column (row).
2	Scale each column (row, if IROW = 1) by the range of the column (row).

LDDIST — Leading dimension of DIST exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDIST = size (DIST,1).

FORTRAN 90 Interface

Generic: CALL CDIST (X, DIST [, ...])
Specific: The specific interface names are S_CDIST and D_CDIST.

FORTRAN 77 Interface

Single: CALL CDIST (NROW, NCOL, X, LDX, NDSTM, IND, IMETH, IROW, ISCALE, DIST, LDDIST)
Double: The double precision name is DCDIST.

Description

Routine **CDIST** computes an upper triangular matrix (excluding the diagonal) of dissimilarities (or similarities) between the columns or rows of a matrix. Nine different distance measures can be computed. For the first three measures, three different scaling options can be employed. Output from **CDIST** is generally used as input to clustering or multidimensional scaling routines.

The following discussion assumes that the distance measure is being computed between the columns of the matrix, i.e., that **IROW** is not 1. If distances between the rows of the matrix are desired, set **IROW** to 1.

For **IMETH** = 0 to 2, each row of **X** is first scaled according to the value of **ISCALE**. The scaling parameters are obtained from the values in the row scaled as either the standard deviation of the row or the row range; the standard deviation is computed from the unbiased estimate of the variance. If **ISCALE** is 0, no scaling is performed, and the parameters in the following discussion are all 1.0. Once the scaling value (if any) has been computed, the distance between column *i* and column *j* is computed via the difference vector $z_k = (x_k - y_k)/s_k$, $i = 1, \ldots,$ **NDSTM**, where x_k denotes the *k*-th element in the *i*-th column, and y_k denotes the corresponding element in the *j*-th column. For given z_i, the metrics 0 to 2 are defined as:

IMETH	**Metric**
0 $\sqrt{\left(\sum_{i=1}^{\text{NDSTM}} z_i^2\right)}$	Euclidean distance
1 $\sum_{i=1}^{\text{NDSTM}} \lvert z_i \rvert$	$L_1 norm$
2 $\max_i \lvert z_i \rvert$	$L_\infty norm$

Distance measures corresponding to **IMETH** = 3 to 8 do not allow for scaling. These measures are defined via the column vectors $X = (x_i)$, $Y = (y_i)$, and $Z = (x_i - y_i)$ as follows:

IMETH	**Metric**
3	$Z'\hat{\Sigma}^{-1}Z$ = Mahalanobis distance, where $\hat{\Sigma}$ is the usual unbiased sample estimate of the covariance matrix of the rows.
4	$\cos(\theta) = X^T Y / \left(\sqrt{X^T X}\sqrt{Y^T Y}\right)$ = the dot product of *X* and *Y* divided by the length of *X* times the length of *Y*.
5	θ, where θ is defined in 4.
6	ρ = the usual (centered) estimate of the correlation between *X* and *Y*.
7	The absolute value of ρ (where ρ is defined in 6).
8	The number of times $x_i = y_i$, where x_i and y_i are elements of *X* and *Y*.

For the Mahalanobis distance, any variable used in computing the distance measure that is (numerically) linearly dependent upon the previous variables in the IND vector is omitted from the distance measure.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2IST/DC2IST. The reference is:

   ```
   CALL C2IST (NROW, NCOL, X, LDX, NDSTM, IND, IMETH, IROW, ISCALE, DIST, LDDIST,
      X1, X2, SCALE, WK, IND1)
   ```

   The additional arguments are as follows:

   X1 — Work vector of length NDSTM. Not used if IMETH = 8.

   X2 — Work vector of length NDSTM. Not used if IMETH = 8.

   SCALE — Work vector of length NDSTM if IMETH is less than 4; of length NCOL or NROW when IROW is 0 or 1, respectively, and IMETH is 4 or 5; and of length 2 * NCOL or 2 * NROW when IROW is 0 or 1 and IMETH is 6 or 7. SCALE is not used when IMETH is 8.

   WK — Work vector of length NDSTM * NDSTM when IMETH is 3, or of length NDSTM when IMETH = 6 or 7. Not used otherwise.

   IND1 — Integer work vector of length NDSTM.

2. Informational error

Type	Code	Description
3	3	A variable is numerically linearly dependent on the previous variables when IMETH is 3. The variable detected as being linearly dependent is omitted from the distance measure.

Example

The following example illustrates the use of CDIST for computing the Euclidean distance between the rows of a matrix.

```
      USE WRRRN_INT
      USE CDIST_INT

      IMPLICIT   NONE
      INTEGER    IROW, LDDIST, LDX, NCOL, NDSTM, NROW, IMETH
      PARAMETER  (IMETH=0, IROW=1, NCOL=2, NROW=4, LDDIST=NROW, LDX=NROW)
!
      REAL       DIST(LDDIST,NROW), X(NROW,NCOL), IND
!
      DATA IND/0/
      DATA X/1, 1, 1, 1, 1, 0, -1, 2/
      DATA DIST/16*0.0/
!                                Print input matrix
      CALL WRRRN ('X', X)
```

```
!
      CALL CDIST (X, DIST)
!                                    Print distance matrix
      CALL WRRRN ('DIST', DIST)
!
      END
```

Output

```
            X
           1       2
1   1.000   1.000
2   1.000   0.000
3   1.000  -1.000
4   1.000   2.000

               DIST
         1       2       3       4
1   0.000   1.000   2.000   1.000
2   0.000   0.000   1.000   2.000
3   0.000   0.000   0.000   3.000
4   0.000   0.000   0.000   0.000
```

CLINK

Performs a hierarchical cluster analysis given a distance matrix.

Required Arguments

DIST — NPT by NPT matrix containing the distance (or similarity) matrix.(Input/Output)
DIST is a symmetric matrix. On input, only the upper triangular part needs to be present. The routine CLINK saves the upper triangular part of DIST in the lower triangle. On return from CLINK, the upper triangular part of DIST is restored, and the matrix has been made symmetric.

CLEVEL — Vector of length NPT - 1 containing the level at which the clusters are joined. (Output)
CLEVEL(*k*) contains the distance (or similarity) level at which cluster NPT + *k* was formed. If the original data in DIST was transformed via the option parameter IDIST, the inverse transformation is applied to the values in CLEVEL prior to exit from CLINK.

ICLSON — Vector of length NPT - 1 containing the left sons of each merged cluster. (Output)
Cluster NPT + *k* is formed by merging clusters ICLSON(*k*) and ICRSON(*k*).

ICRSON — Vector of length NPT - 1 containing the right sons of each merged cluster. (Output)
Cluster NPT + *k* is formed by merging clusters ICLSON(*k*) and ICRSON(*k*).

Optional Arguments

NPT — Number of data points to be clustered. (Input)
Default: NPT = size (DIST,2).

IMETH — Option giving the method to be used for clustering. (Input)
Default: IMETH = 0.

IMETH	Method
0	Single linkage (minimum distance)
1	Complete linkage (maximum distance)
2	Average distance within (average distance between objects within the merged cluster)
3	Average distance between (average distance between objects in the two clusters)
4	Ward's method (minimize the within-cluster sums of squares). For Ward's method, the elements of DIST are assumed to be Euclidean distances.

IDIST — Option giving the type of transformation to be applied to the measures in DIST. (Input)
Default: IDIST = 0.

IDIST	Transformation
0	No transformation is required. The elements of DIST are distances.
1	Convert similarities to distances by multiplication by -1.0.
2	Convert similarities (usually correlations) to distances by taking the reciprocal of the absolute value.

LDDIST — Leading dimension of DIST exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDIST = size (DIST,1).

FORTRAN 90 Interface

Generic: CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON [, ...])
Specific: The specific interface names are S_CLINK and D_CLINK.

FORTRAN 77 Interface

Single: CALL CLINK (NPT, IMETH, IDIST, DIST, LDDIST, CLEVEL, ICLSON, ICRSON)
Double: The double precision name is DCLINK.

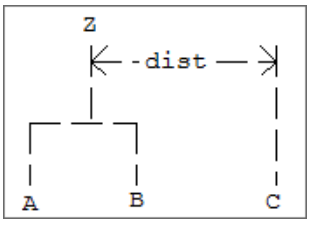
Description

Routine CLINK performs hierarchical cluster analysis based upon a distance matrix, or by appropriate use of the IDIST option, based upon a similarity matrix. Only the upper triangular part of the matrix needs to be input to CLINK.

Hierarchical clustering in **CLINK** proceeds as follows. Initially, each data point is considered to be a cluster, numbered 1 to *n* = **NPT**.

1. If the data matrix contains similarities, they are converted to distances by the method specified in **IDIST**. Set *k* = 1.
2. A search is made of the distance matrix to find the two closest clusters. These clusters are merged to form a new cluster, numbered *n* + *k*. The cluster numbers of the two clusters joined at this stage are saved in **ICRSON** and **ICLSON**, and the distance measure between the two clusters is stored in **CLEVEL**.
3. Based upon the method of clustering, updating of the distance measure in the row and column of **DIST** corresponding to the new cluster is performed.
4. Set *k* = *k* + 1. If *k* < *n*, go to Step 2.

The five methods differ primarily in how the distance matrix is updated after two clusters have been joined. The **IMETH** option parameter specifies how the distance of the cluster just merged with each of the remaining clusters will be updated. Routine **CLINK** allows five methods of computing the distances. To understand these measures, suppose in the following discussion that clusters "*A*" and "*B*" have just been joined to form cluster "*Z*", and interest is in computing the distance of *Z* with another cluster called "*C*".



IMETH	Method
0	This is the single linkage method. The distance from *Z* to *C* is the minimum of the distances (*A* to *C*, *B* to *C*).
1	This is the complete linkage method. The distance from *Z* to *C* is the maximum of the distances (*A* to *C*, *B* to *C*).
2	This is the average-distance-within-clusters method. The distance from *Z* to *C* is the average distance of all objects that would be within the cluster formed by merging clusters *Z* and *C*. This average may be computed according to formulas given by Anderberg (1973, page 139).

IMETH Method

3 This is the average-distance-between-clusters method. The distance from *Z* to *C* is the average distance of objects within cluster *Z* to objects within cluster *C*. This average may be computed according to methods given by Anderberg (1973, page 140).

4 This is Ward's method. Clusters are formed so as to minimize the increase in the within-cluster sums of squares. The distance between two clusters is the increase in these sums of squares if the two clusters were merged. A method for computing this distance from a squared Euclidean distance matrix is given by Anderberg (1973, pages 142-145).

In general, single linkage will yield long thin clusters while complete linkage will yield clusters that are more spherical. Average linkage and Ward's linkage tend to yield clusters that are similar to those obtained with complete linkage.

Routine **CLINK** produces a unique representation of the binary cluster tree via the following three conventions; the fact that the tree is unique should aid in interpreting the clusters. First, when two clusters are joined and each cluster contains two or more data points, the cluster that was initially formed with the smallest level (in **CLEVEL**) becomes the left son. Second, when a cluster containing more than one data point is joined with a cluster containing a single data point, the cluster with the single data point becomes the right son. Finally, when two clusters containing only one object are joined, the cluster with the smallest cluster number becomes the right son.

Comments

1. Workspace may be explicitly provided, if desired, by use of **C2INK/DC2INK**. The reference is:

   ```
   CALL C2INK (NPT, IMETH, IDIST, DIST, LDDIST, CLEVEL, ICLSON, ICRSON, IPTR,
       ICLUS, CWT, CSUM)
   ```

   The additional arguments are as follows:

   IPTR — Integer work vector of length **NPT**.

   ICLUS — Integer work vector of length **NPT**.

   CWT — Work vector of length **NPT**. Not used if **IMETH** = 0 or 1.

   CSUM — Work vector of length **NPT**. Not used if **IMETH** = 0 or 1.

2. The clusters corresponding to the original data points are numbered from 1 to **NPT**. The **NPT** - 1 clusters formed by merging clusters are numbered **NPT** + 1 to **NPT** + (**NPT** - 1).

3. Raw correlations, if used as similarities, should be made positive and transformed to a distance measure. One such transformation can be performed by specifying **IDIST** = 2 in **CLINK**.

4. The user may cluster either variables or observations in **CLINK** since a dissimilarity matrix, not the original data, is used. Routine **CDIST** may be used to compute the matrix **DIST**.

Routine **TREEP** (see Chapter 16, "Line Printer Graphics") in the graphics chapter can be used to obtain a line printer plot of the clustering tree. Routine **CNUMB** can be used to obtain the cluster number assigned to each of the original clusters when a specified number of clusters is desired.

Example

In the following example, the average distance within clusters method is used to perform a hierarchical cluster analysis of the Fisher iris data. Routine **GDATA** (see Chapter 19, "Utilities") first used to obtain the Fisher iris data. The example is typical in that after the program obtains the data, routine **CDIST** computes the distance matrix (**DIST**) prior to calling **CLINK**.

```
      USE GDATA_INT
      USE CDIST_INT
      USE CLINK_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    IDATA, IMETH, IPRINT, IROW, ISCALE, LDDIST, LDX,&
                 NCOL, NPT, NROW, NVAR
      PARAMETER  (IDATA=3, IMETH=2, IPRINT=0, IROW=1, ISCALE=1, &
                 NCOL=5, NROW=150, NVAR=4, LDX=NROW, &
                 NPT=NROW, LDDIST=LDX)
!
      INTEGER    I, ICLSON(NROW-1), ICRSON(NROW-1), IND(4), NOUT, &
                 NXCOL, NXROW
      REAL       CLEVEL(NROW-1), DIST(LDDIST,LDDIST), X(LDX,NCOL)
!
      DATA IND/2, 3, 4, 5/
!
      CALL GDATA (IDATA, X, NXROW, NXCOL)
!                                Compute the distances
      CALL CDIST (X, DIST, IND=IND, ISCALE=ISCALE)
!                                Clustering
      CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON, IMETH=IMETH)
!                                Print some results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99996) (I,I=1,149,15)
      WRITE (NOUT,99997) (CLEVEL(I),I=1,149,15)
      WRITE (NOUT,99998) (ICLSON(I),I=1,149,15)
      WRITE (NOUT,99999) (ICRSON(I),I=1,149,15)
!
99996 FORMAT (' OBS # ', 10I6)
99997 FORMAT (' CLEVEL', 10F6.2)
99998 FORMAT (' ICLSON', 10I6)
99999 FORMAT (' ICRSON', 10I6)
!
      END
```

Output

```
 OBS #      1    16    31    46    61    76    91   106   121   136
 CLEVEL  0.00  0.17  0.23  0.27  0.31  0.37  0.41  0.48  0.60  0.78
```

```
    ICLSON   143   153    17   140    53   198   186   218   261   249
    ICRSON   102    29     6   113    51    91   212   243   266   262
```

CNUMB

Computes cluster membership for a hierarchical cluster tree.

Required Arguments

NODE — Number of data points clustered. (Input)

ICLSON — Vector of length `NODE - 1` containing the left son cluster numbers. (Input)
Cluster `NODE + I` is formed by merging clusters `ICLSON(I)` and `ICRSON(I)`.

ICRSON — Vector of length `NODE - 1` containing the right son cluster numbers. (Input)
Cluster `NODE + I` is formed by merging clusters `ICLSON(I)` and `ICRSON(I)`.

K — Desired number of clusters. (Input)

ICLUS — Vector of length `NODE` containing the cluster membership of each observation. (Output)
Observation `I` is in cluster `ICLUS(I)` when `K` clusters are specified.

NCLUS — Vector of length `K` containing the number of observations in each cluster. (Output)

FORTRAN 90 Interface

Generic: `CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)`
Specific: The specific interface name is `CNUMB`.

FORTRAN 77 Interface

Single: `CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)`

Description

Given a fixed number of clusters (*K*) and the cluster tree (vectors `ICRSON` and `ICLSON`) produced by the hierarchical clustering algorithm (see routine `CLINK`), routine `CNUMB` determines the cluster membership of each observation. The routine `CNUMB` first determines the root nodes for the *K* distinct subtrees forming the *K* clusters and then traverses each subtree to determine the cluster membership of each observation. The routine `CNUMB` also returns the number of observations found in each cluster.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2UMB. The reference is:

   ```
   CALL C2UMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS, IPT)
   ```

   The additional argument is:

   IPT — Work vector of length 2 * NODE.

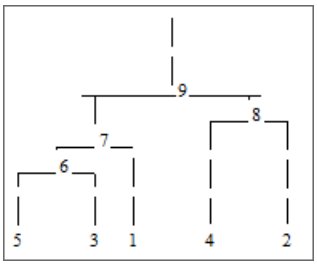
2. Informational errors

Type	Code	Description
4	1	The tree structure specified by ICLSON and ICRSON is invalid because an attempt to assign an observation to more than one cluster is being made.
4	2	The tree structure specified by ICLSON and ICRSON is incorrect because an observation is not assigned to a cluster.

Examples

Example 1

In the following example, cluster membership for $K = 2$ clusters is found for the displayed cluster tree. The output vector ICLUS contains the cluster numbers for each observation.




```
      USE CNUMB_INT
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    K, NODE
      PARAMETER  (K=2, NODE=5)
!
      INTEGER    ICLSON(NODE-1), ICLUS(NODE), ICRSON(NODE-1), NCLUS(K)
!
      DATA ICLSON/5, 6, 4, 7/
      DATA ICRSON/3, 1, 2, 8/
!                                Compute cluster membership
      CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)
!                                Print output
```

```
      CALL WRIRN ('ICLUS', ICLUS, 1, NODE, 1)
      CALL WRIRN ('NCLUS', NCLUS, 1, K, 1)
!
      END
```

Output

```
      ICLUS
1   2   3   4   5
1   2   1   2   1

      NCLUS
1   2
3   2
```

Example 2

This example illustrates the typical usage of **CNUMB**. The Fisher iris data (see routine **GDATA**, Chapter 19, "Utilities", is clustered. First the distance between the irises are computed using routine **CDIST**. The resulting distance matrix is then clustered using routine **CLINK**. The cluster membership for 5 clusters is then obtained via routine **CNUMB** using the output from **CLINK**. The need for 5 clusters can be obtained either by theoretical means or by examining a cluster tree. Because the cluster tree is too large to be included in this example, the call to routine **TREEP** (see Chapter 16, "Line Printer Graphics") that would ordinarily print the cluster tree has been commented in the example code. The cluster membership for each of the iris observations is printed.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    IDATA, IPRINT, IROW, K, &
                 LDDIST, LDX, NCOL, NODE, NODEX, NROW, NVAR
      PARAMETER  (IDATA=3, IPRINT=0, IROW=1, K=5, LDDIST=150, &
                  LDX=150, NCOL=5, NODE=150, NODEX=5, NROW=150, NVAR=4)
!
      INTEGER    I, ICLSON(NROW-1), ICLUS(NODE), ICRSON(NROW-1), &
                 IND(4), J, NCLUS(K), NSCALE, NXCOL, NXROW
      REAL       AMAX1, CLEVEL(NROW-1), DIST(LDDIST,LDDIST), RN, &
                SCALE(2), X(LDX,NCOL)
      CHARACTER  NODENM(NODE)*7
      INTRINSIC  AMAX1
!
      DATA IND/2, 3, 4, 5/
      DATA NSCALE/1/
      DATA SCALE/0.0, 3.5/
!                                Get IRIS data.
      CALL GDATA (IDATA, X, NXROW, NXCOL)
!                                Compute the dissimilarities.
      CALL CDIST (X, DIST, IND=IND)
!                                Make sure each distance is unique,
!                                then copy the upper triangle matrix
!                                to the lower triangle matrix.
      CALL RNSET (4)
      DO 20  I=1, NODE
         DO 10  J=I + 1, NODE
            RN = RNUNF()
```

```
               DIST(I,J) = AMAX1(0.0,DIST(I,J)+(0.001*RN))
      10    CONTINUE
            DIST(I,I) = 0.0
            CALL SCOPY (I-1, DIST(1:,I), 1, DIST(I:,1), LDDIST)
      20 CONTINUE
!                                  The initial clustering
      CALL CLINK (DIST, CLEVEL, ICLSON, ICRSON)
!                                  Print the tree.
      NODENM(1) = 'DEFAULT'
!     CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM)
!                                  Compute membership for 5 clusters
      CALL CNUMB (NODE, ICLSON, ICRSON, K, ICLUS, NCLUS)
!                                  Print output
      CALL WRIRN ('ICLUS', ICLUS, 1, NODE, 1)
      CALL WRIRN ('NCLUS', NCLUS, 1, K, 1)
!
      END
```

Output

```
                                    ICLUS
  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
  5  5  5  5  5  5  5  5  5   5   5   5   5   5   5   5   5   5   5   5

  21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
   5  5  5  5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5

  41 42 43 44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
   5  5  5  5   5   5   5   5   5   5   2   2   2   2   2   2   2   1   2   2

  61 62 63 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
   1  2  2  2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2

  81 82 83 84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
   2  2  2  2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   1

  100 101 102 103  104  105  106  107  108  109  110  111  112  113  114  115
    2   2   2   2    2    2    2    3    2    2    2    2    2    2    2    2

  116 117 118 119  120  121  122  123  124  125  126  127  128  129  130  131
    2   2   4   2    2    2    2    2    2    2    2    2    2    2    2    2

  132 133 134 135  136  137  138  139  140  141  142  143  144  145  146  147
    4   2   2   2    2    2    2    2    2    2    2    2    2    2    2    2

  148  149  150
    2    2    2

         NCLUS
  1    2    3    4    5
  4   93    1    2   50
```

KMEAN

Performs a *K*-means (centroid) cluster analysis.

Required Arguments

X — NOBS by NCOL matrix containing the observations to be clustered. (Input)
The only columns of X used are those indicated by IND and possibly IFRQ and/or IWT.

CM — K by NVAR matrix containing, on input, the cluster seeds, i.e., estimates for the cluster centers, and the cluster means on output. (Input/Output)
The cluster seeds must be unique.

SWT — K by NVAR matrix containing the sum of the weights used to compute each cluster mean. (Output)
Missing observations are excluded from SWT.

IC — Vector of length NOBS containing the cluster membership for each observation. (Output)

NC — Vector of length K containing the number of observations in each cluster. (Output)

WSS — Vector of length K containing the within sum of squares for each cluster. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

NVAR — Number of variables to be used in computing the metric. (Input)
Default: NVAR = size (CM,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means all frequencies are 1. For positive IFRQ, column number IFRQ of X contains the nonnegative frequencies.
Default: IFRQ = 0.

IWT — Weighting option. (Input)
IWT = 0 means all weights are 1. For positive IWT, column number IWT contains the nonnegative weights.
Default: IWT = 0.

IND — Vector of length NVAR containing the columns of X to be used in computing the metric. (Input)
In the usual case in which X is the data matrix, no observation has multiple frequency, and unequal weighting is not desired, IND = (1, 2, 3, ..., NVAR).
By default, IND(I) = (I)

K — Number of clusters. (Input)
Default: K = size (CM,1).

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 is usually sufficient.
Default: MAXIT = 30.

LDCM — Leading dimension of CM exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCM = size (CM,1).

LDSWT — Leading dimension of SWT exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSWT = size (SWT,1).

FORTRAN 90 Interface

Generic: CALL KMEAN (X, CM, SWT, IC, NC, WSS [, ...])
Specific: The specific interface names are S_KMEAN and D_KMEAN.

FORTRAN 77 Interface

Single: CALL KMEAN (NOBS, NCOL, NVAR, X, LDX, IFRQ, IWT, IND, K, MAXIT, CM, LDCM, SWT, LDSWT, IC, NC, WSS)
Double: The double precision name is DKMEAN.

Description

Routine **KMEAN** is an implementation of Algorithm AS 136 by Hartigan and Wong (1979). It computes *K*-means (centroid) Euclidean metric clusters for an input matrix starting with initial estimates of the *K* cluster means. Routine **KMEAN** allows for missing values (coded as NaN, "not a number") and for weights and frequencies.

Let *p* = **NVAR** denote the number of variables to be used in computing the Euclidean distance between observations. The idea in *K*-means cluster analysis is to find a clustering (or grouping) of the observations so as to minimize the total within-cluster sums of squares. In this case, the total sums of squares within each cluster is computed as the sum of the centered sum of squares over all nonmissing values of each variable. That is,

$$\phi = \sum_{i=1}^{K}\sum_{j=1}^{p}\sum_{m=1}^{n_i} f_{v_{im}} w_{v_{im}} \delta_{v_{im},j}\left(x_{v_{im},j} - \overline{x}_{ij} \right)^2$$

where v_{Im} denotes the row index of the *m*-th observation in the *i*-th cluster in the matrix **X**; n_{i} is the number of rows of **X** assigned to group *i*; *f* denotes the frequency of the observation; *w* denotes its weight; **δ** is zero if the *j*-th variable on observation v_{Im} is missing, otherwise **δ** is one; and

$$\overline{x}_{ij}$$

is the average of the nonmissing observations for variable *j* in group *i*. This method sequentially processes each observation and reassigns it to another cluster if doing so results in a decrease in the total within-cluster sums of squares. The user in referred to Hartigan and Wong (1979) or Hartigan (1975) for the details.

Comments

1. Workspace may be explicitly provided, if desired, by use of **K2EAN/DK2EAN**. The reference is:

   ```
   CALL K2EAN (NOBS, NCOL, NVAR, X, LDX, IFRQ, IWT, IND, K, MAXIT, CM, LDCM, SWT,
       LDSWT, IC, NC, WSS, IC2, NCP, D, ITRAN, LIVE)
   ```

   The additional arguments are as follows:

   IC2 — Work vector of length **NOBS**.

   NCP — Work vector of length **K**.

   D — Work vector of length **NOBS**.

   ITRAN — Work vector of length **K**.

   LIVE — Work vector of length **K**.

2. Informational Error

Type	Code	Description
3	1	Convergence did not occur within MAXIT iterations.

Example

This example performs *K*-means cluster analysis on Fisher's iris data, which is first obtained via routine **GDATA** (see Chapter 19, "Utilities"). The initial cluster seed for each iris type is an observation known to be in the iris type.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    IPRINT, K, LDCM, LDSWT, LDX, NCOL, NOBS, NV, NVAR
          PARAMETER  (IPRINT=0, K=3, NCOL=5, NOBS=150, NV=5, NVAR=4, &
                      LDCM=K, LDSWT=K, LDX=NOBS)
    !
          INTEGER    IC(NOBS), IND(NVAR), NC(K), NXCOL, NXROW
          REAL       CM(K,NVAR), SWT(K,NVAR), WSS(K), X(NOBS,NV)
    !
          DATA IND/2, 3, 4, 5/
    !
          CALL GDATA (3, X, NXROW, NXCOL)
    !                                  Copy the cluster seeds into CM
          CALL SCOPY (NVAR, X(1:,2), LDX, CM(1:,1), LDCM)
          CALL SCOPY (NVAR, X(51:,2), LDX, CM(2:,1), LDCM)
          CALL SCOPY (NVAR, X(101:,2), LDX, CM(3:,1), LDCM)
    !
          CALL KMEAN (X, CM, SWT, IC, NC, WSS, IND=IND)
    !
          CALL WRRRN ('CM', CM)
          CALL WRRRN ('SWT', SWT)
          CALL WRIRN ('IC', IC, 1, NOBS, 1)
          CALL WRIRN ('NC', NC, 1, K, 1)
          CALL WRRRN ('WSS', WSS, 1, K, 1)
          END
```

Output

```
                  CM
          1       2       3       4
1    5.006   3.428   1.462   0.246
2    5.902   2.748   4.394   1.434
3    6.850   3.074   5.742   2.071

                  SWT
          1       2       3       4
1    50.00   50.00   50.00   50.00
2    62.00   62.00   62.00   62.00
3    38.00   38.00   38.00   38.00

                                        IC
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
1  1  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
    21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
     1  1  1  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

    41 42 43 44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
     1  1  1  1   1   1   1   1   1   1   2   2   3   2   2   2   2   2   2   2

    61 62 63 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
     2  2  2  2   2   2   2   2   2   2   2   2   2   2   2   2   2   3   2   2

    81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
     2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2

    100 101 102 103  104  105  106  107  108  109  110  111  112  113  114  115
      2   3   2   3    3    3    3    2    3    3    3    3    3    3    2    2

    116 117 118 119  120  121  122  123  124  125  126  127  128  129  130  131
      3   3   3   3    2    3    2    3    2    3    3    2    2    3    3    3
    132 133 134 135  136  137  138  139  140  141  142  143  144  145  146  147
      3   3   2   3    3    3    3    2    3    3    3    2    3    3    3    2

    148  149  150
      3    3    2

         NC
     1    2    3
    50   62   38

              WSS
        1       2       3
    15.15   39.82   23.88
```

Sampling

Routines

12.1 Sampling Routines

Usage Notes

The routines for inferences regarding proportions require only counts as the input data. The other routines described in this chapter require the actual data. Since the amount of data may be quite large, these routines allow for the data to be input in small quantities (or even to be deleted after it has already been passed to the subroutine). This is accomplished by means of the processing option parameter, **IDO**, and an indicator of the number of observations being passed in, **NROW**. **IDO** has the following meaning:

IDO	Action
0	This is the only invocation of the subroutine for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to the subroutine will be made. Initialization and updating for the data are performed.
2	This is an intermediate invocation of the subroutine, and updating for the data is performed.
3	This is the final invocation of the routine. Updating for the data and any wrap-up computations are performed.

NROW can be positive or negative or zero. Its absolute value is the number of sample values being input. If **NROW** is negative, it is assumed that the observations being input have already been input once and now it is desired to delete them from the analysis. When **IDO** is 3, **NROW** can be set to 0. In this case, only postprocessing is performed; no accumulation of statistics is done. This allows input of summary statistics rather than the actual data. See Example 2 in the documentation for the routine SMPSR.

There are other variables used by several routines in this chapter that have a common meaning in all routines:

Y — The variable of interest.

X — The auxiliary variable.

NSAMP — The sample size.

NPOP — The population size.

CONPER — Confidence level.

STAT — Output statistics.

For stratified sampling, the following variables are often used:

NSTRAT — Number of strata.

NROWS — Vector with elements like **NROW** for strata.

NSAMPS — The strata sample sizes.

NPOPS — The population sizes for strata.

YBARS — The strata sample means.

YVARS — The strata sample variances.

SMPPR

Computes statistics for inferences regarding the population proportion and total given proportion data from a simple random sample.

Required Arguments

NINT — Number of sample units in the class of interest, for the population (or subpopulation) of interest. (Input)

NSAMP — Number of units in the entire random sample. (Input)

NPOP — Number of units in the population. (Input)

CONPER — Confidence level for two-sided interval estimates, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).

STAT — Vector of length 10 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the proportion.
2	Estimate of the total.
3	Variance estimate of the proportion estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the proportion.
6	Upper confidence limit for the proportion.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient or variation of the total estimate. Not defined if NINT = 0.
10	Indicator of the distribution used to approximate the hypergeometric distribution for the confidence interval calculations. If STAT(10) = 0, then the normal is used. If STAT(10) = 1, then the Poisson is used. If STAT(10) = 2, then the binomial is used.

FORTRAN 90 Interface

Generic: CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)

Specific: The specific interface names are S_SMPPR and D_SMPPR.

FORTRAN 77 Interface

Single: CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)

Double: The double precision name is DSMPPR.

Description

The routine SMPPR computes point and interval estimates for the population proportion and total from a simple random sample. The simplest and most common case for which this routine is appropriate is one in which the population sampled contains two or more classes, and it is desired to estimate the proportion of the population falling into a particular class ("class of interest"). The data required by SMPPR consist of counts of the number of sample items in the class of interest, the sample size, and the population size. If there are more than two classes in the population, some of the classes may not be of interest.

Since the hypergeometric distribution is the appropriate probability model for the sampling for proportions in a finite population without replacement, exact confidence limits could be computed using that distribution. For populations with sizes that occur in practice (more than a hundred, often in the thousands or even millions), the confidence limits can be approximated very well by use of the normal, the binomial, or the Poisson distribution. Routine SMPPR uses one of these distributions in setting confidence limits, following the guidelines in the table on page 58 of Cochran (1977).

Examples

Example 1

The first example is from Cochran (1977, page 52). A simple random sample of size 200 was drawn from a list of 3042 names and addresses. Verification of the addresses in the sample showed 38 to be wrong. The objective is to estimate the total number of incorrect addresses.

```
      USE UMACH_INT
      USE SMPPR_INT
      INTEGER    NINT, NOUT, NPOP, NSAMP
      REAL       CONPER, SQRT, STAT(10), STDP, STDT
      INTRINSIC  SQRT
!
      CALL UMACH (2, NOUT)
      NINT   = 38
      NSAMP  = 200
```

```
      NPOP   = 3042
      CONPER = 0.0
      CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)
      STDP = SQRT(STAT(3))
      STDT = SQRT(STAT(4))
      WRITE (NOUT,99999) STAT(1), STAT(2), STDP, STDT, STAT(9)
99999 FORMAT (' Estimate of proportion bad:              ', F5.3, /,&
             ' Estimate of total bad:                  ', F5.0, /, &
             ' Standard deviation estimate, proportion: ', F5.3, /, &
             ' Standard deviation estimate, total:      ', F5.1, /, &
             ' Coefficient of variation:                ', F5.1,'%')
      END
```

Output

```
Estimate of proportion bad:              0.190
Estimate of total bad:                    578.
Standard deviation estimate, proportion: 0.027
Standard deviation estimate, total:       81.8
Coefficient of variation:                 14.1%
```

Example 2

The next example is also from Cochran (1977, page 68). A simple random sample of size 200 from 2000 colleges showed 120 colleges to be in favor of a certain proposal, 57 to be opposed, and 23 to have no opinion. We wish to estimate the number of colleges, out of the 2000, that favor the proposal.

```
      USE UMACH_INT
      USE SMPPR_INT

      IMPLICIT   NONE
      INTEGER    NINT, NOUT, NPOP, NSAMP
      REAL       CONPER, STAT(10)
!
      CALL UMACH (2, NOUT)
      NINT   = 120
      NSAMP  = 200
      NPOP   = 2000
      CONPER = 95.0
      CALL SMPPR (NINT, NSAMP, NPOP, CONPER, STAT)
      WRITE (NOUT,99999) STAT(2), STAT(7), STAT(8)
99999 FORMAT (' Estimate of number in favor:   ', F5.0, /, ' 95% ', &
             'confidence interval: (', F5.0, ',', F5.0, ')')
      END
```

Output

```
Estimate of number in favor:   1200.
95% confidence interval: (1066.,1334.)
```

SMPPS

Computes statistics for inferences regarding the population proportion and total given proportion data from a stratified random sample.

Required Arguments

NINTS — Vector of length NSTRAT containing the observed number of units in each stratum from the class of interest. (Input)

NSAMPS — Vector of length NSTRAT containing the sample size in each stratum. (Input)

NPOPS — Vector of length NSTRAT containing the population in the strata. (Input)
If the population strata sizes are not known, estimates must be entered in their place.

PROPOR — Vector of length NSTRAT containing the within-strata proportion estimates. (Output)

STAT — Vector of length 10 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the proportion.
2	Estimate of the total.
3	Variance estimate of the proportion estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the proportion.
6	Upper confidence limit for the proportion.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the total estimate.
10	Variance estimate of the proportion estimate assuming that sampling was simple random instead of stratified random.

Optional Arguments

NSTRAT — Number of strata into which the sample is divided. (Input)
In the vectors of length NSTRAT, the elements are all ordered in the same way.
Default: NSTRAT = size (NINTS,1).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPPS (NINTS, NSAMPS, NPOPS, PROPOR, STAT [, ...])
Specific: The specific interface names are S_SMPPS and D_SMPPS.

FORTRAN 77 Interface

Single: CALL SMPPS (NSTRAT, NINTS, NSAMPS, NPOPS, CONPER, PROPOR, STAT)
Double: The double precision name is DSMPPS.

Description

Routine **SMPPS** computes point and interval estimates for the population proportion and total from a stratified random sample. If the strata are formed so that the proportions differ greatly from one stratum to the next, considerable gain in statistical efficiency can be realized by use of stratified sampling (see Cochran 1977, page 107).

Let N_h be the number in the population in the *h*-th stratum, let n_h be the number in the sample from the *h*-th stratum, let a_h be the number of the class of interest in the sample from the *h*-th stratum, let *N* be the population size (ΣN_h), let p_h be the proportion in the *h*-th stratum, a_h/n_h, and let *L* be the number of strata. Then, the estimate of the proportion is

$$p_{st} = \sum_{h=1}^{L} \frac{N_h a_h}{N n_h}$$

and the estimate of the variance is

$$v\left(p_{st}\right) = \frac{1}{N^2} \sum_{h=1}^{L} N_h \left(N_h - n_h\right) \frac{p_h\left(1 - p_h\right)}{n_h - 1}$$

The confidence intervals are computed using a normal approximation.

Example

This example is an artificial modification of an example used in routine **SMPPR**, which is from Cochran (1977, page 52). A list of 3042 names and addresses was built by an experienced secretary and a part-time student worker. The secretary entered 1838 names and addresses, and the student entered the remainder. Samples of size 100 were taken from the names entered by each. Verification of the addresses in the sample from the secretary's work showed 12 to be wrong, and verification of the student's sample showed 26 to be wrong. The objective is to estimate the total number of incorrect addresses.

```
          USE UMACH_INT
          USE SMPPS_INT

          IMPLICIT   NONE
          INTEGER    NSTRAT
          PARAMETER  (NSTRAT=2)
    !
          INTEGER    NINTS(NSTRAT), NOUT, NPOPS(NSTRAT), NSAMPS(NSTRAT)
          REAL       CONPER, PROPOR(NSTRAT), SQRT, STAT(10), STDP, STDSRS, &
                     STDT
          INTRINSIC  SQRT
    !
          CALL UMACH (2, NOUT)
          NINTS(1)  = 12
          NINTS(2)  = 26
          NSAMPS(1) = 100
```

```
      NSAMPS(2) = 100
      NPOPS(1)  = 1838
      NPOPS(2)  = 1204
      CONPER    = 0.0
!
      CALL SMPPS (NINTS, NSAMPS, NPOPS, PROPOR, STAT, CONPER=CONPER)
!
      STDP   = SQRT(STAT(3))
      STDT   = SQRT(STAT(4))
      STDSRS = SQRT(STAT(10))
!
      WRITE (NOUT,99999) STAT(1), STAT(2), STDP, STDT, STAT(9), STDSRS
99999 FORMAT (' Estimate of proportion bad:              ', F7.3, /, &
            ' Estimate of total bad:                   ', F4.0, /, &
            ' Standard deviation estimate, proportion: ', F7.3, /, &
            ' Standard deviation estimate, total:      ', F5.1, /, &
            ' Coefficient of variation:                ', F5.1, &
            '%', /, ' Std. dev. under simple random sampling:  ', &
            F7.3)
      END
```

Output

```
 Estimate of proportion bad:               0.175
 Estimate of total bad:                  534.
 Standard deviation estimate, proportion:   0.025
 Standard deviation estimate, total:       77.4
 Coefficient of variation:                 14.5%
 Std. dev. under simple random sampling:    0.027
```

SMPRR

Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, or inferences regarding the population ratio given a simple random sample.

Required Arguments

NROW — The absolute value of NROW is the number of observations currently input in X and Y. (Input)
NROW may be positive, zero, or negative. Negative -NROW means delete the NROW rows of data from the analysis.

X — Vector of length |NROW| containing the data for the auxiliary variable in the random sample. (Input)

Y — Vector of length |NROW| containing the data for the variable of interest in the random sample. (Input)
The value of Y(I) corresponds to that of X(I).

NPOP — Size of the population (number of pairs of elements in the sampled population). (Input)

XMEAN — Population mean of the auxiliary variable. (Input)
XMEAN is not used if IOPT = 1.

STAT — Vector of length 20 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3)

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the ratio.
10	Variance estimate of the estimate of the ratio. The population mean of the auxiliary variable is used in STAT(10) if the mean is known; otherwise, the sample estimate of the population mean is used.

11	Lower confidence limit for the ratio.
12	Upper confidence limit for the ratio.
13	Estimate (expressed as a percentage) of the coefficient of variation of the mean, total, and ratio and regression coefficient estimates that are defined, as controlled by IOPT. The standard deviation in the numerator of this quantity has been divided by the square root of the sample size. The coefficients of variation in STAT(14) and STAT(15) use the sample standard deviations without that divisor.
14	Estimate (expressed as a percentage) of the coefficient of variation of the auxiliary variable.
15	Estimate (expressed as a percentage) of the coefficient of variation of the variable of interest.
16	Sample mean of the auxiliary variable.
17	Sample mean of the variable of interest.
18	Estimate of the regression coefficient.
19	Sample size.
20	Number of pairs in the sample with one or both values missing.

STAT(1) through STAT(8) and STAT(13) are undefined when IOPT = 1. STAT(9) through STAT(12) are undefined when IOPT = 2 or 3. STAT(18) is defined only when IOPT = 3. The elements of STAT that are undefined due to IOPT or an error are set to NaN (not a number).

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPRR for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPRR will be made. Initialization and updating for the data in X and Y are performed.
2	This is an intermediate invocation of SMPRR and updating for the data in X and Y is performed.
3	This is the final invocation of this routine. Updating for the data in X and Y, and wrap-up computations are performed.

IOPT — Estimation option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Ratio estimation is used for inference about the population mean, total, and ratio.
1	The population mean of the auxiliary variable is not used, and only inference about the population ratio is desired.
2	Regression estimation with preassigned regression coefficient (in COEF) is used for inference about the population mean and total.
3	Regression estimation with estimated regression coefficient (returned in STAT(18)) is used for inference about the population mean and total.

COEF — Reassigned regression coefficient. (Input)
COEF is used only when IOPT = 2.
Default: COEF = 1.0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed, hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT [, ...])
Specific: The specific interface names are S_SMPRR and D_SMPRR.

FORTRAN 77 Interface

Single: CALL SMPRR (IDO, NROW, X, Y, NPOP, IOPT, XMEAN, COEF, CONPER, STAT)
Double: The double precision name is DSMPRR.

Description

Routine SMPRR computes point and interval estimates for the population mean, total, and (optionally) ratio or regression coefficient, using a simple random sample of a variable of interest and an auxiliary variable. Routine SMPRR allows various options for the estimation techniques, which are discussed in Chapters 3, 6, and 7 of Cochran (1977). Let

$$\overline{x} \text{ and } \overline{y}$$

be the sample means of the auxiliary variable and the variable of interest, respectively. Let

$$\overline{X}$$

be the population mean of the auxiliary variable. Then, the ratio estimate of the population mean is

$$\overline{y}_R = \frac{\overline{y}}{\overline{x}}\overline{X}$$

The linear regression estimate of the population mean is

$$\overline{y}_{lr} = \overline{y} + b\left(\overline{X} - \overline{x}\right)$$

where *b* is the regression coefficient, which can be either preassigned, based on previous knowledge, or estimated from the data using least squares. The least-squares estimate of *b* is

$$\frac{\sum_{i=1}^{n}\left(y_i - \overline{y}\right)\left(x_i - \overline{x}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}$$

The confidence limits for the mean and for the total are computed using the normal approximation. If the coefficient of variation of either variable exceeds 10%, then this approximation may not be very accurate.

The parameters **IDO** and **NROW** allow either all or part of the data to be brought in.

Examples

The data for these examples come from Cochran (1977, Table 6.1, page 152). The variable of interest is the population of large U.S. cities in 1930; the auxiliary variable is the 1920 population of the same cities. There are 196 (**NPOP**) cities that are sampled (that is, that are in the population of interest). (Note that the word "population" is being used in two ways in this discussion.) The total 1920 population of these cities is 22,919 (**XMEAN** = 116.934). There are 49 cities in the sample. The data can be seen in the **DATA** statements in the programs below (actual values are 1000 times greater). There are no "missing data"; therefore, the sample size, **STAT**(19), is 49. Because the coefficient of variation is larger than 10%, **SMPRR** produces an informational "warning error" message in each example. When the coefficient of variation is larger than 10% (generally speaking), the confidence limits computed using the normal approximation are likely to be shorter than the actual limits at the same confidence level.

Example 1

In this example, ratio estimation is used, as on page 151 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NROW
```

```
      PARAMETER  (NROW=49)
!
      INTEGER    I, NOUT, NPOP
      REAL       COEF, CONPER, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                All data are input at once.
!                                Ratio estimation.
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,17), STAT(19), STAT(20)
99999 FORMAT (/, '                    RATIO ESTIMATION', /, &
            ' Mean estimate = ', F8.1, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.1, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.1, &
            ',', F8.1, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Ratio estimate = ', F8.3, '      Vhat of ' &
            , 'ratio = ', F8.4, /, ' Confidence limits for ratio ', &
            F8.3, ',', F8.3, /, ' Coefficient of variation of mean ', &
            'estimate = ', F8.1, /, ' CV of X =   ', F8.1, &
            '               CV of Y = ', F8.1, /, ' Mean of X = ', &
            F8.1, '            Mean of Y = ', F8.1, /, ' Sample size ' &
            , '= ', F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
 *** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
 ***          both of the variables exceeds 10%.  The confidence limits,
 ***          which are computed using a normal approximation, may not be
 ***          very accurate.

                    RATIO ESTIMATION
Mean estimate =    144.9      Total estimate =  28397.1
Vhat of mean  =      9.5      Vhat of total  = 364860.1
Confidence limits for mean     138.8,   150.9
Confidence limits for total  27213.3, 29581.0
Ratio estimate =    1.239      Vhat of ratio =   0.0007
Confidence limits for ratio    1.187,   1.291
Coefficient of variation of mean estimate =      2.1
CV of X =       89.3              CV of Y =     96.3
Mean of X =    103.1            Mean of Y =    127.8
Sample size =     49.0      Number missing =      0.0
```

Example 2

In this example, regression estimation with an estimated coefficient is used, as in Exercise 7.3 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NROW
      PARAMETER  (NROW=49)
!
      INTEGER    I, IOPT, NOUT, NPOP
      REAL       CONPER, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                 All data are input at once.
!                                 Regression estimation, with estimated
!                                 coefficient (Cochran, Exercise 7.3)
      IOPT = 3
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT)
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,8), (STAT(I),I=13,20)
99999 FORMAT (/, '                    REGRESSION ESTIMATION', /, &
            ' Mean estimate = ', F8.1, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.1, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.1, &
            ',', F8.1, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Coefficient of variation of mean ', &
            'estimate = ', F8.1, /, ' CV of X =   ', F8.1, &
            '               CV of Y = ', F8.1, /, ' Mean of X = ', &
            F8.1, '            Mean of Y = ', F8.1, /, ' Estimated ', &
            'regression coefficient = ', F8.1, /, ' Sample size = ', &
            F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
*** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
***          both of the variables exceeds 10%.  The confidence limits,
***          which are computed using a normal approximation, may not be
***          very accurate.

                     REGRESSION ESTIMATION
Mean estimate =    143.8      Total estimate =  28177.4
Vhat of mean  =      8.6      Vhat of total  = 329372.3
Confidence limits for mean     138.0,   149.5
Confidence limits for total  27052.6, 29302.3
Coefficient of variation of mean estimate =      2.0
CV of X =       89.3              CV of Y =     96.3
Mean of X =    103.1           Mean of Y =    127.8
Estimated regression coefficient =      1.2
```

```
    Sample size =     49.0      Number missing =      0.0
```

Example 3

In this example, regression estimation with a preassigned coefficient is used, as in Exercise 7.4 of Cochran (1977).

```
      USE SMPRR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NROW
      PARAMETER  (NROW=49)
!
      INTEGER    I, IOPT, NOUT, NPOP
      REAL       COEF, STAT(20), X(NROW), XMEAN, Y(NROW)
!
      DATA X/76., 138., 67., 29., 381., 23., 37., 120., 61., 387., &
          93., 172., 78., 66., 60., 46., 2., 507., 179., 121., 50., &
          44., 77., 64., 64., 56., 40., 40., 38., 136., 116., 46., &
          243., 87., 30., 71., 256., 43., 25., 94., 43., 298., 36., &
          161., 74., 45., 36., 50., 48./
      DATA Y/80., 143., 67., 50., 464., 48., 63., 115., 69., 459., &
          104., 183., 106., 86., 57., 65., 50., 634., 260., 113., &
          64., 58., 89., 63., 77., 142., 60., 64., 52., 139., 130., &
          53., 291., 105., 111., 79., 288., 61., 57., 85., 50., 317., &
          46., 232., 93., 53., 54., 58., 75./
      DATA NPOP/196/, XMEAN/116.934/
!                                 All data are input at once.
!                                 Regression estimation, with assigned
!                                 coefficient (Cochran, Exercise 7.4)
      IOPT = 2
      COEF = 1.0
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT, COEF=COEF)
!                                 Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,8), (STAT(I),I=13,17), STAT(19), &
                         STAT(20)
99999 FORMAT (/, '                     REGRESSION ESTIMATION, FIXED ', &
            'COEF', /, ' Mean estimate = ', F8.1, '      Total ', &
            'estimate = ', F8.1, /, ' Vhat of mean  = ', F8.1, &
            '      Vhat of total  = ', F8.1, /, ' Confidence limits ' &
            , 'for mean  ', F8.1, ',', F8.1, /, ' Confidence limits ' &
            , 'for total ', F8.1, ',', F8.1, /, ' Coefficient of ', &
            'variation of mean estimate = ', F8.1, /, ' CV of X =   ' &
            , F8.1, '               CV of Y = ', F8.1, /, ' Mean of ' &
            , 'X = ', F8.1, '            Mean of Y = ', F8.1, /, &
            ' Sample size = ', F8.1, '      Number missing = ', F8.1)
      END
```

Output

```
 *** WARNING  ERROR 7 from SMPRR.  The coefficient of variation of one or
 ***          both of the variables exceeds 10%.  The confidence limits,
 ***          which are computed using a normal approximation, may not be
 ***          very accurate.

       REGRESSION ESTIMATION, FIXED COEF
 Mean estimate =    141.6      Total estimate =  27751.1
```

```
    Vhat of mean  =     12.5      Vhat of total  = 481977.4
    Confidence limits for mean     134.6,   148.5
    Confidence limits for total  26390.4, 29111.8
    Coefficient of variation of mean estimate =      2.5
    CV of X =       89.3              CV of Y =     96.3
    Mean of X =    103.1            Mean of Y =    127.8
    Sample size =     49.0      Number missing =      0.0
```

SMPRS

Computes statistics for inferences regarding the population mean and total using ratio or regression estimation given continuous data from a stratified random sample.

Required Arguments

NROWS — Vector of length NSTRAT in which |NROWS(I)| is the number of items from the I-th stratum currently input in X and Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for NROWS(I) means delete the -NROWS(I) elements of the I-th stratum in X and Y from the analysis.

X — Vector containing the data for the auxiliary variable in the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in X. The first |NROWS(1)| elements of X are from the first stratum, and so on.

Y — Vector containing the data for the variable of interest in the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first stratum, and so on. The value of Y(I) corresponds to that of X(I).

NPOPS — Vector of length NSTRAT containing the sizes of the population in the strata. (Input)
The entries in NSTRAT must be ordered in correspondence with the ordering of strata in the other vectors. If the population strata sizes are not known, estimates must be entered in their place.

XMEANS — Vector of length NSTRAT containing, for each stratum, the population mean of the auxiliary variate, provided ITOPT = 0. (Input)
If ITOPT = 1, only XMEANS(1) is defined and it must contain the population mean of the auxiliary variate.

COEFS — Vector of length NSTRAT containing the ratio estimates or the regression coefficients. (Input, if IOPT = 1; Output, if IOPT = 0 or 2 and IDO = 0 or 1; Input/Output, if IOPT = 0 or 2 and IDO = 2 or 3)
If IOPT = 0, COEFS contains ratio estimates. When ITOPT = 0, COEFS contains the estimate of the ratio for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the combined estimate of the ratio. If IOPT = 1, COEFS contains preassigned regression coefficients. When ITOPT = 0, COEFS contains the preassigned regression coefficient for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the preassigned regression coefficient common to all strata. If IOPT = 2, COEFS contains estimated regression coefficients. When ITOPT = 0, COEFS contains the estimated regression coefficient for each stratum. When ITOPT = 1, only COEFS(1) is defined and contains the estimated regression coefficient common to all strata.

XBARS — Vector of length NSTRAT containing the strata means for the auxiliary variable. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

XVARS — Vector of length NSTRAT containing the within-strata variances of the auxiliary variable. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

XCVS — Vector of length NSTRAT containing the within-strata coefficients of variation for the auxiliary variable. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

YBARS — Vector of length NSTRAT containing the strata means for the variable of interest. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

YVARS — Vector of length NSTRAT containing the within-strata variances of the variable of interest. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

YCVS — Vector of length NSTRAT containing the within-strata coefficients of variation for the variable of interest. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

XYCOVS — Vector of length NSTRAT containing the within-strata covariances of the auxiliary variable and the variable of interest. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

NSAMPS — Vector of length NSTRAT containing the number of nonmissing observations from each stratum. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

STAT — Vector of length 12 containing the resulting statistics. (Output)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the coefficient of variation for the mean and total estimate.
10	Unstratified mean of the auxiliary variate.
11	Unstratified mean of the variable of interest.
12	The number of pairs in the sample that had one or both values missing.

Optional Arguments

IDO — Processing option. (Input)

Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPRS for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPRS will be made. Initialization and updating for the data in X and Y are performed.
2	This is an intermediate invocation of SMPRS, and updating for the data in X and Y is performed.
3	This is the final invocation of this routine. Updating for the data in X and Y and wrap-up computations are performed.

NSTRAT — Number of strata into which the population is divided. (Input)

In the vectors of length NSTRAT, the elements are all ordered in the same way. That is, the first stratum is always the first, the second is always the second, and so on.

Default: NSTRAT = size (NROWS,1).

IOPT — Estimation option. (Input)

Default: IOPT = 0.

IOPT	Action
0	Ratio estimation used for inference about the population mean and total.
1	Regression estimation used with the preassigned regression coefficient(s) contained in COEFS.
2	Regression estimation used with the regression coefficient(s) estimated from the data.

ITOPT — Estimation technique option. (Input)
Default: ITOPT = 0.

ITOPT	Action
0	Separate ratio or regression estimation.
1	Combined ratio or regression estimation.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT [, ...])
Specific: The specific interface names are S_SMPRS and D_SMPRS.

FORTRAN 77 Interface

Single: CALL SMPRS (IDO, NSTRAT, NROWS, X, Y, NPOPS, IOPT, ITOPT, XMEANS, CONPER, COEFS, XBARS, XVARS, XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT)
Double: The double precision name is DSMPRS.

Description

Routine SMPRS computes point and interval estimates for the population mean and total from a stratified random sample of a variable of interest and an auxiliary variable. Routine SMPRS allows for either ratio estimation, regression estimation with preassigned coefficients, or regression estimation with estimated coefficients.

This routine follows the standard methods discussed in Chapters 6 and 7 of Cochran (1977). The statistics are similar to those discussed in the documentation for routine SMPRR, except that they are computed from stratified data. The option parameter IOPT allows selection of either ratio or regression estimation, and the parameter ITOPT allows selection of separate or combined estimators. "Separate" estimators means that each stratum is allowed to have different ratios or regression coefficients, while "combined" means these are assumed to be the same over all strata.

The confidence limits for the mean and for the total are computed using the normal approximation. If the coefficient of variation of either variable exceeds 10%, then this approximation may not be very accurate.

The parameters IDO and NROW allow either all or part of the data to be brought in at one time.

Examples

Example 1

In the following example, we use a stratified sample from the data in Table 5.1 of Cochran (1977), which consists of the 1920 and the 1930 population (in 1000's) of 64 cities in the United States. The objective is to estimate the mean and total 1930 population of the 64 cities, using a sample of size 24 of the 1920 and 1930 populations. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. The same example is also used to illustrate routine SMPSS, except here we have an auxiliary variable.

In this example, separate ratio estimation is used.

```
      USE SMPRS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, NOUT, NPOPS(NSTRAT), NROWS(NSTRAT), NSAMPS(NSTRAT)
      REAL       COEFS(NSTRAT), STAT(12), X(24), &
                 XBARS(NSTRAT), XCVS(NSTRAT), XMEANS(NSTRAT), &
                 XVARS(NSTRAT), XYCOVS(NSTRAT), Y(24), YBARS(NSTRAT), &
                 YCVS(NSTRAT), YVARS(NSTRAT)
!
      DATA X/773., 748., 734., 577., 507., 438., 415., 401., 387., &
          381., 324., 315., 258., 237., 235., 216., 201., 179., 136., &
          132., 118., 118., 106., 104./
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
          464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
          170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
!                                  All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
```

```
!                                  Use separate ratio estimation.
      XMEANS(1) = 521.8
      XMEANS(2) = 165.4
!
      CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, &
                  XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,9), STAT(12), COEFS
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.5, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' C. V.       = ', F8.2, '      Number ', &
            'missing = ', F8.1, /, ' Estimated ratios = ', 2F10.3)
      END
```

Output

```
Mean estimate =  315.511      Total estimate =  20192.7
Vhat of mean  = 55.56254      Vhat of total  = 227584.2
Confidence limits for mean   300.901, 330.120
Confidence limits for total  19257.7, 21127.7
C. V.       =     2.36      Number missing =      0.0
Estimated ratios =      1.225     1.255
```

Example 2

In the following example, we use a stratified sample from the data in Table 5.1 of Cochran (1977), which consists of the 1920 and the 1930 population (in 1000's) of 64 cities in the United States. The objective is to estimate the mean and total 1930 population of the 64 cities, using a sample of size 24 of the 1920 and 1930 populations. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. The same example is also used to illustrate routine SMPSS, except here we have an auxiliary variable.

In this example, regression estimation is used, and it is assumed that the regression equation is the same in the two strata.

```
      USE SMPRS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, IDO, IOPT, ITOPT, NOUT, NPOPS(NSTRAT), &
                 NROWS(NSTRAT), NSAMPS(NSTRAT)
      REAL       COEFS(NSTRAT), STAT(12), X(24), &
                 XBARS(NSTRAT), XCVS(NSTRAT), XMEANS(1), &
                 XVARS(NSTRAT), XYCOVS(NSTRAT), Y(24), YBARS(NSTRAT), &
                 YCVS(NSTRAT), YVARS(NSTRAT)
!
      DATA X/773., 748., 734., 577., 507., 438., 415., 401., 387., &
          381., 324., 315., 258., 237., 235., 216., 201., 179., 136., &
          132., 118., 118., 106., 104./
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
```

```
            464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
            170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
!                                  All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
!                                  Use combined regression estimation.
      IOPT      = 2
      ITOPT     = 1
      XMEANS(1) = 254.5
!
      CALL SMPRS (NROWS, X, Y, NPOPS, XMEANS, COEFS, XBARS, XVARS, &
                  XCVS, YBARS, YVARS, YCVS, XYCOVS, NSAMPS, STAT, &
                  IOPT=IOPT, ITOPT=ITOPT)
!                                  Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,9), STAT(12), COEFS(1)
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.5, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' C. V.       = ', F8.1, '      Number ', &
            'missing = ', F8.1, /, ' Estimated combined regression ', &
            'coefficient = ', F8.3)
      END
```

Output

```
 Mean estimate =  315.517      Total estimate =  20193.1
 Vhat of mean  = 54.84098      Vhat of total  = 224628.6
 Confidence limits for mean   301.003, 330.031
 Confidence limits for total  19264.2, 21122.0
 C. V.       =      2.3      Number missing =      0.0
 Estimated combined regression coefficient =    1.175
```

SMPSC

Computes statistics for inferences regarding the population mean and total using single stage cluster sampling with continuous data.

Required Arguments

NROWS — Vector of length NCLSTR in which |NROWS(I)| is the number of items from the I-th cluster currently input in Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for NROWS(I) means delete the -NROWS(I) elements of the I-th cluster in Y from the analysis.

Y — Vector containing the cluster sample. (Input)
The observations within any one cluster must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first cluster, and so on.

NCLPOP — Number of clusters in the sampled population. (Input)

NPOP — Number of elements in the population (sum of all the cluster sizes in the population). (Input)
NPOP is not required when IOPT = 3.

CLMEAN — Vector of length NCLSTR containing the cluster means. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

CLVAR — Vector of length NCLSTR containing the within-cluster variances. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

NSAMPS — Vector of length NCLSTR containing the number of nonmissing observations from each cluster. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

STAT — Vector of length 11 containing the resulting statistics. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.

I	STAT(I)
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the mean and total estimate.
10	The total sample size.
11	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSC for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSC will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSC and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NCLSTR — Number of clusters into which the sample is divided. (Input)
In the vectors of length NCLSTR, the elements are all ordered in the same way. That is, the first cluster is always the first, the second always the second, and so on.
Default: NCLSTR = size (NROWS,1).

IOPT — Estimation option. (Input)
Default: IOPT = 0.

IOPT	Action
0	Ratio-to-size estimation is used.
1	Unbiased estimation is used.
2	Probability-proportional-to-size estimation is used and all clusters in population are of known size.
3	Probability-proportional-to-size estimation is used and the cluster sizes are known only approximately or a measure of cluster size other than the number of elements per cluster is to be used.

SIZE — If IOPT = 3, vector of length NCLSTR containing a measure of cluster size for each cluster in the sample. (Input)
The sampled cluster size measures must be ordered in correspondence with the ordering of clusters in Y. SIZE is required only when IOPT = 3.

TSIZE — If IOPT = 3, measure of total size of all clusters in the population. (Input)
TSIZE is required only when IOPT = 3.
Default: TSIZE = 1.0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPSC (NROWS, Y, NCLPOP, NPOP, CLMEAN, CLVAR, NSAMPS, STAT [, ...])
Specific: The specific interface names are S_SMPSC and D_SMPSC.

FORTRAN 77 Interface

Single: CALL SMPSC (IDO, NCLSTR, NROWS, Y, IOPT, NCLPOP, NPOP, SIZE, TSIZE, CONPER, CLMEAN, CLVAR, NSAMPS, STAT)
Double: The double precision name is DSMPSC.

Description

Routine SMPSC computes point and interval estimates for the population mean and total from a single-stage cluster sample. The routine uses the standard methods discussed in Chapters 9 and 9A of Cochran (1977). The sample means for the individual clusters are accumulated in CLMEAN, and the corrected sums of squares are accumulated in CLVAR. In the postprocessing phase, the quantities in STAT are computed using the cluster statistics in CLMEAN, CLVAR, and NSAMPS. The parameters IDO and NROWS allow either all or part of the data to be brought in at one time.

Following the notation of Cochran (1977), let N be the number of clusters in the population, let M_i be the number of elements in the i-th cluster unit, let M_0 be the total number of elements in the population, let y_{ij} be the j-th element in the i-th cluster, and let n be the number of clusters in the sample. Any of three different estimators of the population total may be useful. An unbiased estimate of the total is

$$\frac{N}{n}\sum_{i=1}^{n}\sum_{j=1}^{M_i} y_{ij}$$

The ratio-to-size estimate is

$$M_0 \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} M_i}$$

The probability-proportional-to-size estimate is

$$\frac{M_0}{n}\sum_{i=1}^{n}\left(\frac{y_i}{M_i}\right)$$

The confidence limits for the mean and total are computed using the normal approximation.

Example

In this example, we have a sample of two clusters from a population that contains 20 clusters. The sizes of the clusters in the sample are four and six, and there is a total of 100 elements in the population.

```
      USE SMPSC_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NCLSTR
      PARAMETER  (NCLSTR=2)
!
      INTEGER    NCLPOP, NOUT, NPOP, NROWS(NCLSTR), NSAMPS(NCLSTR)
      REAL       CLMEAN(NCLSTR), CLVAR(NCLSTR), SIZE(NCLSTR), &
                 STAT(11), TSIZE, Y(10)
!
      DATA Y/2.7, 5.1, 4.3, 2.8, 1.9, 6.2, 4.8, 5.1, 7.2, 6.5/
!
      NCLPOP = 20
      NPOP   = 100
!                                All data are input at once.
      NROWS(1) = 4
      NROWS(2) = 6
      CALL SMPSC (NROWS, Y, NCLPOP, NPOP, CLMEAN, CLVAR, NSAMPS, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
            , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
```

```
                ',', F8.1, /, ' C. V.       = ', F8.1, '%', /, &
                ' Sample size =   ', F8.0, '      Number missing = ', &
                F8.0)
        END
```

Output

```

    Mean estimate =    4.660      Total estimate =    466.0
    Vhat of mean  =    0.504      Vhat of total  =   5035.5
    Confidence limits for mean     3.269,   6.051
    Confidence limits for total    326.9,   605.1
    C. V.         =    15.2%
    Sample size =        10.      Number missing =       0.
```

SMPSR

Computes statistics for inferences regarding the population mean and total, given data from a simple random sample.

Required Arguments

NROW — The absolute value of NROW is the number of rows of data currently input in Y. (Input)
NROW may be positive, zero, or negative. Negative -NROW means delete the NROW rows of data from the analysis.

Y — Vector of length |NROW| containing the sample data. (Input)

NPOP — Size of the (full) population. (Input)

STAT — Vector of length 11 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Within-sample variance estimate.
4	Variance estimate of the mean estimate.
5	Variance estimate of the total estimate.
6	Lower confidence limit for the mean.
7	Upper confidence limit for the mean.
8	Lower confidence limit for the total.
9	Upper confidence limit for the total.
10	The sample size.
11	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSR for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSR will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSR, and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

IOPT — Subpopulation option. (Input)

If IOPT = 0, no subpopulation is assumed. If IOPT = 1, the input data come from a subpopulation ("domain of study") of unknown size.

Default: IOPT = 0.

NSAMPO — Size of the sample from the full population, if a subpopulation is sampled (that is, if IOPT = 1). (Input)

Default: NSAMPO = ABS (NROW).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)

A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPSR (NROW, Y, NPOP, STAT [, ...])

Specific: The specific interface names are S_SMPSR and D_SMPSR.

FORTRAN 77 Interface

Single: CALL SMPSR (IDO, NROW, Y, NPOP, IOPT, NSAMPO, CONPER, STAT)

Double: The double precision name is DSMPSR.

Description

Routine SMPSR computes point and interval estimates for the population mean and total from a simple random sample of one variable. The routine uses the standard methods discussed in Chapter 2 of Cochran (1977). The sample mean is accumulated in STAT(1) and the corrected sum of squares is accumulated in STAT(3). In the postprocessing phase, STAT(3) is divided by the sample size minus one, and then the other quantities in STAT are computed. The parameters IDO and NROW allow either all or part of the data to be brought in at one time.

By use of IOPT and NSAMPO, SMPSR can also be used to analyze data from a subpopulation or "domain of study". (See Cochran 1977, pages 34-38.) In the case of a subpopulation, only the estimates relating to the subpopulation total differ from the corresponding estimates when no subpopulation is assumed. Of course, if a subpopulation is of known size, it should be considered the full population.

Examples

Example 1

This example uses artificial data to illustrate a simple use of SMPSR to compute point and interval estimates of the population mean and total. The sample size is 15, from a population of size 150.

```
      USE SMPSR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NROW
      PARAMETER  (NROW=15)
!
      INTEGER    NOUT, NPOP, NSAMPO
      REAL       STAT(11), Y(NROW)
!
      DATA Y/21., 14., 17., 22., 19., 21., 20., 15., 24., 28., 20., &
          17., 16., 22., 19./
!
      NPOP   = 150
!                                All data are input at once.
!                                No subpopulation is assumed.
      CALL SMPSR (NROW, Y, NPOP, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F8.1, /, ' Within-sample variance estimate = ', F8.3, /, &
            ' VHAT of mean  = ', F8.5, '      VHAT of total  = ', &
            F8.1, /, ' Confidence limits for mean  ', F8.3, &
            ',', F8.3, /, ' Confidence limits for total ', F8.1, &
            ',', F8.1, /, ' Sample size = ', F8.1, '      Number ', &
            'missing = ', F8.0)
      END
```

Output

```
    Mean estimate =   19.667      Total estimate =   2950.0
    Within-sample variance estimate =   13.238
    VHAT of mean  =  0.79429      VHAT of total  =  17871.4
    Confidence limits for mean    17.755,  21.578
    Confidence limits for total   2663.3,  3236.7
    Sample size =     15.0      Number missing =       0.
```

Example 2

This example is a problem of estimation in a subpopulation described on page 37 of Cochran (1977). The example illustrates how the IDO and NROW parameters can be used to allow input other than the actual data. Cochran gives only the sample total and uncorrected sum of squares, so these values are transformed to the mean and corrected sum of squares prior to input as STAT(1) and STAT(3).

```
          USE SMPSR_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    IDO, IOPT, NOUT, NPOP, NROW, NSAMPO
          REAL       SQRT, STAT(11), Y(1)
          INTRINSIC  SQRT
    !
          NPOP   = 2422
    !                                There are 180 items in the complete
    !                                sample, but only a subpopulation is
    !                                of interest.
          IOPT   = 1
          NSAMPO = 180
    !                                For this example, STAT is
    !                                initialized as if the data
    !                                have been already processed and only
    !                                the postprocessing computations are
    !                                to be done.  There are 152 items of
    !                                interest in the sample.  The sample
    !                                total is 343.5 and the uncorrected
    !                                sum of squares is 1491.38.
    !                                STAT(1) is initialized to the sample
    !                                mean by dividing the total by the
    !                                sample size, and STAT(3) is
    !                                initialized to the corrected sum of
    !                                squares.
          STAT(1)  = 343.5/152.0
          STAT(3)  = 1491.38 - 152.0*STAT(1)**2
          STAT(10) = 152.0
          STAT(11) = 0.0
          IDO      = 3
          NROW     = 0
          CALL SMPSR (NROW, Y, NPOP, STAT, IDO=IDO, IOPT=IOPT, NSAMPO=NSAMPO)
    !                                Print results
          CALL UMACH (2, NOUT)
          WRITE (NOUT,99999) STAT(2), SQRT(STAT(5))
    99999 FORMAT ('      Total estimate = ', F8.1, /, '      Standard ', &
                'deviation of the estimate = ', F8.1)
          END
```

Output

```
    Total estimate =   4622.0
    Standard deviation of the estimate =    375.3
```

SMPSS

Computes statistics for inferences regarding the population mean and total, given data from a stratified random sample.

Required Arguments

NROWS — Vector of length NSTRAT in which |NROWS(I)| is the number of items from the I-th stratum currently input in Y. (Input)
Each element of NROWS may be positive, zero, or negative. A negative value for NROWS(I) means delete the -NROWS(I) elements of the I-th stratum in Y from the analysis.

Y — Vector containing the stratified random sample. (Input)
The observations within any one stratum must appear contiguously in Y. The first |NROWS(1)| elements of Y are from the first stratum, and so on.

NPOPS — Vector of length NSTRAT containing the sizes of the population in the strata. (Input)
The entries must be ordered in correspondence with the ordering of strata in the other vectors. If the population strata sizes are not known, estimates must be entered in their place.

YBARS — Vector of length NSTRAT containing the strata means. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

YVARS — Vector of length NSTRAT containing the within-strata variances. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

NSAMPS — Vector of length NSTRAT containing the number of nonmissing observations from each stratum. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3.)

STAT — Vector of length 13 containing the resulting statistics. (Output, if IDO = 0 or 1; input/output, if IDO = 2 or 3.)
These are:

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance estimate of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.

6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.
8	Upper confidence limit for the total.
9	Estimate of the coefficient of variation of the mean and total estimates.
10	Number of degrees of freedom associated with the variance estimates of the mean and total estimates. When IVOPT = 1, STAT(10) contains an effective number of degrees of freedom determined according to the Satterthwaite approximation.
11	Variance estimate of the mean estimate assuming that sampling was simple random instead of stratified random.
12	Pooled estimate of the common variance, when IVOPT = 0. If IVOPT = 1, STAT(12) is not defined.
13	The number of missing values.

Optional Arguments

IDO — Processing option. (Input)

Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPSS for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPSS will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPSS, and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NSTRAT — Number of strata into which the population is divided. (Input)

In the vectors of length NSTRAT, the elements are all ordered in the same way. That is, the first stratum is always the first, the second always the second, and so on.

Default: NSTRAT = size (NROWS,1).

IVOPT — Within-stratum variance assumption indicator. (Input)

If IVOPT = 0, the true within-stratum variance is assumed constant, and a pooled estimate of that variance is returned in STAT(12). If IVOPT = 1, separate within-strata variance estimates are assumed.

Default: IVOPT = 0.

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)
A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).
Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPSS (NROWS, Y, NPOPS, YBARS, YVARS, NSAMPS, STAT[, ...])
Specific: The specific interface names are S_SMPSS and D_SMPSS.

FORTRAN 77 Interface

Single: CALL SMPSS (IDO, NSTRAT, NROWS, Y, NPOPS, IVOPT, CONPER, YBARS, YVARS, NSAMPS, STAT)
Double: The double precision name is DSMPSS.

Description

Routine SMPSS computes point and interval estimates for the population mean and total from a stratified random sample of one variable. The routine uses the standard methods discussed in Chapters 5 and 5A of Cochran (1977). The sample means for the individual strata are accumulated in YBARS, and the corrected sums of squares are accumulated in YVARS. In the postprocessing phase, the quantities in STAT are computed using the strata statistics in YBARS, YVARS, and NSAMPS. The parameters IDO and NROWS allow either all or part of the data to be brought in at one time.

Comments

Information Error

Type	Code	Description
4	1	The population size for each stratum is equal to one.

Example

In this example, we use a stratified sample from the data in Table 5.1 of Cochran (1977): the 1930 population (in 1000's) of 64 cities in the United States. The 64 cities are the "population", and our objective is to estimate the mean and total number of inhabitants in these 64 cities. There are two strata: the largest 16 cities and the remaining cities. We use stratified sampling with equal sample sizes. To choose the random sample, we use routine **RNSRI** (see Chapter 18, "Random Number Generation"), as follows:

```
      USE RNSET_INT
      USE RNSRI_INT

      IMPLICIT    NONE
      INTEGER     ISEED, NSAMP, NPOP, INDEX(12)
      NSAMP = 12
      NPOP = 16
      ISEED = 123457
      CALL RNSET(ISEED)
      CALL RNSRI(NPOP, INDEX)
      WRITE(*, *) INDEX
      NPOP = 48
      CALL RNSRI(NPOP, INDEX)
      WRITE(*, *) INDEX
      END
```

This yields the population indices {2, 3, 4, 6, 8, 10, 11, 12, 13, 14, 15, 16} for the first stratum and {4, 8, 10, 11, 13, 16, 29, 30, 36, 37, 45, 46} for the second stratum. The corresponding values from Table 5.1 are encoded in the program below.

```
      USE SMPSS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NSTRAT
      PARAMETER  (NSTRAT=2)
!
      INTEGER    I, IVOPT, NOUT, NPOPS(NSTRAT), NROWS(NSTRAT), &
                 NSAMPS(NSTRAT)
      REAL       STAT(13), Y(24), YBARS(NSTRAT), YVARS(NSTRAT)
!
      DATA Y/822., 781., 805., 1238., 634., 487., 442., 451., 459., &
          464., 400., 366., 302., 291., 272., 284., 270., 260., 139., &
          170., 154., 140., 163., 116./
!
      NPOPS(1) = 16
      NPOPS(2) = 48
      IVOPT    = 1
!                                All data are input at once.
      NROWS(1) = 12
      NROWS(2) = 12
      CALL SMPSS (NROWS, Y, NPOPS, YBARS, YVARS, NSAMPS, STAT, IVOPT=IVOPT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (STAT(I),I=1,11), STAT(13)
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
            F9.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
```

```
               , ' = ', F9.1, /, ' Confidence limits for mean  ', F8.3, &
               ',', F8.3, /, ' Confidence limits for total ', F8.1, &
               ',', F8.1, /, ' C. V.       = ', F8.1, '        Degrees ' &
               , 'of freedom = ', F8.1, /, ' SRS var. estimate = ', &
               F8.3, '  Number missing = ', F8.0)
         END
```

Output

```
 Mean estimate =  313.167      Total estimate =   20042.7
 Vhat of mean  =  264.703      Vhat of total  = 1084224.6
 Confidence limits for mean   279.180, 347.153
 Confidence limits for total  17867.5, 22217.8
 C. V.       =      5.2        Degrees of freedom =     19.6
 SRS var. estimate = 1288.075  Number missing =       0.
```

SMPST

Computes statistics for inferences regarding the population mean and total given continuous data from a two-stage sample with equisized primary units.

Required Arguments

NUNSAM — Number of primary units into which the sample is divided. (Input)

NELSAM — Number of elements in the sample in each sampled primary unit. (Input)

Y — Vector of length NOBS containing the elements of the two-stage sample. (Input)
The elements from each primary unit must occur contiguously within Y. Since there must be an equal number from each primary unit, Y must contain no missing values.

NUNPOP — Number of primary units in the sampled population. (Input)

NELPOP — Number of elements in each primary unit in the population. (Input)

PUMEAN — Vector of length NUNSAM containing the means of the primary units in the sample. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
The estimates are ordered in correspondence with the ordering of primary units in Y.

PUVAR — Vector of length NUNSAM containing the sample variances of the primary units in the sample. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)
The estimates are ordered in correspondence with the ordering of primary units in Y.

STAT — Vector of length 9 containing the resulting statistics. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2 or 3)

I	STAT(I)
1	Estimate of the mean.
2	Estimate of the total.
3	Variance of the mean estimate.
4	Variance estimate of the total estimate.
5	Lower confidence limit for the mean.
6	Upper confidence limit for the mean.
7	Lower confidence limit for the total.

I	STAT(I)
8	Upper confidence limit for the total.
9	Estimate (expressed as a percentage) of the coefficient of variation of the mean and total estimates.

Optional Arguments

IDO — Processing option. (Input)

Default: IDO = 0.

IDO	Action
0	This is the only invocation of SMPST for this data set, and all the data are input at once.
1	This is the first invocation, and additional calls to SMPST will be made. Initialization and updating for the data in Y are performed.
2	This is an intermediate invocation of SMPST, and updating for the data in Y is performed.
3	This is the final invocation of this routine. Updating for the data in Y and wrap-up computations are performed.

NOBS — The number of observations currently input in Y. (Input)

NOBS may be positive or zero. If NOBS = 0, IDO must equal 3, and only wrap-up computations are performed.

Default: NOBS = size (Y,1).

CONPER — Confidence level for two-sided interval estimate, in percent. (Input)

A CONPER percent confidence interval is computed; hence, CONPER must be greater than or equal to 0.0 and less than 100.0. CONPER is often 90.0, 95.0, or 99.0. For a one-sided confidence interval with confidence level ONECL, set CONPER = 100.0 - 2.0 * (100.0 - ONECL).

Default: CONPER = 95.0.

FORTRAN 90 Interface

Generic: CALL SMPST (NUNSAM, NELSAM, Y, NUNPOP, NELPOP, PUMEAN, PUVAR, STAT [, ...])

Specific: The specific interface names are S_SMPST and D_SMPST.

FORTRAN 77 Interface

Single: CALL SMPST (IDO, NUNSAM, NELSAM, NOBS, Y, NUNPOP, NELPOP, CONPER, PUMEAN, PUVAR, STAT)

Double: The double precision name is **DSMPST**.

Description

Routine **SMPST** computes point and interval estimates for the population mean and total from a two-stage sample with primary units that are all equal in size. A two-stage sample might be taken if each unit ("primary unit") in the population can be divided into smaller units. Primary units are selected first, and then those selected are subsampled. The routine uses the standard methods discussed in Chapter 10 of Cochran (1977). The sample means for the individual primary units are accumulated in **PUMEAN**, and the corrected sums of squares are accumulated in **PUVAR**. In the postprocessing phase, the quantities in **STAT** are computed using the primary unit statistics. The parameters **IDO** and **NOBS** allow either all or part of the data to be brought in at one time.

Following the notation of Cochran (1977), let *n* (**NUMSAM**) be the number of primary units in the sample, let *m* (**NELSAM**) be the number of elements (subunits) subsampled from each primary unit, let *N* (**NUMPOP**) be the total number of primary units in the population, let *M* (**NELPOP**) be the total number of elements in each primary unit (in the population), and let y_{ij} be the *j*-th element in the *i*-th primary unit. The sample mean per subunit in the *i*-th primary unit is

$$\bar{y}_i = \frac{1}{m}\sum_{j=1}^{m} y_{ij}$$

The estimate of the population mean is

$$\bar{\bar{y}} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}$$

The estimate of the variance of

$$\bar{\bar{y}} \text{ is } \frac{N-n}{nN(n-1)}\sum_{i=1}^{n}\left(\bar{y}_i - \bar{\bar{y}}\right)^2 + \frac{M-m}{m(m-1)MnN}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{ij} - \bar{y}_i\right)^2$$

Example

In this example, we have a sample of two primary units, with five subunits from each. The population consists of 10 primary units with 15 elements each.

```
      USE SMPST_INT
      USE UMACH_INT

      IMPLICIT   NONE
```

```
      INTEGER    NELPOP, NELSAM, NOBS, NOUT, NUNPOP, NUNSAM
      REAL       PUMEAN(2), PUVAR(2), STAT(9), Y(10)
!
      DATA Y/2.7, 5.1, 4.3, 2.8, 1.9, 6.2, 4.8, 5.1, 7.2, 6.5/
!
      NUNSAM = 2
      NELSAM = 5
      NOBS   = 10
      NUNPOP = 10
      NELPOP = 15
!                                All data are input at once.
      CALL SMPST (NUNSAM, NELSAM, Y, NUNPOP, NELPOP, &
                  PUMEAN, PUVAR, STAT)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STAT
99999 FORMAT (' Mean estimate = ', F8.3, '      Total estimate = ', &
             F8.1, /, ' Vhat of mean  = ', F8.3, '      Vhat of total ' &
             , ' = ', F8.1, /, ' Confidence limits for mean  ', F8.3, &
             ',', F8.3, /, ' Confidence limits for total ', F8.1, &
             ',', F8.1, /, ' C. V.       = ', F8.1, '%')
      END
```

Output

```
 Mean estimate =    4.660      Total estimate =    699.0
 Vhat of mean  =    1.370      Vhat of total  =  30823.7
 Confidence limits for mean     2.366,   6.954
 Confidence limits for total    354.9,  1043.1
 C. V.       =     25.1%
```

Survival Analysis, Life Testing, and Reliability

Routines

Usage Notes

The routines described in this chapter have primary application in the areas of reliability and life testing, but they may find application in any situation in which time is a variable of interest. Kalbfleisch and Prentice (1980), Elandt-Johnson and Johnson (1980), Lee (1980), Gross and Clark (1975), Lawless (1982), and Chiang (1968) are general references for discussing the models and methods used here.

Kaplan-Meier (product-limit) estimates of the survival distribution in a single population is available through routine KAPMR, and these can be printed using KTBLE. Routine TRNBL computes generalized Kaplan-Meier estimates. Routine PHGLM computes the parameter estimates in a proportional hazards model. Routine SVGLM fits any of several generalized linear models, and STBLE computes estimates of survival probabilities based on the same models. Routine ACTBL computes and (optionally) prints an actuarial table based either upon a cohort followed over time or a cross-section of a population.

KAPMR

Computes Kaplan-Meier estimates of survival probabilities in stratified samples.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRT — Column number in X containing the response variable. (Input)
For the *i*-th right-censored observation, X(*i*, IRT) contains the right-censoring time. Otherwise, X(*i*, IRT) contains the failure time. (See ICEN.)

SPROB — NOBS by 2 matrix. (Output)
SPROB(*i*, 1) contains the estimated survival probability at time X(*i*, IRT) in the *i*-th observation's stratum, while SPROB(*i*, 2) contains Greenwood's estimate of the standard deviation of this estimated probability. If the *i*-th observation contains censor codes out of range or if a variable is missing, then the corresponding elements of SPROB are set to missing (NaN, not a number). Similarly, if an element in SPROB is not defined, then it is set to missing.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for this observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

ICEN — Column number in X containing the censoring code for this observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).

If X(*i*, ICEN) is not 0 or 1, then the *i*-th observation is omitted from the analysis.

IGRP — Column number in X containing the stratum number for this observation. (Input)
If IGRP = 0, the data is assumed to be from one stratum. Otherwise, column IGRP of X contains a unique value for each stratum in the data. Kaplan-Meier estimates are computed within each stratum.
Default: IGRP = 0.

ISRT — Sorting option. (Input)
If ISRT = 1, column IRT of X is assumed to be sorted in ascending order within each stratum. Otherwise, a detached sort will be performed by KAPMR. If sorting is performed by KAPMR, all censored individuals are assumed to follow tied failures.
Default: ISRT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

NRMISS — Number of rows of data in X that contain any missing values. (Output)

FORTRAN 90 Interface

Generic: CALL KAPMR (X, IRT, SPROB [, ...])
Specific: The specific interface names are S_KAPMR and D_KAPMR.

FORTRAN 77 Interface

Single: CALL KAPMR (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT, SPROB, LDSPRO, NRMISS)
Double: The double precision name is DKAPMR.

Description

Routine **KAPMR** computes Kaplan-Meier (or product-limit) estimates of survival probabilities for a sample of failure times that possibly contain right censoring. A survival probability $S(t)$ is defined as 1 - $F(t)$, where $F(t)$ is the cumulative distribution function of the failure times (t). Greenwood's estimate of the standard errors of the survival probability estimates are also computed. (See Kalbfleisch and Prentice, 1980, pages 13 and 14.)

Let (t_i, δ_i), for i = 1, ..., n denote the failure/censoring times and the censoring codes for the n observations in a single sample. Here, t_i = **X**(i, **IRT**) is a failure time if δ_i is 0, where δ_i = **X**(i, **ICEN**). Also, t_i is a censoring time if δ_i is 1. Rows in X containing values other than 0 or 1 for δ_i are ignored. Let the number of observations in the sample that have not failed by time $s_{(i)}$ be denoted by $n_{(i)}$, where $s_{(i)}$ is an ordered (from smallest to largest) listing of the distinct failure times (censoring times are omitted). Then the Kaplan-Meier estimate of the survival probabilities is a step function, which in the interval from $s_{(i)}$ to $s_{(i+1)}$ (including the lower endpoint) is given by

$$\hat{S}(t) = \prod_{j=1}^{i} \left(\frac{n_{(j)} - d_{(j)}}{n_{(j)}} \right)$$

where $d_{(j)}$ denotes the number of failures occurring at time $s_{(j)}$. Note that one row of X may correspond to more than one failed (or censored) observation when the frequency option is in effect (**IFRQ** is not zero). The Kaplan-Meier estimate of the survival probability prior to time $s_{(1)}$ is 1.0, while the Kaplan-Meier estimate of the survival probability after the last failure time is not defined.

Greenwood's estimate of the variance of

$$\hat{S}(t)$$

in the interval from $s_{(i)}$ to $s_{(i+1)}$ is given as

$$\text{est. var}(\hat{S}(t)) = \hat{S}^2(t) \sum_{j=1}^{i} \frac{d_{(j)}}{n_{(j)}(n_{(j)} - d_{(j)})}$$

Routine **KAPMR** computes the single sample estimates of the survival probabilities for all samples of data included in X during a single call. This is accomplished through the **IGRP** column of X, which if present, must contain a distinct code for each sample of observations. If **IGRP** = 0, there is no grouping column, and all observations are assumed to be from the same sample.

When failures and right-censored observations are tied and the data are to be sorted by **KAPMR** (**ISRT** is not 1), **KAPMR** assumes that the time of censoring for the tied-censored observations is immediately after the tied failure (within the same sample). When the **ISRT** = 1 option is in effect, the data are assumed to be sorted from

smallest to largest according to column **IRT** of *X* within each stratum. Furthermore, a small increment of time is assumed (theoretically) to elapse between the failed and censored observations that are tied (in the same sample). Thus, when the **ISRT** = 1 option is in effect, the user must sort all of the data in *X* from smallest to largest according to column **IRT** (and column **IGRP**, if present). By appropriate sorting of the observations, the user can handle censored and failed observations that are tied in any manner desired.

Comments

1. Workspace may be explicitly provided, if desired, by use of **K2PMR/DK2PMR**. The reference is:

   CALL K2PMR (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT, SPROB, LDSPRO, NRMISS, IGP, IPERM, INDDR, IWK, WK, IPER)

   The additional arguments are as follows:

   IGP — Work vector of length **NOBS**.

   IPERM — Work vector of length **NOBS** + **NCOL**.

   INDDR — Work vector of length **NOBS**.

   IWK — Work vector of length max(**NOBS**, **NCOL**).

   WK — Work vector of length 2 * max(**NOBS**, **NCOL**).

   IPER — Work vector of length **NOBS**.

2. Missing values may occur in any of the columns of **X**. Any row of **X** that contains missing values in the **IRT**, **ICEN**, or **IFRQ** columns (when the **ICEN** and **IFRQ** columns are present) is omitted from the analysis. Missing values in the **IGRP** column, if present, are classified into an additional "missing" group.

Example

The following example is taken from Kalbfleisch and Prentice (1980, page 1). The first column in *X* contains the death/censoring times for rats suffering from vaginal cancer. The second column contains information as to which of two forms of treatment were provided, while the third column contains the censoring code. Finally, the fourth column contains the frequency of each observation. The product-limit estimates of the survival probabilities are computed for both groups with one call to **KAPMR**. In this example, the output in **SPROB** has been equivalenced with columns 5 and 6 of *X* so that the input and output matrices could be printed together. Routine **KAPMR** could have been called with the **ISRT** = 1 option in effect if the censored observations had been sorted with respect to the failure time variable.

```
      USE KAPMR_INT
      USE WRRRL_INT
      USE UMACH_INT

      IMPLICIT   NONE
```

```
      INTEGER    ICEN, IFRQ, IGRP, IRT, ISRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=3, IFRQ=4, IGRP=2, IRT=1, ISRT=0, LDSPRO=33, &
                 LDX=33, NCOL=6, NOBS=33)
!
      INTEGER    NOUT, NRMISS
      REAL       SPROB(LDSPRO,2), X(LDX,NCOL)
      CHARACTER  XLABEL(7)*6, YLABEL(1)*6
!
      EQUIVALENCE (X(1,5), SPROB)
!
      DATA XLABEL/'OBS', 'TIME', 'GROUP', 'CENSOR', 'FREQ', 'S-HAT', &
          'SE'/
      DATA YLABEL/'NUMBER'/
      DATA X/143, 164, 188, 190, 192, 206, 209, 213, 216, 220, 227, &
          230, 234, 246, 265, 304, 216, 244, 142, 156, 163, 198, 205, &
          232, 233, 239, 240, 261, 280, 296, 323, 204, 344, 18*5, &
          15*7, 16*0, 2*1, 13*0, 4*1, 2, 20*1, 2, 4, 3*1, 2*2, 3*1, &
          66*0/
!
      CALL KAPMR (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP, &
                  NRMISS=NRMISS)
!
      CALL WRRRL ('X/SPROB', X, YLABEL, XLABEL, FMT='(W10.6)')
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(//'' NRMISS = '', I5)') NRMISS
      END
```

Output

```
                                X/SPROB
 OBS     TIME       GROUP      CENSOR       FREQ       S-HAT          SE
   1   143.000      5.000       0.000      1.000      0.947       0.051
   2   164.000      5.000       0.000      1.000      0.895       0.070
   3   188.000      5.000       0.000      2.000      0.789       0.094
   4   190.000      5.000       0.000      1.000      0.737       0.101
   5   192.000      5.000       0.000      1.000      0.684       0.107
   6   206.000      5.000       0.000      1.000      0.632       0.111
   7   209.000      5.000       0.000      1.000      0.579       0.113
   8   213.000      5.000       0.000      1.000      0.526       0.115
   9   216.000      5.000       0.000      1.000      0.474       0.115
  10   220.000      5.000       0.000      1.000      0.414       0.115
  11   227.000      5.000       0.000      1.000      0.355       0.112
  12   230.000      5.000       0.000      1.000      0.296       0.108
  13   234.000      5.000       0.000      1.000      0.237       0.101
  14   246.000      5.000       0.000      1.000      0.158       0.093
  15   265.000      5.000       0.000      1.000      0.079       0.073
  16   304.000      5.000       0.000      1.000      0.000         NaN
  17   216.000      5.000       1.000      1.000      0.474       0.115
  18   244.000      5.000       1.000      1.000      0.237       0.101
  19   142.000      7.000       0.000      1.000      0.952       0.046
  20   156.000      7.000       0.000      1.000      0.905       0.064
  21   163.000      7.000       0.000      1.000      0.857       0.076
  22   198.000      7.000       0.000      1.000      0.810       0.086
  23   205.000      7.000       0.000      1.000      0.759       0.094
  24   232.000      7.000       0.000      2.000      0.658       0.105
  25   233.000      7.000       0.000      4.000      0.455       0.111
  26   239.000      7.000       0.000      1.000      0.405       0.110
  27   240.000      7.000       0.000      1.000      0.354       0.107
```

```
    28    261.000       7.000       0.000       1.000       0.304       0.103
    29    280.000       7.000       0.000       2.000       0.202       0.090
    30    296.000       7.000       0.000       2.000       0.101       0.068
    31    323.000       7.000       0.000       1.000       0.051       0.049
    32    204.000       7.000       1.000       1.000       0.810       0.086
    33    344.000       7.000       1.000       1.000         NaN         NaN

    NRMISS =     0
```

KTBLE

Prints Kaplan-Meier estimates of survival probabilities in stratified samples.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

IRT — Column number of X containing the response variable. (Input)
For the *i*-th right-censored observation, X(*i*, IRT) contains the right-censoring time. Otherwise, X(*i*, IRT) contains the failure time. See argument ICEN.

SPROB — NOBS by 2 matrix. (Input)
SPROB (*i*, 1) contains the estimated survival probability at time X(*i*, IRT) in the *i*-th observation's stratum, while SPROB(*i*, 2) contains Greenwood's estimate of the standard deviation of this estimated probability. SPROB will usually be computed by routine KAPMR. It may contain missing values after the last failed observation in each group.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
IFRQ = 0 means that all frequencies are 1.0. For positive IFRQ, column number IFRQ of X contains the frequencies.
Default: IFRQ = 0.

ICEN — Column number of X containing the censoring code for this observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

Code	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).

If X(*i*, ICEN) is not zero or one, then the *i*-th observation is omitted from the analysis.

IGRP — Column number of X containing the stratum number for this observation. (Input)
If IGRP = 0, the data are assumed to be from one stratum. Otherwise, column IGRP of X contains a unique value for each stratum in the data. Kaplan-Meier estimates are computed within each stratum.
Default: IGRP = 0.

ISRT — Sorting option. (Input)
If ISRT = 1, column IRT of X is assumed to be sorted in ascending order within each stratum. Otherwise, a detached sort will be performed by KTBLE. If sorting is performed by KTBLE, all censored observations are assumed to follow failing observations with the same response time in X (*i*, IRT).
Default: ISRT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

FORTRAN 90 Interface

Generic: CALL KTBLE (X, IRT, SPROB [, ...])
Specific: The specific interface names are S_KTBLE and D_KTBLE.

FORTRAN 77 Interface

Single: CALL KTBLE (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT, SPROB, LDSPRO)
Double: The double precision name is DKTBLE.

Description

Routine KTBLE prints life tables based upon the Kaplan-Meier estimates of the survival probabilities (see routine KAPMR). One table for each stratum is printed. In addition to the survival probabilities at each failure point, the following is also printed: the number of individuals remaining at risk, Greenwood's estimate of the standard errors for the survival probabilities, and the Kaplan-Meier log-likelihood. The Kaplan-Meier log-likelihood is computed as:

$$\ell = \sum_j d(j)\ln d(j) + \big(n(j) - d(j)\big)\ln\big(n(j) - d(j)\big) - n(j)\ln n(j)$$

where the sum is with respect to the distinct failure times *s*(*j*), *d*(*j*) is the number of failures occurring at time *s*(*j*), and *n*(*j*) is the number of observations that had not yet failed immediately prior to *s*(*j*). Note that sorting is performed by both KAPMR, and by routine KTBLE. The user may sort the data to be increasing in failure time and then use the ISRT = 1 option to avoid this double sorting.

Comments

1. Workspace may be explicitly provided, if desired, by use of K2BLE/DK2BLE. The reference is:

   CALL K2BLE (NOBS, NCOL, X, LDX, IRT, IFRQ, ICEN, IGRP, ISRT, SPROB, LDSPRO, ALGL, IPERM, INDDR, WK, WK1, IWK)

   The additional arguments are as follows:

   ALGL — Work vector of length NOBS that contains the log likelihoods of the Kaplan-Meier estimates. If the number of groups is known to be *m* or less, then ALGL can be of length *m*.

   IPERM — Work vector of length NOBS.

   INDDR — Work vector of length NOBS.

   WK — Work vector of length NOBS.

   WK1 — Work vector of length 2 * max(NOBS, NCOL).

   IWK — Work vector of length max(NOBS, NCOL).

2. Informational errors

Type	Code	Description
4	1	An invalid value for SPROB has been detected. The estimated survival probability must be between zero and one, inclusive, and nonincreasing with failure time within each group.
4	2	A negative frequency has been detected.
4	3	A missing value for SPROB has been detected but later failures occur. Missing values are not allowed prior to the last failed observation.

3. Missing values may occur in any of the columns of X. Any row of X that contains missing values in the IRT, ICEN, or IFRQ columns (when the ICEN and IFRQ columns are present) is omitted from the analysis. Missing values in the IGRP column, if present, are classified into an additional "missing" group.

Example

This example illustrates the typical use of KTBLE. First, routine KAPMR is used to compute the survival probabilities. This is followed by a call to KTBLE that performs the printing. The input data is given as:

143, 164, 188(2), 190, 192, 206, 209, 213, 216, 220, 227, 230, 234, 246, 265, 304, 216*, 244*,
142, 156, 163, 198, 205, 232(2), 233(4), 239, 240, 261, 280(2), 296(2), 323, 204*, 344*

where items marked with an * are right censored, and the frequency of each failure time, if different from 1, is given in parenthesis.

```
      USE KAPMR_INT
      USE KTBLE_INT

      IMPLICIT   NONE
      INTEGER    ICEN, IFRQ, IGRP, IRT, ISRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=3, IFRQ=4, IGRP=2, IRT=1, ISRT=0, LDSPRO=33, &
                 LDX=33, NCOL=4, NOBS=33)
!
      INTEGER    NRMISS
      REAL       SPROB(LDSPRO,2), X(LDX,NCOL)
!
      DATA X/143, 164, 188, 190, 192, 206, 209, 213, 216, 220, 227, &
          230, 234, 246, 265, 304, 216, 244, 142, 156, 163, 198, 205, &
          232, 233, 239, 240, 261, 280, 296, 323, 204, 344, 18*5, &
          15*7, 16*0, 2*1, 13*0, 4*1, 2, 20*1, 2, 4, 3*1, 2*2, 3*1/
!
      CALL KAPMR (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP)
!
      CALL KTBLE (X, IRT, SPROB, IFRQ=IFRQ, ICEN=ICEN, IGRP=IGRP)
      END
```

Output

```
               Kaplan Meier Survival Probabilities
                 For Group Value =      5.00000

Number      Number                  Survival     Estimated
at risk     Failing        Time  Probability    Std. Error
     19           1         143      0.94737       0.05123
     18           1         164      0.89474       0.07041
     17           2         188      0.78947       0.09353
     15           1         190      0.73684       0.10102
     14           1         192      0.68421       0.10664
     13           1         206      0.63158       0.11066
     12           1         209      0.57895       0.11327
     11           1         213      0.52632       0.11455
     10           1         216      0.47368       0.11455
      8           1         220      0.41447       0.11452
      7           1         227      0.35526       0.11243
      6           1         230      0.29605       0.10816
      5           1         234      0.23684       0.10145
      3           1         246      0.15789       0.09343
      2           1         265      0.07895       0.07279
      1           1         304      0.00000           NaN
```

```
Total number in group    =       19
Total number failing     =       17
Product Limit Likelihood =      -49.1692


               Kaplan Meier Survival Probabilities
                  For Group Value =      7.00000
Number      Number                  Survival     Estimated
at risk     Failing        Time  Probability    Std. Error
     21           1         142      0.95238       0.04647
     20           1         156      0.90476       0.06406
     19           1         163      0.85714       0.07636
     18           1         198      0.80952       0.08569
     16           1         205      0.75893       0.09409
     15           2         232      0.65774       0.10529
     13           4         233      0.45536       0.11137
      9           1         239      0.40476       0.10989
      8           1         240      0.35417       0.10717
      7           1         261      0.30357       0.10311
      6           2         280      0.20238       0.09021
      4           2         296      0.10119       0.06778
      2           1         323      0.05060       0.04928

Total number in group    =       21
Total number failing     =       19
Product Limit Likelihood =      -50.4277
```

TRNBL

Computes Turnbull's generalized Kaplan-Meier estimates of survival probabilities in samples with interval censoring.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

ILT — For interval-censored and left-censored observations, the column number in X that contains the upper endpoint of the failure interval. (Input)
See argument ICEN. If ILT = 0, left-censored and interval-censored observations cannot be input.

IRT — For interval-censored and right-censored observations, the column number in X that contains the lower endpoint of the failure interval. (Input)
See argument ICEN. IRT must not be zero.

NINTVL — Number of failure intervals found. (Output)

SPROB — NINTVL by 4 matrix. (Output)

Col.	Description
1	Lower endpoint of the failure interval
2	Upper endpoint of the failure interval
3	Estimated change in the survival probability density within the failure interval
4	Estimate of the survival probability for the interval

The estimated survival probability is a constant equal to SPROB(*i*, 4) from SPROB (*i*, 2) to SPROB(*i* + 1, 1). The estimated survival probability is 1 prior to SPROB(1, 1). The estimated survival probability is undefined in the interval SPROB(*i*, 1) to SPROB(*i*, 2). If the NINTVL-th interval is from SPROB(NINTVL, 1) to infinity, then SPROB(NINTVL, 2) is set to positive machine infinity.

ALGL — Optimized log-likelihood for the input data. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Frequency option. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed. For positive IFRQ, column number IFRQ contains the frequency of response for each observation.
Default: IFRQ = 0.

ICEN — Censoring code option. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. For positive ICEN, column number ICEN contains the censoring code for each observation. Valid censoring codes are:

Code	Meaning
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).
2	Left censored. The response is less than or equal to X(*i*, ILT).
3	Interval censored. The response is greater than X(*i*, IRT), but less than or equal to X(*i*, ILT).

MAXIT — Maximum number of iterations. (Input)
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the relative change in the log-likelihood from one iteration to the next is less than EPS. EPS = 0.00001 is typical.
Default: EPS = 0.00001.

IPRINT — Printing option. (Input)
IPRINT = 0 means that no printing is performed. IPRINT = 1 means that printing is performed.
Default: IPRINT = 0.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
If LDSPRO is less than NINTVL, only the first LDSPRO intervals are returned in SPROB.
Default: LDSPRO = size (SPROB,1).

NRMISS — Number of rows of data in **X** that contain missing values. (Output)
Any row of **X** that contains missing values in the **ILT**, **IRT**, **ICEN**, or **IFRQ** columns (when the **ILT**, **ICEN** or **IFRQ** is positive) is omitted from the analysis.

FORTRAN 90 Interface

Generic: **CALL TRNBL** (**X**, **ILT**, **IRT**, **NINTVL**, **SPROB**, **ALGL** [, ...])
Specific: The specific interface names are **S_TRNBL** and **D_TRNBL**.

FORTRAN 77 Interface

Single: **CALL TRNBL** (**NOBS**, **NCOL**, **X**, **LDX**, **ILT**, **IRT**, **IFRQ**, **ICEN**, **MAXIT**, **EPS**, **IPRINT**, **NINTVL**, **SPROB**, **LDSPRO**, **ALGL**, **NRMISS**)
Double: The double precision name is **DTRNBL**.

Description

Routine **TRNBL** computes nonparametric maximum likelihood estimates of a survival distribution based upon a random sample of data containing exact failure, right-censored, leftcensored (interval censored with a left endpoint of zero), or interval-censored observations. The computational method of Turnbull (1976) is used in computing the probability estimates. The model used is also discussed by Peto (1973).

Routine **TRNBL** begins by finding a set of regions or "failure intervals" (to distinguish them from "observation failure intervals") on the positive real axis in which a change in the survival probability occurs. The survival probability is constant outside of these regions, and undefined within them. Each region (failure interval) is composed of a single left and a single right endpoint obtained from the left and right endpoints of the observation failure intervals (for exact failure times, the left and right endpoints are equal). The regions are defined by the fact that no observation interval endpoints are allowed within a region, except at its endpoints. Note that the endpoints of the intervals need not correspond to a single observation. Regions defined by endpoints from two distinct observations are often obtained.

Let p_i, i = 1, ..., **NINTVL** denote the change in the survival probability within the i-th region, and let the region be denoted by c_i. Let n = **NOBS** and suppose that the observation failure interval for observation j is denoted by I_j. The EM (expectation, maximization) algorithm of Dempster, Laird and Rubin (1977) is used to find the optimal

$$\hat{p}_i's$$

The algorithm is defined as follows:

For given

$$\hat{p}_i$$

compute the expected contribution of the *j*-th observation to the *i*-th change interval as

$$\hat{\mu}_{ij} = \frac{f_j \hat{p}_i \delta_{ij}}{\sum_k f_k \hat{p}_i \delta_{ik}}$$

where $\delta_{ij} = 1$ if $c_i \subseteq I_j$ and $\delta_{ij} = 0$ otherwise, and f_j is the observation frequency.

For given expectations

$$\hat{\mu}_{ij}$$

compute the new probability estimate as

$$\hat{p}_i = \frac{\sum_j \hat{\mu}_{ij}}{\sum_j f_j}$$

Iterate in this manner until convergence. Convergence is assumed when the relative change in the log-likelihood

$$(\ell = \sum_j f_j \ln(\sum_i \delta_{ij} \hat{p}_j))$$

is small (less than **EPS**). Because the algorithm is slow to converge, 5 expectation-maximization cycles are considered to be one iteration of the algorithm. The initial estimate for all the

$$\hat{p}_i's$$

is taken to be one divided by the number of regions (failure intervals).

Comments

1. Workspace may be explicitly provided, if desired, by use of **T2NBL/DT2NBL**. The reference is:

   ```
   CALL T2NBL (NOBS, NCOL, X, LDX, ILT, IRT, IFRQ, ICEN, MAXIT, EPS, IPRINT, NINTVL,
       SPROB, LDSPRO, ALGL, NRMISS, WK, IPERM, INDDR, WWK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 7 * **NOBS**.

   IPERM — Work vector of length **NOBS**.

   INDDR — Work vector of length **NOBS**.

   WWK — Work vector of length 2 * max(**NOBS**, 7).

IWK — Work vector of length max(NOBS, 7).

2. Informational errors

Type	Code	Description
3	3	The maximum number of iterations was exceeded. Convergence is assumed.
4	1	There are no valid observations.
4	2	There are no finite failure intervals present in the data.

Example

The following example contains exact failure, right-, left-, and interval-censored observations. The 20 observations yield 15 change intervals. The last interval is from 192 to ∞, and corresponds to a right-censored observation. When the last interval is infinite, as is the case here, the second column of SPROB contains +∞ in the NINTVL-th position. Left-or right-censored observations input in X are arbitrarily assigned the value 0.0 for the non-specified endpoint.

```
      USE WRRRN_INT
      USE TRNBL_INT

      IMPLICIT   NONE
      INTEGER    ICEN, IFRQ, ILT, IPRINT, IRT, LDSPRO, LDX, NCOL, NOBS
      PARAMETER  (ICEN=4, IFRQ=3, ILT=1, IPRINT=1, IRT=2, LDSPRO=20, &
                  LDX=20, NCOL=4, NOBS=20)
!
      INTEGER    NINTVL, NRMISS
      REAL       ALGL, SPROB(LDSPRO,4), X(LDX,NCOL)
!
      DATA X/0.9, 1.9, 2.5, 3.5, 6.3, 7.1, 18., 25.1, 25.3, 30.3, 45.9, &
          63.5, 70.1, 73.0, 93.0, 94.4, 96.0, 0.0, 191.4, 0.0, 0.9, &
          0.0, 0.0, 0.0, 6.3, 1.9, 1.8, 25.1, 9.5, 30.3, 45.9, &
          60.7, 70.1, 71.0, 74.0, 94.4, 96.0, 96.0, 191.4, 192.0, &
          17*1.0, 5.0, 1.0, 1.0, 0.0, 2.0, 2.0, 2.0, 0.0, 3.0, 3.0, &
          0.0, 3.0, 0.0, 0.0, 3.0, 0.0, 3.0, 3.0, 0.0, 0.0, 1.0, 0.0, &
          1.0/
!
      CALL WRRRN ('X', X)
!
      CALL TRNBL (X, ILT, IRT, NINTVL, SPROB, ALGL, IFRQ=IFRQ, &
                  ICEN=ICEN, IPRINT=IPRINT)
!
      END
```

Output

```
                  X
          1       2       3       4
 1      0.9     0.9     1.0     0.0
 2      1.9     0.0     1.0     2.0
```

```
     3     2.5     0.0     1.0     2.0
     4     3.5     0.0     1.0     2.0
     5     6.3     6.3     1.0     0.0
     6     7.1     1.9     1.0     3.0
     7    18.0     1.8     1.0     3.0
     8    25.1    25.1     1.0     0.0
     9    25.3     9.5     1.0     3.0
    10    30.3    30.3     1.0     0.0
    11    45.9    45.9     1.0     0.0
    12    63.5    60.7     1.0     3.0
    13    70.1    70.1     1.0     0.0
    14    73.0    71.0     1.0     3.0
    15    93.0    74.0     1.0     3.0
    16    94.4    94.4     1.0     0.0
    17    96.0    96.0     1.0     0.0
    18     0.0    96.0     5.0     1.0
    19   191.4   191.4     1.0     0.0
    20     0.0   192.0     1.0     1.0

          Iteration   Log-Likelihood   Relative convergence
                  0     -54.94             ............
                  1     -52.14               0.5367E-01
                  2     -52.09               0.8407E-03
                  3     -52.09               0.1372E-03
                  4     -52.09               0.2476E-04
                  5     -52.08               0.4614E-05

    SPROB
                      Lower        Upper     Interval     Survival
    Interval      Endpoint     Endpoint  Probability  Probability
           1        0.9000       0.9000       0.0972       0.9028
           2        1.9000       1.9000       0.1215       0.7813
           3        6.3000       6.3000       0.0729       0.7083
           4        9.5000      18.0000       0.0000       0.7083
           5       25.1000      25.1000       0.0833       0.6250
           6       30.3000      30.3000       0.0417       0.5833
           7       45.9000      45.9000       0.0417       0.5417
           8       60.7000      63.5000       0.0417       0.5000
           9       70.1000      70.1000       0.0417       0.4583
          10       71.0000      73.0000       0.0417       0.4167
          11       74.0000      93.0000       0.0417       0.3750
          12       94.4000      94.4000       0.0417       0.3333
          13       96.0000      96.0000       0.1111       0.2222
          14      191.4000     191.4000       0.1111       0.1111
          15      192.0000          Inf       0.1111       0.0000
```

PHGLM

Analyzes time event data via the proportional hazards model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)
When ITIE = 1, the observations in X must be grouped by stratum and sorted from largest to smallest failure time within each stratum, with the strata separated.

IRT — Column number in X containing the response variable. (Input)
For point observations, X(*i*, IRT) contains the time of the *i*-th event. For right-censored observations, X(*i*, IRT) contains the right-censoring time. Note that because PHGLM only uses the order of the events, negative "times" are allowed.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input)

INDEF — Index vector of length NVEF(1) + … + NVEF(NEF) containing the column numbers of X associated with each effect. (Input)
The first NVEF(1) elements of INDEF contain the column numbers of X for the variables in the first effect. The next NVEF(2) elements in INDEF contain the column numbers for the second effect, etc.

MAXCL — An upper bound on the sum of the number distinct values taken by the classification variables. (Input)

NCOEF — Number of estimated coefficients in the model. (Output)

COEF — NCOEF by 4 matrix containing the parameter estimates and associated statistics. (Output, if INIT = 0; Input, if INIT = 1 and MAXIT = 0; Input/Output, if INIT = 1 and MAXIT > 0)

Col.	Statistic
1	Coefficient estimate $\hat{\beta}$
2	Estimated standard deviation of the estimated coefficient.
3	Asymptotic normal score for testing that the coefficient is zero against the two-sided alternative.
4	*p*-value associated with the normal score in column 3.

When COEF is input, only column 1 needs to be given.

ALGL — The maximized log-likelihood. (Output)

COV — NCOEF by NCOEF matrix containing the estimated asymptotic variance-covariance matrix of the parameters. (Output)
For MAXIT = 0, COV is the inverse of the Hessian of the negative of the log-likelihood, computed at the estimates input in COEF.

XMEAN — Vector of length NCOEF containing the means of the design variables. (Output)

CASE — NOBS by 5 matrix containing the case statistics for each observation. (Output if MAXIT > 0; used as working storage otherwise)

Col.	Statistic
1	Estimated survival probability at the observation time.
2	Estimated observation influence or leverage.
3	A residual estimate.
4	Estimated cumulative baseline hazard rate.
5	Observation proportionality constant.

GR — Vector of length NCOEF containing the last parameter updates (excluding step halvings). (Output)
For MAXIT = 0, GR contains the inverse of the Hessian times the gradient vector computed at the estimates input in COEF.

IGRP — Vector of length NOBS giving the stratum number used for each observation. (Output)
If RATIO is not -1.0, additional "strata" (other than those specified by column ISTRAT of X) may be generated. IGRP also contains a record of the generated strata. See the "Description" section for more detail.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for each observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

IFIX — Column number in X containing a constant to be added to the linear response. (Input)
Default: IFIX = 0.
The linear response is taken to be $w_i + z_i\hat{\beta}$, where w_i is the observation constant, z_i is the observation design row vector, and $\hat{\beta}$ is the vector of estimated parameters. The "fixed" constant allows one to test hypotheses about parameters via the log-likelihoods. If IFIX = 0, the fixed parameter is assumed to be 0.

ICEN — Column number in X containing the censoring code for each observation. (Input)
Default: ICEN = 0.

If ICEN = 0 a censoring code of 0 is assumed for all observations.

X(*i*, ICEN)	Censoring
0	Point observation at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).

ISTRAT — Column number in X containing the stratification variable. (Input)
If ISTRAT = 0, all observations are considered to be in one stratum. Otherwise, column ISTRAT in X contains a unique number for each stratum. The risk set for an observation is determined by the its stratum.
Default: ISTRAT = 0.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 will usually be sufficient. Use MAXIT = 0 to compute the Hessian and gradient, stored in COV and GR, at the initial estimates. When MAXIT = 0, INIT must be 1.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the relative change in ALGL from one iteration to the next is less than EPS. If EPS is zero, EPS = 0.0001 is assumed.
Default: EPS = 0.0001.

RATIO — Ratio at which a stratum is split into two strata. (Input)
Default: RATIO = 1000.0.
Let

$$r_k = \exp\left(z_k\hat{\beta} + w_k\right)$$

be the observation proportionality constant, where z_k is the design row vector for the k-th observation and w_k is the optional fixed parameter specified by **X**(k, **IFIX**). Let r_{min} be the minimum value r_k in a stratum, where, for failed observations, the minimum is over all times less than or equal to the time of occurrence of the k-th observation. Let r_{max} be the maximum value of r_k for the remaining observations in the group. Then, if $r_{min} >$ **RATIO** r_{max}, the observations in the group are divided into two groups at k. **RATIO** = 1000 is usually a good value. Set **RATIO** = **-**1.0 if no division into strata is to be made.

NCLVAR — Number of classification variables. (Input)
Dummy variables are generated for classification variables using the **IDUMMY** = 2 option of IMSL routine **GRGLM** (see Chapter 2, "Regression"). See also Comment 3.
Default: **NCLVAR** = 0.

INDCL — Index vector of length **NCLVAR** containing the column numbers of **X** that are the classification variables. (Input, if **NCLVAR** is positive, not used otherwise)
If **NCLVAR** is 0, **INDCL** is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: **NEF** = size(**NVEF**,1).

INIT — Initialization option. (Input)
If **INIT** = 1, then the **NCOEF** elements of column 1 of **COEF** contain the initial estimates on input to **PHGLM**. For **INIT** = 0, all initial estimates are taken to be 0.
Default: **INIT** = 0.

ITIE — Option parameter containing the method to be used for handling ties. (Input)
Default: ITIE = 0.

ITIE	Method
0	Breslow's approximate method
1	Failures are assumed to occur in the same order as the observations input in X. The observations in X must be sorted from largest to smallest failure time within each stratum, and grouped by stratum. All observations are treated as if their failure/censoring times were distinct when computing the log-likelihood.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed, but observational statistics are not printed.
2	All output statistics are printed.

NCLVAL — Vector of length NCLVAR containing the number of values taken by each classification variable. (Output, if NCLVAR is positive, not used otherwise)
NCLVAL(*i*) is the number of distinct values for the *i*-th classification variable. If NCLVAR is zero, NCLVAL is not used and can be dimensioned of length 1 in the calling program.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + ... + NCLVAL(NCLVAR) containing the distinct values of the classification variables. (Output, if NCLVAR is positive, not used otherwise)
The first NCLVAL(1) elements of CLVAL contain the values for the first classification variable, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then NCLVAL is not referenced and can be dimensioned of length 1 in the calling program.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of data in **X** that contain missing values in one or more columns **IRT**, **IFRQ**, **IFIX**, **ICEN**, **ISTRAT**, **INDCL**, or **INDEF** of **X**. (Output)

FORTRAN 90 Interface

Generic: **CALL PHGLM** (**X**, **IRT**, **NVEF**, **INDEF**, **MAXCL**, **NCOEF**, **COEF**, **ALGL**, **COV**, **XMEAN**, **CASE**, **GR**, **IGRP** [, ...])

Specific: The specific interface names are **S_PHGLM** and **D_PHGLM**.

FORTRAN 77 Interface

Single: **CALL PHGLM** (**NOBS**, **NCOL**, **X**, **LDX**, **IRT**, **IFRQ**, **IFIX**, **ICEN**, **ISTRAT**, **MAXIT**, **EPS**, **RATIO**, **NCLVAR**, **INDCL**, **NEF**, **NVEF**, **INDEF**, **INIT**, **ITIE**, **IPRINT**, **MAXCL**, **NCLVAL**, **CLVAL**, **NCOEF**, **COEF**, **LDCOEF**, **ALGL**, **COV**, **LDCOV**, **XMEAN**, **CASE**, **LDCASE**, **GR**, **IGRP**, **NRMISS**)

Double: The double precision name is **DPHGLM**.

Description

Routine **PHGLM** computes parameter estimates and other statistics in Proportional Hazards Generalized Linear Models. These models were first proposed by Cox (1972). Two methods for handling ties are allowed in **PHGLM**. Time-dependent covariates are not allowed. The user is referred to Cox and Oakes (1984), Kalbfleisch and Prentice (1980), Elandt-Johnson and Johnson (1980), Lee (1980), or Lawless (1982), among other texts, for a thorough discussion of the Cox proportional hazards model.

Let $\boldsymbol{\lambda}(t, z_i)$ represent the hazard rate at time t for observation number i with covariables contained as elements of row vector z_i. The basic assumption in the proportional hazards model (the proportionality assumption) is that the hazard rate can be written as a product of a time varying function $\boldsymbol{\lambda}_0(t)$, which depends only on time, and a function $\boldsymbol{f}(z_i)$, which depends only on the covariable values. The function $\boldsymbol{f}(z_i)$ used in **PHGLM** is given as $\boldsymbol{f}(z_i) = \exp(w_i + \boldsymbol{\beta} z_i)$ where w_i is a fixed constant assigned to the observation, and $\boldsymbol{\beta}$ is a vector of coefficients to be estimated. With this function one obtains a hazard rate $\boldsymbol{\lambda}(t, z_i) = \boldsymbol{\lambda}_0(t) \exp(w_i + \boldsymbol{\beta} z_i)$. The form of $\boldsymbol{\lambda}_0(t)$ is not important in proportional hazards models.

The constants w_i may be known theoretically. For example, the hazard rate may be proportional to a known length or area, and the w_i can then be determined from this known length or area. Alternatively, the w_i may be used to fix a subset of the coefficients $\boldsymbol{\beta}$ (say, $\boldsymbol{\beta}_1$) at specified values. When w_i is used in this way, constants $w_i = \boldsymbol{\beta}_1 z_{1i}$ are used, while the remaining coefficients in $\boldsymbol{\beta}$ are free to vary in the optimization algorithm. If user-specified constants are not desired, the user should set **IFIX** to 0 so that $w_i = 0$ will be used.

With this definition of $\lambda(t, z_i)$, the usual partial (or marginal, see Kalbfleisch and Prentice (1980)) likelihood becomes

$$L = \prod_{i=1}^{n_d} \frac{\exp(w_i + \beta z_i)}{\sum_{j \in R(t_i)} \exp(w_j + \beta z_j)}$$

where $R(t_i)$ denotes the set of indices of observations that have not yet failed at time t_i (the risk set), t_i denotes the time of failure for the i-th observation, n_d is the total number of observations that fail. Right-censored observations (i.e., observations that are known to have survived to time t_i, but for which no time of failure is known) are incorporated into the likelihood through the risk set $R(t_i)$. Such observations never appear in the numerator of the likelihood. When **ITIE** = 0, all observations that are censored at time t_i are not included in $R(t_i)$, while all observations that fail at time t_i are included in $R(t_i)$.

If it can be assumed that the dependence of the hazard rate upon the covariate values remains the same from stratum to stratum, while the time-dependent term, $\lambda_0(t)$, may be different in different strata, then **PHGLM** allows the incorporation of strata into the likelihood as follows. Let k index the m = **NSTRAT** strata. Then, the likelihood is given by

$$L_s = \prod_{k=1}^{m} \left[\prod_{i=1}^{n_k} \frac{\exp(w_{ki} + \beta z_{ki})}{\sum_{j \in R(t_{ki})} \exp(w_{kj} + \beta z_{kj})} \right]$$

In **PHGLM**, the log of the likelihood is maximized with respect to the coefficients β. A quasi-Newton algorithm approximating the Hessian via the matrix of sums of squares and cross products of the first partial derivatives is used in the initial iterations (the "Q-N" method in the output). When the change in the log-likelihood from one iteration to the next is less than 100***EPS**, Newton-Raphson iteration is used (the "N-R" method). If, during any iteration, the initial step does not lead to an increase in the log-likelihood, then step halving is employed to find a step that will increase the log-likelihood.

Once the maximum likelihood estimates have been computed, **PHGLM** computes estimates of a probability associated with each failure. Within stratum k, an estimate of the probability that the i-th observation fails at time t_i given the risk set $R(t_{ki})$ is given by

$$p_{ki} = \frac{\exp\left(w_{ki} + z_{ki}\beta\right)}{\sum_{j \in R\left(t_{ki}\right)} \exp\left(w_{kj} + z_{kj}\beta\right)}$$

A diagnostic "influence" or "leverage" statistic is computed for each noncensored observation as:

$$l_{ki} = -g'_{ki} H_s^{-1} g'_{ki}$$

where H_S is the matrix of second partial derivatives of the log-likelihood, and

$$g'_{ki}$$

is computed as:

$$g'_{ki} = z_{ki} - \frac{z_{ki}\exp\left(w_{ki} + z_{ki}\beta\right)}{\sum_{j \in R\left(t_{ki}\right)} \exp\left(w_{kj} + z_{kj}\beta\right)}$$

Influence statistics are not computed for censored observations.

A "residual" is computed for each of the input observations according to methods given in Cox and Oakes (1984, page 108). Residuals are computed as

$$r_{ki} = \exp(w_{ki} + z_{ki}\hat{\beta}) \sum_{j \in R(t_{ki})} \frac{d_{kj}}{\sum_{l \in R(t_{kj})} \exp(w_{kl} + z_{kl}\hat{\beta})}$$

where d_{kj} is the number of tied failures in group k at time t_{kj}. Assuming that the proportional hazards assumption holds, the residuals should approximate a random sample (with censoring) from the unit exponential distribution. By subtracting the expected values, centered residuals can be obtained. (The j-th expected order statistic from the unit exponential with censoring is given as

$$e_j = \sum_{l \le j} \frac{1}{h - l + 1}$$

where h is the sample size, and censored observations are not included in the summation.)

An estimate of the cumulative baseline hazard within group k is given as

$$\hat{H}_{k0}(t_{ik}) = \sum_{t_{kj} \le t_{ki}} \frac{d_{kj}}{\sum_{l \in R(t_{kj})} \exp(w_{kl} + z_{kl}\hat{\beta})}$$

The observation proportionality constant is computed as

$$\exp\left(w_{ki} + z_{ki}\hat{\beta}\right)$$

Comments

1. Workspace may be explicitly provided, if desired, by use of P2GLM/DP2GLM. The reference is:

   CALL P2GLM (NOBS, NCOL, X, LDX, IRT, IFRQ, IFIX, ICEN, ISTRAT, MAXIT, EPS, RATIO, NCLVAR, INDCL, NEF, NVEF, INDEF, INIT, ITIE, IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF, COEF, LDCOEF, ALGL, COV, LDCOV, XMEAN, CASE, LDCASE, GR, IGRP, NRMISS, OBS, SMG, SMH, IPTR, IDT, IWK)

   The additional arguments are as follows:

   OBS — Work vector of length NCOEF + 1.

   SMG — Work vector of length NCOEF.

   SMH — Work vector of length max(NCOEF * NCOEF, 2).

   IPTR — Work vector of length NOBS + NCOEF.

   IDT — Work vector of length NOBS.

   IWK — Work vector of length 3 * max(NOBS, NCOL)

2. Informational errors

Type	Code	Description
3	1	Too many iterations required. Convergence assumed.
3	2	Too many step halvings. Convergence assumed.
3	3	Additional strata were formed as required because of the detection of infinite parameter estimates.
4	4	The number of distinct values of the classification variables exceeds MAXCL.
4	5	The model specified by NEF, NVEF, and INDEF yields no covariates.
4	6	After eliminating observations with missing values, no valid observations remain.
4	7	After eliminating observations with missing values, only one covariate vector remains.
4	8	The number of distinct values for each classification variable must be greater than one.
4	9	LDCOEF or LDCOV must be greater or equal to NCOEF.

3. Dummy variables are generated for the classification variables as follows: An ascending list of all distinct values of the classification variable is obtained and stored in CLVAL. Dummy variables are then generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case, the dummy variable is one. See argument IDUMMY for IDUMMY = 2 in routine GRGLM in Chapter 2, "Regression".

4. The "product" of a classification variable with a covariate yields dummy variables equal to the product of the covariate with each of the dummy variables associated with the classification variable.

5. The "product" of two classification variables yields dummy variables in the usual manner. Each dummy variable associated with the first classification variable multiplies each dummy variable associated with the second classification variable. The resulting dummy variables are such that the index of the second classification variable varies fastest.

Programming Notes

1. The covariate vectors z_{ki} are computed from each row of the input matrix **X** via routine **GRGLM** in Chapter 2, "Regression".). Thus, class variables are easily incorporated into the z_{ki}. The reader is referred to the document for **GRGLM** in the regression chapter for a more detailed discussion. Note that **PHGLM** calls **GRGLM** with the option **IDUMMY** = 2.

2. The average of each of the explanatory variables is subtracted from the variable prior to computing the product $z_{ki}\boldsymbol{\beta}$. Subtraction of the mean values has no effect on the computed log-likelihood or the estimates since the constant term occurs in both the numerator and denominator of the likelihood. Subtracting the mean values does help to avoid invalid exponentiation in the algorithm and may also speed convergence.

3. Routine **PHGLM** allows for two methods of handling ties. In the first method (**ITIE** = 1), the user is allowed to break ties in any manner desired. When this method is used, it is assumed that the user has sorted the rows in **X** from largest to smallest with respect to the failure/censoring times **X**(*i*, **IRT**) within each stratum (and across strata), with tied observations (failures or censored) broken in the manner desired. The same effect can be obtained with **ITIE** = 0 by adding (or subtracting) a small amount from each of the tied observations failure/censoring times t_i = **X**(*i*, **IRT**) so as to break the ties in the desired manner.

   The second method for handling ties (**ITIE** = 0) uses an approximation for the tied likelihood proposed by Breslow (1974). The likelihood in Breslow's method is as specified above, with the risk set at time t_i, including all observations that fail at time t_i, while all observations that are censored at time t_i are not included. (Tied censored observations are assumed to be censored immediately prior to the time t_i).

4. If **INIT** = 1, then it is assumed that the user has provided initial estimates for the model coefficients $\boldsymbol{\beta}$ in the first column of the matrix **COEF**. When initial estimates are provided by the user, care should be taken to ensure that the estimates correspond to the generated covariate vector z_{ki}. If **INIT** = 0, then initial estimates of zero are used for all of the coefficients. This corresponds to no effect from any of the covariate values.

5. If a linear combination of covariates is monotonically increasing or decreasing with increasing failure times, then one or more of the estimated coefficients is infinite and extended maximum likelihood estimates must be computed. Such estimates may be written as

$$\hat{\beta} = \hat{\beta}_f + \rho\hat{\gamma}$$

where $\rho = \infty$ at the supremum of the likelihood so that

$$\hat{\beta}_f$$

is the finite part of the solution. In **PHGLM**, it is assumed that extended maximum likelihood estimates must be computed if, within any group k, for any time t,

$$\min_{t_{ki}<t} \exp\left(w_{ki} + z_{ki}\hat{\beta}\right) > \rho \max_{t_{ki}<t} \exp\left(w_{ki} + z_{ki}\hat{\beta}\right)$$

where ρ = **RATIO** is specified by the user. Thus, for example, if ρ = 10000, then **PHGLM** does not compute extended maximum likelihood estimates until the estimated proportionality constant

$$\exp\left(w_{ki} + z_{ki}\hat{\beta}\right)$$

is 10000 times larger for all observations prior to *t* than for all observations after *t*. When this occurs, **PHGLM** computes estimates for

$$\hat{\beta}_f$$

by splitting the failures in stratum *k* into two strata at *t* (see Bryson and Johnson 1981). Censored observations in stratum *k* are placed into a stratum based upon the associated value for

$$\exp\left(w_{ki} + z_{ki}\hat{\beta}\right)$$

The results of the splitting are returned in **IGRP**.

The estimates

$$\hat{\beta}_f$$

based upon the stratified likelihood represent the finite part of the extended maximum likelihood solution. Routine **PHGLM** does not compute $\hat{\gamma}$ explicitly, but an estimate for $\hat{\gamma}$ may be obtained in some circumstances by setting **RATIO** = **-**1 and optimizing the log-likelihood without forming additional strata. The solution

$$\hat{\beta}$$

obtained will be such that

$$\hat{\beta} = \hat{\beta}_f + \rho\hat{\gamma}$$

for some finite value of **ρ** > 0. At this solution, the Newton-Raphson algorithm will not have "converged" because the Newton-Raphson step sizes returned in **GR** will be large, at least for some variables. Convergence will be declared, however, because the relative change in the log-likelihood during the final iterations will be small.

Examples

Example 1

The following data are taken from Lawless (1982, page 287) and involve the survival of lung cancer patients based upon their initial tumor types and treatment type. In the first example, the likelihood is maximized with no strata present in the data. This corresponds to Example 7.2.3 in Lawless (1982, page 367). The input data is printed in the output. The model is given as:

$$\ln(\lambda) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \alpha_i + \gamma_j$$

where α_i and γ_j correspond to dummy variables generated from columns 6 and 7 of **X**, respectively, x_1 corresponds to column 3 of **X**, x_2 corresponds to column 4 of **X**, and x_3 corresponds to column 5 of **X**.

```
      USE PHGLM_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    ICEN, IPRINT, IRT, LDCASE, LDCOEF, LDCOV, &
                 LDX, MAXCL, NCLVAR, NCOL, NEF, NOBS

      REAL       RATIO, LABEL
      PARAMETER  (ICEN=2, IPRINT=2, IRT=1, LDCOEF=7, LDX=40, &
                 MAXCL=10, NCLVAR=2, NCOL=7, NEF=5, RATIO=10000.0, &
                 LDCASE=LDX, LDCOV=LDCOEF, NOBS=LDX)

!
      INTEGER    IGRP(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                 NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(6), COEF(LDCOEF,4), &
                 COV(LDCOV,LDCOV), GR(LDCOV), X(LDX,NCOL), XMEAN(LDCOV)
      CHARACTER  NUMBER(1)*6
      DATA       NUMBER(1)/'NUMBER'/

!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
```

```
            10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
            44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
            5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
            3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
            4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
            48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
            54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
            39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
            25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
            22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
            5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      LABEL = 'NUMBER'
      CALL WRRRL ('The First 10 Rows of the Input Data', &
                  X, NUMBER, NUMBER, 10, NCOL, LDX, FMT='(I7)')
!
      CALL PHGLM (X, IRT, NVEF, INDEF, MAXCL, NCOEF, COEF, ALGL, &
                  COV, XMEAN, CASE, GR, IGRP, ICEN=ICEN, RATIO=RATIO,&
                  NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, IPRINT=IPRINT, &
                  NClVAL=NClVAL, CLVAL=CLVAL, NRMISS=NRMISS)
!
      END
```

Output

```
                The First 10 Rows of the Input Data
          1       2       3       4       5       6       7
 1      411       0       7      64       5       1       0
 2      126       0       6      63       9       1       0
 3      118       0       7      65      11       1       0
 4       92       0       4      69      10       1       0
 5        8       0       4      63      58       1       0
 6       25       1       7      48       9       1       0
 7       11       0       7      48      11       1       0
 8       54       0       8      63       4       2       0
 9      153       0       6      63      14       2       0
10       16       0       3      53       4       2       0

                    Initial Estimates
     1       2       3       4       5       6       7
0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

 Method  Iteration  Step size  Maximum scaled     Log
                               coef. update      likelihood
   Q-N        0                                 -102.4
   Q-N        1      1.0000      0.5034           -91.04
   Q-N        2      1.0000      0.5782           -88.07
   N-R        3      1.0000      0.1131           -87.92
   N-R        4      1.0000      0.6958E-01       -87.89
   N-R        5      1.0000      0.8144E-03       -87.89

 Log-likelihood       -87.88779

                Coefficient Statistics
                    Standard    Asymptotic    Asymptotic
    Coefficient        error   z-statistic       p-value
1        -0.585        0.137        -4.272         0.000
2        -0.013        0.021        -0.634         0.526
```

3	0.001	0.012	0.064	0.949	
4	-0.367	0.485	-0.757	0.449	
5	-0.008	0.507	-0.015	0.988	
6	1.113	0.633	1.758	0.079	
7	0.380	0.406	0.936	0.349	

Asymptotic Coefficient Covariance

	1	2	3	4	5
1	0.1873E-01	0.2530E-03	0.3345E-03	0.5745E-02	0.9750E-02
2		0.4235E-03	-0.4120E-04	-0.1663E-02	-0.7954E-03
3			0.1397E-03	0.8111E-03	-0.1831E-02
4				0.2350	0.9799E-01
5					0.2568

	6	7
1	0.4264E-02	0.2082E-02
2	-0.3079E-02	-0.2898E-02
3	0.5995E-03	0.1684E-02
4	0.1184	0.3735E-01
5	0.1253	-0.1944E-01
6	0.4008	0.6289E-01
7		0.1647

Case Analysis

	Survival Probability	Influence	Residual	Cumulative hazard	Proportionality constant
1	0.00	0.04	2.05	6.10	0.3
2	0.30	0.11	0.74	1.21	0.61
3	0.34	0.12	0.36	1.07	0.33
4	0.43	0.16	1.53	0.84	1.83
5	0.96	0.56	0.09	0.05	2.05
6	0.74	NaN	0.13	0.31	0.42
7	0.92	0.37	0.03	0.08	0.42
8	0.59	0.26	0.14	0.53	0.27
9	0.26	0.12	1.20	1.36	0.88
10	0.85	0.15	0.97	0.17	5.76
11	0.55	0.31	0.21	0.60	0.36
12	0.74	0.21	0.96	0.31	3.12
13	0.03	0.06	3.02	3.53	0.86
14	0.94	0.09	0.17	0.06	2.71
15	0.96	0.16	1.31	0.05	28.89
16	0.89	0.23	0.59	0.12	4.82
17	0.18	0.09	2.62	1.71	1.54
18	0.89	0.19	0.33	0.12	2.68
19	0.14	0.23	0.72	1.96	0.37
20	0.05	0.09	1.66	2.95	0.56
21	0.39	0.22	1.17	0.94	1.25
22	0.00	0.00	1.73	21.11	0.08
23	0.08	NaN	2.19	2.52	0.87
24	0.00	0.00	2.46	8.89	0.28
25	0.99	0.31	0.05	0.01	4.28
26	0.11	0.17	0.34	2.23	0.15
27	0.66	0.25	0.16	0.41	0.38
28	0.87	0.22	0.15	0.14	1.02
29	0.39	NaN	0.45	0.94	0.48
30	0.98	0.25	0.06	0.02	2.53
31	0.77	0.26	1.03	0.26	3.90
32	0.63	0.35	1.80	0.46	3.88
33	0.82	0.26	1.06	0.19	5.47
34	0.47	0.26	1.65	0.75	2.21

```
    35          0.51          0.32          0.39          0.67          0.58
    36          0.22          0.18          0.49          1.53          0.32
    37          0.80          0.26          1.08          0.23          4.77
    38          0.70          0.16          0.26          0.36          0.73
    39          0.01          0.23          0.87          4.66          0.19
    40          0.08          0.20          0.81          2.52          0.32

                          Last Coefficient Update
            1           2           3           4           5           6
-1.016E-07   1.918E-09  -1.305E-08  -7.190E-07  -2.854E-07   2.108E-08


            7
-6.947E-08

                     Covariate Means
      1       2       3       4       5       6       7
5.65   56.58   15.65    0.35    0.28    0.12    0.53

Distinct Values For Each Class Variable
Variable  1:     1.0         2.0         3.0         4.0
Variable  2:      0.         1.0

                        Stratum Numbers For Each Observation
1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
1  1  1  1  1  1  1  1  1   1  1  1  1  1  1  1  1  1  1  1

21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

Number of Missing Values           0
```

Example 2

This example illustrates the use of **PHGLM** when there are strata present in the data. The observations from Example 1 are arbitrarily grouped into four strata (the first ten observations form stratum 1, the next 10 for stratum 2, etc.). Otherwise, the problem is unchanged. The resulting coefficients are very similar to those obtained when there is no stratification variable. The model is the same as in Example 1.

```
      USE PHGLM_INT

      IMPLICIT   NONE
      INTEGER    LDCASE, LDCOEF, LDCOV, LDX, MAXCL, NCLVAR, NCOL, NEF, &
                 NOBS
      REAL       RATIO
      PARAMETER  (LDCOEF=7, LDX=40, MAXCL=10, NCLVAR=2, NCOL=8, NEF=5, &
                LDCASE=LDX, LDCOV=LDCOEF, NOBS=LDX, RATIO=10000.0)
!                                SPECIFICATIONS FOR PARAMETERS
      INTEGER    ICEn, IPRINT, IRT, ISTRAT
      PARAMETER  (ICEN=2, IPRINT=2, IRT=1, ISTRAT=8)
!                                SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER    IGRP(NOBS), NCLVAL(NCLVAR), NCOEF, NRMISS
      REAL       ALGL, CASE(LDCASE,5), CLVAL(6), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), XMEAN(LDCOV)
!                                SPECIFICATIONS FOR SAVE VARIABLES
      INTEGER    INDCL(NCLVAR), INDEF(NEF), NVEF(NEF)
      REAL       X(LDX,NCOL)
```

```
      SAVE       INDCL, INDEF, NVEF, X
!                                 SPECIFICATIONS FOR SUBROUTINES
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1, 10*1, 10*2, 10*3, 10*4/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
     CALL PHGLM (X, IRT, NVEF, INDEF, MAXCL, NCOEF,COEF, ALGL, COV, XMEAN,&
                 CASE, GR, IGRP, ICEN=ICEN, ISTRAT=ISTRAT, RATIO=RATIO, &
                 NCLVAR=NCLVAR, INDCL=INDCL, IPRINT=IPRINT, NCLVAL=NCLVAL, &
                 CLVAL=CLVAL, NRMISS=NRMISS)
!
      END
```

Output

```
                    Initial Estimates
      1        2        3        4        5        6        7
 0.0000   0.0000   0.0000   0.0000   0.0000   0.0000   0.0000

 Method  Iteration  Step size  Maximum scaled     Log
                                coef. update      likelihood
    Q-N         0                                -55.90
    Q-N         1      1.0000      0.6748        -45.79
    Q-N         2      1.0000      0.7105        -42.85
    N-R         3      1.0000      0.2315        -42.59
    N-R         4      1.0000      0.1674        -42.55
    N-R         5      1.0000      0.3372E-02    -42.55

  Log-likelihood       -42.54570

                 Coefficient Statistics
                     Standard    Asymptotic    Asymptotic
     Coefficient        error   z-statistic       p-value
1        -0.716        0.170        -4.222         0.000
2        -0.033        0.030        -1.122         0.262
3         0.001        0.015         0.048         0.961
4        -0.100        0.999        -0.100         0.921
5        -0.405        0.729        -0.555         0.579
6         1.136        0.769         1.478         0.139
7        -0.087        1.454        -0.060         0.952

 Asymptotic Coefficient Covariance
              1            2            3            4            5
1    0.2877E-01   0.8662E-03   0.3119E-03   0.5057E-02   0.2480E-01
2                 0.8842E-03  -0.8137E-04  -0.7623E-02  -0.6925E-03
3                              0.2158E-03  -0.2567E-02  -0.3738E-02
4                                           0.9975       0.5109
5                                                        0.5319
```

```
                6             7
1    -0.7669E-02    0.6405E-02
2    -0.8800E-03    0.4120E-02
3     0.1170E-02   -0.3699E-02
4     0.1944        0.8056
5     0.1802        0.4905
6     0.5909        0.1858
7                   2.114

                              Case Analysis
         Survival                                  Cumulative  Proportionality
      Probability     Influence      Residual         hazard         constant
 1           0.00          0.00          2.01           7.83             0.26
 2           0.09          0.06          1.32           2.42             0.55
 3           0.20          0.04          0.40           1.59             0.25
 4           0.40          0.04          1.69           0.91             1.87
 5           0.92          0.47          0.21           0.09             2.36
 6           0.73           NaN          0.14           0.31             0.44
 7           0.82          0.47          0.09           0.20             0.44
 8           0.55          0.67          0.06           0.61             0.10
 9           0.02          0.07          1.59           3.94             0.40
10           0.73          0.10          1.50           0.31             4.79
11           0.39          0.68          0.17           0.93             0.19
12           0.60          0.14          1.12           0.51             2.19
13           0.00          0.00          2.32           6.32             0.37
14           0.90          0.16          0.15           0.10             1.49
15           0.98          0.04          0.75           0.02            35.42
16           0.75          0.21          1.12           0.29             3.83
17           0.25          0.07          1.55           1.39             1.12
18           0.75          0.21          0.63           0.29             2.14
19           0.10          0.18          0.69           2.31             0.30
20           0.03          0.11          1.48           3.60             0.41
21           0.50          0.61          1.00           0.70             1.44
22           0.00          0.00          1.28          13.59             0.09
23           0.33           NaN          1.92           1.09             1.76
24           0.05          0.00          1.32           2.94             0.45
25           0.95          0.15          0.47           0.05             9.84
26           0.33          0.24          0.23           1.09             0.21
27           0.62          0.40          0.22           0.47             0.47
28           0.76          0.13          0.71           0.27             2.63
29           0.50           NaN          0.37           0.70             0.53
30           0.87          0.23          0.49           0.14             3.53
31           0.88          0.35          0.67           0.13             5.07
32           0.71          0.22          1.56           0.34             4.54
33           0.97          0.52          0.20           0.03             7.00
34           0.44          0.03          2.64           0.83             3.19
35           0.56          0.20          0.29           0.57             0.50
36           0.11          0.00          0.61           2.24             0.27
37           0.94          0.19          0.82           0.07            12.50
38           0.79          0.43          0.24           0.23             1.05
39           0.00          0.00          1.69          11.13             0.15
40           0.01          0.00          1.28           4.54             0.28
                          Last Coefficient Update
          1           2           3           4           5           6
 -7.363E-07   8.762E-09   1.252E-08  -1.697E-06  -1.642E-06   1.075E-06

          7
 -1.772E-06
```

```
                    Covariate Means
       1       2       3       4       5       6       7
    5.65   56.58   15.65    0.35    0.28    0.12    0.53

    Distinct Values For Each Class Variable
    Variable  1:     1.0         2.0         3.0         4.0
    Variable  2:      0.         1.0

                      Stratum Numbers For Each Observation
    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
    1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2

   21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
    3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4   4

   Number of Missing Values           0
```

SVGLM

Analyzes censored survival data using a generalized linear model.

Required Arguments

X — NOBS by NCOL matrix containing the data. (Input)

MODEL — Model option parameter. (Input)
MODEL specifies the distribution of the response variable and the relationship of the linear model to a distribution parameter.

MODEL	Distribution
0	Exponential
1	Linear hazard
2	Log-normal
3	Normal
4	Log-logistic
5	Logistic
6	Log least extreme value
7	Least extreme value
8	Log extreme value
9	Extreme value
10	Weibull

For further discussion of the models and parameterizations used, see the "Description" section.

ILT — For interval-censored and left-censored observations, the column number in X that contains the upper endpoint of the failure interval. (Input)
See argument ICEN. If ILT = 0, left-censored and interval-censored observations cannot be input.

IRT — For interval-censored and right-censored observations, the column number in X that contains the lower endpoint of the failure interval. (Input)
For exact-failure observations, X(*i*, IRT) contains the exact-failure time. IRT must not be zero. See argument ICEN.

MAXCL — An upper bound on the sum of the number of distinct values taken by the classification variables. (Input)

NCOEF — Number of estimated coefficients in the model. (Output, if `INIT` = 0; Input, if `INIT` = 1)

COEF — `NCOEF` by 4 matrix containing parameter estimates and associated statistics. (Output, if `INIT` = 0; Input/Output, if `INIT` = 1; Input, if `MAXIT` = 0)

	Statistic
1	Coefficient estimate.
2	Estimated standard deviation of the estimated coefficient.
3	Asymptotic normal score for testing that the coefficient is zero.
4	*p*-value associated with the normal score in column 3.

When `COEF` is input, only column 1 is referenced as input data, and columns 2 to 4 need not be set. When present in the model, the initial coefficient in `COEF` estimates a "nuisance" parameter, and the remaining coefficients estimate parameters associated with the "linear" model, beginning with the intercept, if present. Nuisance parameters are as follows:

	Nuisance Parameter
1	Coefficient of the quadratic term in time, θ
2 – 9	Scale parameter, σ
10	Shape parameter, θ

ALGL — Maximized log-likelihood. (Output)

COV — `NCOEF` by `NCOEF` matrix containing the estimated asymptotic covariance matrix of the coefficients. (Output)
`COV` is computed as the inverse of the matrix of second partial derivatives of negative one times the log-likelihood. When `MAXIT` = 0, `COV` is computed at the initial estimates.

XMEAN — Vector of length `NCOEF` containing the means of the design variables. (Output)

CASE — `NOBS` by 5 vector containing the case analysis. (Output)

	Statistics
1	Estimated predicted value
2	Estimated influence or leverage
3	Residual estimate
4	Estimated cumulative hazard
5	For non-censored observations, the estimated density at the observation failure time and covariate values. For censored observations, the corresponding estimated probability.

If MAXIT = 0, CASE is a NOBS by 1 vector containing the estimated probability (for censored observations) or the estimated density (for non censored observations).

GR — Vector of length NCOEF containing the last parameter updates, excluding step halvings. (Output) GR is computed as the inverse of the matrix of second partial derivatives times the vector of first partial derivatives of the log-likelihood. When MAXIT = 0, the derivatives are computed at the initial estimates.

IADDS — Vector of length NOBS indicating which observations have and have not been included in the model. (Output, if MAXIT > 0; Input/Output, if MAXIT = 0)

	Status of Observation
0	Observation *i* has been included in the model.
1	Observation *i* has not been included in the model due to missing values in the X matrix.
2	Observation *i* has not been included in the model because of infinite estimates in extended maximum likelihood estimation. If MAXIT = 0, then the IADDS array must be initialized prior to calling SVGLM.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NCOL — Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

IFRQ — Column number in X containing the frequency of response for each observation. (Input)
If IFRQ = 0, a response frequency of 1 for each observation is assumed.
Default: IFRQ = 0.

IFIX — Column number in X containing a constant to be added to the linear response. (Input)
Default: IFIX = 0.
The estimated linear response is taken to be $w_i + z_i\hat{\beta}$ where w_i is the observation constant, z_i is the observation design vector, $\hat{\beta}$ is the vector of estimated parameters output in the first column of COEF, and *i* indexes the observations. The "fixed" constant allows one to test hypotheses about parameters via the log-likelihoods. If IFIX = 0, the fixed parameter is assumed to be 0.

ICEN — Column number in X containing the censoring code for each observation. (Input)
Default: ICEN = 0.
If ICEN = 0, a censoring code of 0 is assumed. Valid censoring codes are:

X(ICEN)	Censoring
0	Exact failure at X(*i*, IRT).
1	Right censored. The response is greater than X(*i*, IRT).
2	Left censored. The response is less than or equal to X(*i*, ILT).
3	Interval censored. The response is greater than X(*i*, IRT), but less than or equal to X(*i*, ILT).

INFIN — Method to be used for handling infinite estimates. (Input)
Default: INFIN = 0.

INFIN	Method
0	Remove a right or left-censored observation from the loglikelihood whenever the probability of the observation exceeds 0.995. At convergence, use linear programming to check that all removed observations actually have infinite linear response $z_i\hat{\beta}.$ Set IADDS(i) for observation i to 2 if the linear response is infinite. If not all removed observations have infinite linear response $z_i\hat{\beta}.$
1	Iterate without checking for infinite estimates.

See the "Description" section for more discussion.

MAXIT — Maximum number of iterations. (Input)
MAXIT = 30 will usually be sufficient. Use MAXIT = 0 to compute the Hessian and score vector at the initial estimates.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
Convergence is assumed when the maximum relative change in any coefficient estimate is less than EPS from one iteration to the next, or when the relative change in the log-likelihood, ALGL, from one iteration to the next is less than EPS/100. If EPS is negative, EPS = 0.001 is assumed.
Default: EPS = 0.001.

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	No intercept is in the model (unless otherwise provided for by the user).
1	An intercept is automatically included in the model..

NCLVAR — Number of classification variables. (Input)
Dummy or indicator variables are generated for classification variables using the IDUMMY = 2 option of routine GRGLM (see Chapter 2, "Regression"). See Comment 3.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are classification variables. (Input, if NCLVAR is positive, not used otherwise)
If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: NEF = 0.

NVEF — Vector of length NEF containing the number of variables associated with each effect in the model. (Input, if NEF is positive; not used otherwise)
If NEF is zero, NVEF is not used and can be dimensioned of length 1 in the calling program.

INDEF — Index vector of length NVEF(1) + NVEF(2) + ... + NVEF(NEF) containing the column numbers in X associated with each effect. (Input, if NEF is positive; not used otherwise)
The first NVEF(1) elements of INDEF give the column numbers in X of the variables in the first effect. The next NVEF(2) elements of INDEF give the column numbers for the second effect, etc. If NEF is zero, INDEF is not used and can be dimensioned of length one in the calling program.

INIT — Initialization option. (Input)
Default: INIT = 0.

INIT	Action
0	Unweighted linear regression is used to obtain initial estimates.
1	The NCOEF elements in the first column of COEF contain initial estimates of the parameters on input to SVGLM (requiring that the user know NCOEF prior to calling SVGLM).

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed..
1	Printing is performed, but observational statistics are not printed..
2	All output statistics are printed.

NCLVAL — Vector of length NCLVAR containing the number of values taken by each classification variable. (Output, if NCLVAR is positive; not used otherwise)
NCLVAL(*i*) is the number of distinct values for the *i*-th classification variable. If NCLVAR is zero, NCLVAL is not used and can be dimensioned of length 1 in the calling program.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + … + NCLVAL(NCLVAR) containing the distinct values of the classification variables in ascending order. (Output, if NCLVAR is positive; not used otherwise)
The first NCLVAL(1) elements contain the values for the first classification variables, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then CLVAL is not referenced and can be dimensioned of length 1 in the calling program.

LDCOEF — Leading dimension of COEF exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOEF = size (COEF,1).

LDCOV — Leading dimension of COV exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOV = size (COV,1).

LDCASE — Leading dimension of CASE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCASE = size (CASE,1).

NRMISS — Number of rows of data in X that contain missing values in one or more columns ILT, IRT, IFRQ, ICOEF, ICEN, INDCL or INDEF of X. (Output)

FORTRAN 90 Interface

Generic: CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, XMEAN, CASE, GR, IADDS [, ...])

Specific: The specific interface names are S_SVGLM and D_SVGLM.

FORTRAN 77 Interface

Single: CALL SVGLM (NOBS, NCOL, X, LDX, MODEL, ILT, IRT, IFRQ, IFIX, ICEN, INFIN, MAXIT, EPS, INTCEP, NCLVAR, INDCL, NEF, NVEF, INDEF, INIT, IPRINT, MAXCL, NCLVAL, CLVAL, NCOEF, COEF, LDCOEF, ALGL, COV, LDCOV, XMEAN, CASE, LDCASE, GR, IADDS, NRMISS)

Double: The double precision name is DSVGLM.

Description

Routine SVGLM computes maximum likelihood estimates of parameters and associated statistics in generalized linear models commonly found in survival (reliability) analysis. Although the terminology used will be from the survival area, the methods discussed have application in many areas of data analysis, including reliability analysis and event history analysis. Indeed, these methods may be used anywhere a random variable from one of the discussed distributions is parameterized via one of the models available in SVGLM. Thus, while it is not advisable to do so, standard multiple linear regression may be performed by routine SVGLM. Estimates for any of ten standard models can be computed. Exact, left-censored, right-censored, or interval-censored observations are allowed. (Note that left censoring is the same as interval censoring with left endpoint equal to the left endpoint of the support of the distribution.)

Let $\eta = x^T\beta$ be the linear parameterization, where x is a design vector obtained in SVGLM via routine GRGLM (see Chapter 2, "Regression") from a row of X, and β is a vector of parameters associated with the linear model. Let T denote the random response variable and $S(t)$ denote the probability that $T > t$. All models considered also allow a fixed parameter w_i for observation i (input in column IFIX of X). Use of this parameter is discussed below. There may also be nuisance parameters $\theta > 0$, or $\sigma > 0$ to be estimated (along with β) in the various models. Let Φ denote the cumulative normal distribution. The survival models available in SVGLM are:

	Name	*S(t)*
0	Exponential	$\exp\left\{-\exp\left(w_i+\eta\right)\right\}$
1	Linear hazard	$\exp\{-(t+\frac{\theta t^2}{2})\exp[(w_i+\eta)]\}$
2	Log-normal	$1-\Phi(\frac{\ln(t)-\eta-w_i}{\sigma})$
3	Normal	$1-\Phi(\frac{t-\eta-w_i}{\sigma})$
4	Log-logistic	$\{1+\exp(\frac{\ln(t)-\eta-w_i}{\sigma})\}^{-1}$
5	Logistic	$\{1+\exp(\frac{t-\eta-w_i}{\sigma})\}^{-1}$
6	Log least extreme value	$\exp\{-\exp(\frac{\ln(t)-\eta-w_i}{\sigma})\}$
7	Least extreme value	$\exp\{-\exp(\frac{t-\eta-w_i}{\sigma})\}$
8	Log extreme value	$1-\exp\{-\exp[-\frac{\ln(t)-\eta-w_i}{\sigma})]\}$
9	Extreme value	$1-\exp\{-\exp[-(\frac{t-\eta-w_i}{\sigma})]\}$
10	Weibull	$\exp\{-\left(\frac{t}{\exp(w_i+\eta)}\right)^{\theta}\}$

Note that the log-least-extreme-value model is a reparameterization of the Weibull model. Moreover, models 0, 1, 2, 4, 6, 8, and 10 require that $T > 0$, while all of the remaining models allow any value for T, $-\infty < T < \infty$.

Each row in the data matrix can represent a single observation, or, through the use of column **IFRQ**, it can represent several observations. Classification variables and their products are easily incorporated into the models via the usual GLM type specifications through the use of variables **NCLVAR** and **INDCL**, and the model variables **NEF**, **NVEF**, and **INDEF**.

The constant parameter w_i is input in *X* and may be used for a number of purposes. For example, if the parameter in an exponential model is known to depend upon the size of the area tested, volume of a radioactive mass, or population density, etc., then a multiplicative factor of the exponential parameter $\lambda = \exp(x\beta)$ may be known apriori. This factor can be input in w_i (w_i is the log of the factor). An alternate use of w_i is as follows: It may be that $\lambda = \exp(x_1\beta_1 + x_2\beta_2)$, where β_2 is known. Letting $w_i = x_2\beta_2$, estimates for β_1 can be obtained via **SVGLM** with the known fixed values for β_2. Standard methods can then be used to test hypotheses about β_2 via computed log-likelihoods.

Computational details

The computations proceed as follows:

1. The input arguments are checked for consistency and validity.

2. Estimates for the means of the explanatory variables *x* (as generated from the model specification via **GRGLM** (see Chapter 2, "Regression") are computed. Let f_i denote the frequency of the observation. Means are computed as

$$\overline{x} = \frac{\sum_i f_i x_i}{\sum_i f_i}$$

3. If **INIT** = 0, initial estimates of the parameters for all but the exponential models (**MODEL** = 0, 1) are are obtained as follows:

   A. Routine **KAPMR** is used to compute a nonparametric estimate of the survival probability at the upper limit of each failure interval. (Because upper limits are used, intervaland left-censored data are taken to be exact failures at the upper endpoint of the failure interval.) The Kaplan-Meier estimate is computed under the assumption that all failure distributions are identical (i.e., all βs but the intercept, if present, are assumed to be zero).

   B. If **INTCEP** = 0, a simple linear regression is performed predicting

$$S^{-1}(\hat{S}(t)) - w_i = \alpha + \phi t^*$$

   where t^* is computed at the upper endpoint of each failure interval, $t^* = t$ in models 3, 5, 7, and 9, and $t^* = \ln(t)$ in models 2, 4, 6, 8, and 10, and w_i is the fixed constant, if present. If **INTCEP** is zero, α is fixed at zero, and the model

$$S^{-1}(\hat{S}(t)) - \hat{\phi} t^* - w_i = x^T \beta$$

   is fit instead of the model above. In this model, the coefficients β are used in place of the location estimate α above. Here, $\hat{\phi}$ is estimated from the simple linear regression with $\alpha = 0$.

   C. If the intercept is in the model, then in log-location-scale models (models 1–8),

$$\hat{\sigma} = \hat{\phi}$$

   and the initial estimate of the intercept, if present, is taken to be $\hat{\alpha}$.
   In the Weibull model,

$$\hat{\theta} = 1 / \hat{\phi}$$

and the intercept, if present, is taken to be $\hat{\alpha}$.

Initial estimates of all parameters β, other than the intercept, are taken to be zero.

If no intercept is in the model, the scale parameter is estimated as above, and the estimates $\hat{\beta}$ from Step B are used as initial estimates for the β's.

For exponential models (**MODEL** = 0, 1), the average total time on test statistic is used to obtain an estimate for the intercept. Specifically, let T_t denote the total number of failures divided by the total time on test. The initial estimate for the intercept is then $\ln(T_t)$. Initial estimates for the remaining parameters β are taken as zero, and, if **MODEL** = 1, the initial estimate for the linear hazard parameter θ is taken to be a small positive number. When the intercept is not in the model, the initial estimate for the parameter θ is taken as a small positive number, and initial estimates of the parameters β are computed via multiple linear regression as above.

4. A quasi-Newton algorithm is used in the initial iterations based upon a Hessian estimate

$$\hat{H}_{\kappa_j \kappa_l} = \sum_i \ell'_{i\alpha_j} \ell'_{i\alpha_l}$$

where

$$\ell'_{i\alpha_j}$$

is the partial derivative of the *i*-th term in the log-likelihood with respect to the parameter α_j, and α_j denotes one of the parameters to be estimated.

When the relative change in the log-likelihood from one iteration to the next is 0.1 or less, exact second partial derivatives are used for the Hessian so that Newton-Raphson iteration is used.

If the initial step size results in an increase in the log-likelihood, the full step is used. If the log-likelihood decreases for the initial step size, the step size is halved, and a check for an increase in the log-likelihood performed. Step-halving is performed (as a simple line search) until an increase in the log-likelihood is detected, or until the step size is less that 0.0001 (where the initial step size is 1).

5. Convergence is assumed when the maximum relative change in any coefficient update from one iteration to the next is less than **EPS**, or when the relative change in the loglikelihood from one iteration to the next is less than **EPS**/100. Convergence is also assumed after **MAXIT** iterations, or when step halving leads to a step size of less than .0001, with no increase in the log-likelihood.

6. If requested (**INFIN** = 0), then the methods of Clarkson and Jennrich (1988) are used to check for the existence of infinite estimates in

$$\eta_i = x_i^T \beta$$

As an example of a situation in which infinite estimates can occur, suppose that observation *j* is right censored with $t_j > 15$ in a normal distribution model in which we fit the mean as

$$\mu_j = x_j^T \beta = \eta_j$$

where x_j is the observation design vector. If design vector x_j for parameter β_m is such that $x_{jm} = 1$ and $x_{im} = 0$ for all $i \neq j$, then the optimal estimate of β_m occurs at

$$\hat{\beta}_m = \infty$$

leading to an infinite estimate of both β_m and η_j. In **SVGLM**, such estimates may be "computed."

In all models fit by **SVGLM**, infinite estimates can only occur when the optimal estimated probability associated with the left- or right-censored observation is 1. If **INFIN** = 0, left- or right-censored observations that have estimated probability greater than 0.995 at some point during the iterations are excluded from the log-likelihood, and the iterations proceed with a log-likelihood based upon the remaining observations. This allows convergence of the algorithm when the maximum relative change in the estimated coefficients is small and also allows for a more precise determination of observations with infinite

$$\eta_i = x_i^T \beta$$

At convergence, linear programming is used to ensure that the eliminated observations have infinite η_i. If some (or all) of the removed observations should not have been removed (because their estimated η_i's must be finite), then the iterations are restarted with a log-likelihood based upon the finite η_i observations. See Clarkson and Jennrich (1988) for more details.

When **INFIN** = 1, no observations are eliminated during the iterations. In this case, when infinite estimates occur, some (or all) of the coefficient estimates $\hat{\beta}$ will become large, and it is likely that the Hessian will become (numerically) singular prior to convergence.

7. The case statistics are computed as follows:

   Let

$$\ell_i(\theta_i)$$

   denote the log-likelihood of the *i*-th observation evaluated at θ_i, ℓ'_i denote the vector of derivatives of ℓ_i with respect to all parameters,

$$\ell'_{\eta,i}$$

denote the derivative of ℓ_i with respect to $\eta = x^{\top}\beta$, H denote the Hessian, and E denote expectation. Then, the columns of CASE are:

A. Predicted values are computed as $E(T|x)$ according to standard formulas. If **MODEL** is 4 or 8, and if $\sigma \geq 1$, then the expected values cannot be computed because they are infinite.

B. Following Cook and Weisberg (1982), we take the influence (or leverage) of the i-th observation to be

$$(\ell'_i)H^{-1}\ell'_i$$

This quantity is a one-step approximation to the change in the estimates when the i-th observation is deleted (ignoring the nuisance parameters).

C. The "residual" is computed as

$$\ell'_{\hat{\eta},i}$$

D. The cumulative hazard is computed at the observation covariate values and, for interval observations, the upper endpoint of the failure interval. The cumulative hazard can also be used as a "residual" estimate. If the model is correct, the cumulative hazards should follow a standard exponential distribution. See Cox and Oakes (1984).

E. The density (for exact failures) or the interval probability (for censored observations) is computed for given x.

Programming Notes

Classification variables are specified by parameters **NCLVAR** and **INDCL**. Indicator variables are created for the classification variables using routine **GRGLM** (see Chapter 2, "Regression") with **IDUMMY** = 2.

Examples

Example 1

This example is from Lawless (1982, page 287) and involves the mortality of patients suffering from lung cancer. (The first ten rows of the input data are printed in the output.) An exponential distribution is fit for model

$$\eta = \mu + \beta_1 x_3 + \beta_2 x_4 + \beta_3 x_5 + \alpha_i + \gamma_k$$

where α_i is associated with a classification variable with 4 levels, and γ_k is associated with a classification variable with 2 levels. Note that because the computations are performed in single precision, there will be some small variation in the estimated coefficients across different machine environments.

```
      USE SVGLM_INT
      USE WRRRL_INT

      IMPLICIT   NONE
      INTEGER    ICEN, ILT, IPAR, IPRINT, IRT, LDCASE, LDCOEF, &
                 LDCOV, LDX, MAXCL, MODEL, NCLVAR, NCOL, NEF, NOBS
      PARAMETER  (ICEN=2, ILT=0, IPAR=0, IPRINT=2, IRT=1, LDCASE=40, &
                 LDCOEF=8, LDCOV=8, LDX=40, MAXCL=6, MODEL=0, &
                 NCLVAR=2, NCOL=7, NEF=5, NOBS=40)
      CHARACTER  *6 NUMBER(1)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                 NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(MAXCL), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), X(LDX,NCOL), XMEAN(LDCOV)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
      NUMBER(1) = 'NUMBER'
!
      CALL WRRRL ('First 10 rows of the input data.', X, &
                  NUMBER, NUMBER, 10, NCOL, LDX)
!
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, &
                  XMEAN, CASE, GR, IADDS, ICEN=ICEN, NCLVAR=NCLVAR, &
                  INDCL=INDCL, NEF=NEF, NVEF=NVEF, INDEF=INDEF, &
                  IPRINT=IPRINT, NCLVAL=NCLVAL, CLVAL=CLVAL, &
                  NRMISS=NRMISS)
!
      END
```

Output

First 10 rows of the input data.

	1	2	3	4	5	6	7
1	411.0	0.0	7.0	64.0	5.0	1.0	0.0
2	126.0	0.0	6.0	63.0	9.0	1.0	0.0
3	118.0	0.0	7.0	65.0	11.0	1.0	0.0
4	92.0	0.0	4.0	69.0	10.0	1.0	0.0
5	8.0	0.0	4.0	63.0	58.0	1.0	0.0
6	25.0	1.0	7.0	48.0	9.0	1.0	0.0
7	11.0	0.0	7.0	48.0	11.0	1.0	0.0
8	54.0	0.0	8.0	63.0	4.0	2.0	0.0
9	153.0	0.0	6.0	63.0	14.0	2.0	0.0
10	16.0	0.0	3.0	53.0	4.0	2.0	0.0

Initial Estimates

1	2	3	4	5	6	7	8
-5.054	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Method	Iteration	Step size	Maximum scaled coef. update	Log likelihood
Q-N	0			-224.0
Q-N	1	1.0000	0.9839	-213.4
N-R	2	1.0000	3.603	-207.3
N-R	3	1.0000	10.12	-204.3
N-R	4	1.0000	0.1430	-204.1
N-R	5	1.0000	0.1174E-01	-204.1

Log-likelihood -204.1392

Coefficient Statistics

	Coefficient	Standard error	Asymptotic z-statistic	Asymptotic p-value
1	-1.103	1.314	-0.8939	0.4016
2	-0.540	0.108	-4.995	0.0000
3	-0.009	0.020	-0.459	0.6460
4	-0.003	0.012	-0.291	0.7710
5	-0.363	0.445	-0.816	0.4149
6	0.127	0.486	0.261	0.7939
7	0.869	0.586	1.483	0.1385
8	0.270	0.388	0.695	0.4873

Asymptotic Coefficient Covariance

	1	2	3	4	5
1	1.727	-8.1873E-02	-1.9753E-02	-2.2481E-03	-6.5707E-02
2		1.1690E-02	6.4506E-05	2.8955E-04	-3.8734E-04
3			3.8676E-04	-3.9067E-05	-1.2359E-03
4				1.3630E-04	7.5656E-04
5					0.1976

	6	7	8
1	-0.1038	-0.1554	-4.2370E-05
2	8.5772E-03	1.8120E-02	6.5272E-03
3	-3.2789E-04	-1.6986E-03	-2.7794E-03
4	-1.6742E-03	6.2668E-04	1.5432E-03
5	9.0035E-02	0.1122	4.3157E-02
6	0.2365	0.1142	-1.3527E-02
7		0.3436	5.1948E-02

8 0.1507

Case Analysis

	Predicted	Influence	Residual	Cumulative Hazard	Density or Probability
1	262.7	0.0450	-0.565	1.565	0.0008
2	153.8	0.0042	0.181	0.819	0.0029
3	270.5	0.0482	0.564	0.436	0.0024
4	55.3	0.0844	-0.663	1.663	0.0034
5	61.7	0.3765	0.870	0.130	0.0142
6	230.4	0.0025	-0.108	0.108	0.8972
7	232.0	0.1960	0.953	0.047	0.0041
8	272.8	0.1677	0.802	0.198	0.0030
9	95.9	0.0505	-0.596	1.596	0.0021
10	16.8	0.0005	0.045	0.955	0.0230
11	234.0	0.1911	0.761	0.239	0.0034
12	29.1	0.0156	0.278	0.722	0.0167
13	102.2	0.4609	-1.807	2.807	0.0006
14	34.8	0.0686	0.713	0.287	0.0216
15	5.3	0.0838	-0.521	1.521	0.0415
16	25.7	0.0711	0.533	0.467	0.0244
17	65.6	0.4185	-1.698	2.698	0.0010
18	38.4	0.0886	0.688	0.312	0.0191
19	261.0	0.0155	0.234	0.766	0.0018
20	167.2	0.0338	-0.495	1.495	0.0013
21	85.8	0.0082	-0.166	1.166	0.0036
22	947.8	0.0005	-0.054	1.054	0.0004
23	105.9	0.6402	-2.181	2.181	0.1129
24	305.2	0.5757	-2.247	3.247	0.0001
25	24.6	0.3203	0.959	0.041	0.0390
26	572.8	0.0762	0.649	0.351	0.0012
27	217.5	0.1246	0.798	0.202	0.0038
28	96.6	0.1494	0.845	0.155	0.0089
29	173.4	0.1096	-0.594	0.594	0.5522
30	38.7	0.1928	0.948	0.052	0.0245
31	22.5	0.0040	0.112	0.888	0.0183
32	30.7	0.2270	-0.661	1.661	0.0062
33	20.8	0.0058	0.134	0.866	0.0202
34	54.6	0.1094	-0.648	1.648	0.0035
35	168.6	0.0923	0.502	0.498	0.0036
36	256.8	0.0341	0.361	0.639	0.0021
37	21.9	0.0069	0.134	0.866	0.0192
38	124.3	0.0680	0.654	0.346	0.0057
39	417.9	0.0087	0.186	0.814	0.0011
40	257.1	0.0025	0.101	0.899	0.0016

Last Coefficient Update

1	2	3	4	5	6
-1.031E-05	-1.437E-06	3.098E-07	4.722E-08	-1.844E-05	-1.671E-06

7	8
-2.520E-06	8.139E-06

Covariate Means

1	2	3	4	5	6	7
5.65	56.58	15.65	0.35	0.28	0.12	0.53

Distinct Values For Each Class Variable

Variable 1:	1.0	2.0	3.0	4.0
Variable 2:	0.	1.0		

```
                              Observation Codes
   1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
   0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

  21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
   0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

   Number of Missing Values           0
```

Example 2

As a second example, the **MAXIT** = 0 option is used for the model in Example 1 with the coefficients restricted such that **μ** = **-**1.25, β_1 = **-**6, and the remaining 6 coefficients are zero. A chi-squared statistic with 8 degrees of freedom for testing that the coefficients are specified as above (versus the alternative that they are not as specified) may be computed from the output as

$$\chi^2 = g^T \hat{\Sigma}^{-1} g$$

where $\hat{\Sigma}$ is output in **COV**, and *g* is output in **GR**. The resulting test statistic (6.107), based upon no iterations, is comparable to the likelihood ratio test statistic that may be computed from the log-likelihood output in Example 2 (**-**206.6835) and the log-likelihood output in Example 1 (**-**204.1392).

$$\chi^2_{LR} = 2\left(206.6835 - 204.1392\right) = 5.0886$$

Neither test statistic is significant at the **α** = 0.05 level.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    ICEN, ILT, INIT, IPAR, IPRINT, IRT, LDCASE, &
                 LDCOEF, LDCOV, LDX, MAXCL, MAXIT, MODEL, NCLVAR, &
                 NCOL, NEF, NOBS
      PARAMETER  (ICEN=2, ILT=0, INIT=1, IPAR=0, IPRINT=2,IRT=1, &
                  LDCASE=40, LDCOEF=8, LDCOV=8, LDX=40, MAXCL=6, &
                  MAXIT=0, MODEL=0, NCLVAR=2, NCOL=7, NEF=5, NOBS=40)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), IRANK, &
                 NCLVAL(NCLVAR), NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CHISQ, CLVAL(MAXCL), &
                 COEF(LDCOEF,4), COV(LDCOV,LDCOV), GR(LDCOV), &
                 GRD(LDCOV), X(LDX,NCOL), XMEAN(LDCOV)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
```

```
           25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
           22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
           5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      NCOEF = 8
      CALL SSET (NCOEF, 0.0, COEF(3:,1), 1)
      CALL ISET (NOBS, 0, IADDS, 1)
      COEF(1,1) = -1.25
      COEF(2,1) = -0.60
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, COV, &
                  XMEAN, CASE, GR(1:1), IADDS, ICEN=ICEN, MAXIT=MAXIT, &
                  NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, NVEF=NVEF,&
                  INDEF=INDEF, INIT=INIT, IPRINT=IPRINT, &
                  NCLVAL=NCLVAL, CLVAL=CLVAL)
!                                 Compute Chi-squared
      CALL CHFAC (COV, IRANK, COV)
      CALL GIRTS (COV, GR, IRANK, GRD, IPATH=2)
!
      CHISQ = SDOT(NCOEF,GRD(1:1), 1, GRD(1:1), 1)
      WRITE (6,99999) ' Chi-squared statistic with 8 degrees of '// &
                      'freedom ', CHISQ
!
99999 FORMAT (/, A, G12.4)
      END
```

Output

```
 Log-likelihood       -206.6835

                     Coefficient Statistics
                         Standard     Asymptotic     Asymptotic
      Coefficient           error    z-statistic        p-value
 1          -1.25           1.383         -0.904          0.366
 2          -0.60           0.112         -5.365          0.000
 3           0.00           0.021          0.000          1.000
 4           0.00           0.011          0.000          1.000
 5           0.00           0.429          0.000          1.000
 6           0.00           0.530          0.000          1.000
 7           0.00           0.775          0.000          1.000
 8           0.00           0.405          0.000          1.000

                                  Hessian
              1              2              3              4              5
 1      1.914      -8.1835E-02    -2.3464E-02    -1.1634E-03    -9.0646E-02
 2                  1.2507E-02     2.0883E-06     3.1320E-04    -5.3147E-04
 3                                 4.6174E-04    -5.5344E-05    -8.1929E-04
 4                                                1.1797E-04     6.0699E-04
 5                                                               0.1839

                6              7              8
 1    -0.1641        -0.1681         7.7768E-02
 2     1.0372E-02     1.9269E-02     5.9762E-03
 3     5.1246E-04    -1.6419E-03    -4.0106E-03
 4    -2.0693E-03     6.9029E-04     1.7020E-03
 5     9.9640E-02     0.1191         3.5786E-02
 6     0.2808         0.1264        -2.2602E-02
 7                    0.6003         4.6015E-02
 8                                   0.1641
```

```
 Estimated Probability (censored) or Estimated Density (non-censored)
      1        2        3        4        5        6        7        8
 0.0007   0.0029   0.0026   0.0024   0.0211   0.8982   0.0041   0.0021

      9       10       11       12       13       14       15       16
 0.0024   0.0222   0.0021   0.0151   0.0008   0.0200   0.0433   0.0120

     17       18       19       20       21       22       23       24
 0.0011   0.0190   0.0015   0.0015   0.0036   0.0004   0.0371   0.0001

     25       26       27       28       29       30       31       32
 0.0792   0.0015   0.0055   0.0115   0.6424   0.0247   0.0184   0.0042

     33       34       35       36       37       38       39       40
 0.0163   0.0039   0.0019   0.0021   0.0193   0.0056   0.0011   0.0016

                    Newton-Raphson Step
     1       2       3       4       5       6       7       8
 0.171   0.062  -0.011  -0.003  -0.336   0.133   1.297   0.298

                  Covariate Means
    1       2       3       4       5       6       7
 5.65   56.58   15.65    0.35    0.28    0.12    0.53

 Distinct Values For Each Class Variable
 Variable  1:     1.0         2.0         3.0         4.0
 Variable  2:      0.         1.0

                               Observation Codes
1  2  3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

21 22 23 24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

Number of Missing Values           0

Chi-squared statistic with 8 degrees of freedom    6.107
```

STBLE

Estimates survival probabilities and hazard rates for various parametric models.

Required Arguments

XPT — NOBS by NCOL matrix, each row of which contains the covariates for a group for which survival estimates are desired. (Input)

MODEL — Model option parameter. (Input)
MODEL specifies the distribution of the response variable and the relationship of the linear model to a distribution parameter.

	Distribution
0	Exponential
1	Linear hazard
2	Log-normal
3	Normal
4	Log-logistic
5	Logistic
6	Log least extreme value
7	Least extreme value
8	Log extreme value
9	Extreme value
10	Weibull

For further discussion of the models, see the “Description” section.

TIME — Beginning of the time grid for which the survival estimates are desired. (Input)
Survival probabilities and hazard rates are computed for each covariate vector over the grid of time points TIME + *i* * DELTA for *i* = 0, 1, …, NPT - 1.

NPT — Number of points on the time grid for which survival probabilities are desired. (Input)

DELTA — Increment between time points on the time grid. (Input)

IFIX — Column number in XPT containing a constant to be added to the linear response. (Input)
The estimated linear response is *w* + COEF(1) * *z*(1) + COEF(2) * *z*(2) + … + COEF(NCOEF) * *z*(NCOEF), where *z* is the design vector for the I-th observation obtained from a row of XPT. *w* = XPT(I, IFIX) if IFIX is positive, and *w* = 0 otherwise.

NCOEF — Number of coefficients in the model. (Input)

COEF — Vector of length NCOEF containing the model parameter estimates. (Input)
Usually routine SVGLM is first called to estimate COEF as the first column of matrix COEF in SVGLM. When present in the model, the initial coefficient in COEF is a "nuisance" parameter, and the remaining coefficients are parameters associated with the "linear" model, beginning with the intercept, if present. Nuisance parameters are as follows:

	Nuisance Parameter
1	Coefficient of the quadratic term in time, θ
2 – 9	Scale parameter, σ
10	Shape parameter, θ

There is no nuisance parameter for model 0.

SPROB — NPT by 2 * NOBS + 1 matrix. (Output)
SPROB(*i*, 2) contains the estimated survival probability at time SPROB(*i*, 1) = TIME + (*i* - 1) * DELTA for observations with covariates given in row 1 of XPT. SPROB(*i*, 3) contains the estimate for the hazard rate at this time point. Columns 4 and 5 contain the estimated survival probabilities and hazard rates for observations with covariates given in the second row in XPT, etc., up to columns 2 * NOBS and 2 * NOBS + 1, which contain these statistics for observations with covariates in the last row of XPT.

XBETA — Vector of length NOBS containing the estimated linear response *w* + COEF(1) * *z*(1) + … + COEF(NCOEF) * *z*(NCOEF) for each row of XPT. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (XPT,1).

NCOL — Number of columns in XPT. (Input)
Default: NCOL = size (XPT,2).

LDXPT — Leading dimension of XPT exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (XPT,1).

INTCEP — Intercept option. (Input)
Default: INTCEP = 1.

INTCEP	Action
0	No intercept is in the model (unless otherwise provided for by the user).
1	An intercept is automatically included in the model.

NCLVAR — Number of classification variables. (Input)
Dummy or indicator variables are generated for classification variables using the IDUMMY = 2 option of routine GRGLM (see Chapter 2, "Regression"). See also Comment 2.
Default: NCLVAR = 0.

INDCL — Index vector of length NCLVAR containing the column numbers of X that are classification variables. (Input, if NCLVAR is positive, not used otherwise)
If NCLVAR is 0, INDCL is not referenced and can be dimensioned of length 1 in the calling program.

NCLVAL — Vector of length NCLVAR containing the number of values taken on by each classification variable. (Input, if NCLVAR is positive, not referenced otherwise)
NCLVAL(I) is the number of distinct values for the I-th classification variable. NCLVAL is not referenced and can be dimensioned of length 1 in the calling program if NCLVAR is zero.

CLVAL — Vector of length NCLVAL(1) + NCLVAL(2) + ... + NCLVAL(NCLVAR) containing the distinct values of the classification variables. (Input, if NCLVAR is positive; not used otherwise)
The first NCLVAL(1) elements contain the values for the first classification variables, the next NCLVAL(2) elements contain the values for the second classification variable, etc. If NCLVAR is zero, then CLVAL is not referenced and can be dimensioned of length 1 in the calling program.

NEF — Number of effects in the model. (Input)
In addition to effects involving classification variables, simple covariates and the product of simple covariates are also considered effects.
Default: NEF = 0.

NVEF — Vector of length NEF that contains the number of variables associated with each effect. (Input, if NEF is greater than 0; not referenced otherwise)
NVEF is not referenced and can be dimensioned of length 1 in the calling program if NEF is zero.

INDEF — Vector of length NVEF(1) + … + NVEF(NEF) that contains the column numbers in X associated with each effect. (Input, if NEF is greater than 0; not used otherwise)
The first NVEF(1) elements of INDEF contain the column numbers in XPT for the variables in the first effect. The next NVEF(2) elements in INDEF contain the column numbers for the second effect, etc. If NCLVAR is zero, INDEF is not referenced and can be dimensioned of length 1 in the calling program.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed.

LDSPRO — Leading dimension of SPROB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSPRO = size (SPROB,1).

FORTRAN 90 Interface

Generic: CALL STBLE (XPT, MODEL, TIME, NPT, DELTA, IFIX, NCOEF, COEF, SPROB, XBETA [, ...])
Specific: The specific interface names are S_STBLE and D_STBLE.

FORTRAN 77 Interface

Single: CALL STBLE (NOBS, NCOL, XPT, LDXPT, MODEL, TIME, NPT, DELTA, IFIX, INTCEP, NCLVAR, INDCL, NCLVAL, CLVAL, NEF, NVEF, INDEF, NCOEF, COEF, IPRINT, SPROB, LDSPRO, XBETA)
Double: The double precision name is DSTBLE.

Description

Routine STBLE computes estimates of survival probabilities and hazard rates for the parametric survival/reliability models fit by routine SVGLM for one or more vectors of covariate values. Because estimates for the parameters of the model must be given, routine SVGLM is usually invoked to obtain these estimates prior to invoking STBLE.

Let $\eta = x^T\beta$ be the linear parameterization, where x is a design vector obtained in STBLE via routine GRGLM (see Chapter 2, "Regression") from a row of XPT, and β is a vector of parameters associated with the linear model. Let T denote the random response variable and $S(t)$ denote the probability that $T > t$. All models considered also allow a fixed parameter w (input in column IFIX of XPT). Use of this parameter is discussed in the document for routine SVGLM. There may also be nuisance parameters $\theta > 0$, or $\sigma > 0$. Let Φ denote the cumulative normal distribution. The survival models available in STBLE are

Model	Name	*S*(*t*)
0	Exponential	$\exp\{-t\exp(w+\eta)\}$
1	Linear hazard	$\exp\{-(t+\frac{\theta t^2}{2})\exp[(w+\eta)]\}$
2	Log-normal	$1-\Phi(\frac{\ln(t)-\eta-w}{\sigma})$
3	Normal	$1-\Phi(\frac{t-\eta-w}{\sigma})$
4	Log-logistic	$\{1+\exp(\frac{\ln(t)-\eta-w}{\sigma})\}^{-1}$
5	Logistic	$\{1+\exp(\frac{t-\eta-w}{\sigma})\}^{-1}$
6	Log least extreme value	$\exp\{-\exp(\frac{\ln(t)-\eta-w}{\sigma})\}$
7	Least extreme value	$\exp\{-\exp(\frac{t-\eta-w}{\sigma})\}$
8	Log extreme value	$1-\exp\{-\exp[-(\frac{\ln(t)-\eta-w}{\sigma})]\}$
9	Extreme value	$1-\exp\{-\exp[-(\frac{t-\eta-w}{\sigma})]\}$
10	Weibull	$\exp\{-\left(\frac{t}{\exp(w+\eta)}\right)^{\theta}\}$

Let $\lambda(t)$ denote the hazard rate at time t. Then $\lambda(t)$ and $S(t)$ are related as

$$S(t) = \exp\{-\int_{-\infty}^{t} \lambda(s)\,ds\}$$

Models 0, 1, 2, 4, 6, 8, and 10 require that $T > 0$ (in which case, we assume $\lambda(s) = 0$ for $s < 0$), while the remaining models allow arbitrary values for T, $-\infty < T < \infty$. The computations proceed in routine **STBLE** as follows:

1. The input arguments are checked for consistency and validity.
2. For each row of **XPT**, the explanatory variables are generated from the classification and variables and the covariates using routine **GRGLM** with the **IDUMMY** = 2 option. (When **IDUMMY** is two, **GRGLM** assigns an indicator variable the value 1.0 when the observation is in the class, assigns the value 0.0 otherwise, and omits the last indicator variable from the design vector. See the manual documentation for **GRGLM**.) Given the explanatary variables x, η is computed as $\eta = x^{\top}\beta$, where β is input in **COEF**.

3. For each time point requested in the time grid, the survival probabilities and hazard rates are computed.

Comments

1. Workspace may be explicitly provided, if desired, by use of S2BLE/DS2BLE. The reference is:

   CALL S2BLE (NOBS, NCOL, XPT, LDXPT, MODEL, TIME, NPT, DELTA, IFIX, INTCEP, NCLVAR, INDCL, NCLVAL, CLVAL, NEF, NVEF, INDEF, NCOEF, COEF, IPRINT, SPROB, LDSPRO, XBETA, CHWK, Z, RWK)

   The additional arguments are as follows:

   CHWK — CHARACTER * 10 work vector of length NCOL.

   Z — Work vector of length NCOEF.

   RWK — Work vector of length MAX(7, NCOL) if IPRINT = 1, or of length 1 if IPRINT = 0.

2. Dummy variables are generated for the classification variables as follows: The list of all distinct values of each classification variable is as stored in CLVAL. Dummy variables are generated for each but the last of these distinct values. Each dummy variable is zero unless the classification variable equals the list value corresponding to the dummy variable, in which case the dummy variable is one. See argument IDUMMY for IDUMMY = 2 in routine GRGLM (see Chapter 2, "Regression").

3. Informational errors

Type	Code	Description
3	1	Some survival probabilities are less than or equal to zero. The corresponding hazard values cannot be computed.
4	2	The specified number of coefficients, NCOEF, is incorrect.
4	3	The model specified is not defined for negative time.

Example

The example is a continuation of the first example given for routine SVGLM. Prior to calling STBLE, SVGLM is invoked to compute the parameter estimates. The example is taken from Lawless (1982, page 287) and involves the mortality of patients suffering from lung cancer.

```
    !
          USE SVGLM_INT
          USE SCOPY_INT
          USE STBLE_INT

          IMPLICIT   NONE

          INTEGER    ICEN, IFIX, ILT, INFIN, IPRINT, IRT, LDCASE, &
                     LDCOEF, LDCOV, LDSPRO, LDX, LDXPT, MAXCL, &
                     MODEL, NCLVAR, NCOL, NEF, NOBS, NPT
```

```
      REAL       DELTA, TIME, XPWR
      PARAMETER  (DELTA=20.0, ICEN=2, IFIX=0, ILT=0, INFIN=0, IPRINT=1, &
                  IRT=1, LDCASE=40, LDCOEF=9, LDCOV=9, LDX=40, LDXPT=2, &
                  MAXCL=6, MODEL=0, NCLVAR=2, NCOL=7, NEF=5, NOBS=40, &
                  NPT=10, TIME=10.0, XPWR=0.0, LDSPRO=NPT)
!
      INTEGER    IADDS(NOBS), INDCL(NCLVAR), INDEF(5), NCLVAL(NCLVAR), &
                NCOEF, NRMISS, NVEF(NEF)
      REAL       ALGL, CASE(LDCASE,5), CLVAL(MAXCL), COEF(LDCOEF,4), &
                COV(LDCOV,LDCOV), GR(LDCOV), SPROB(LDSPRO,2*NOBS+1), &
                X(LDX,NCOL), XBETA(NOBS), XMEAN(LDCOV), XPT(LDXPT,NCOL)
!
      DATA X/411, 126, 118, 92, 8, 25, 11, 54, 153, 16, 56, 21, 287, &
          10, 8, 12, 177, 12, 200, 250, 100, 999, 231, 991, 1, 201, &
          44, 15, 103, 2, 20, 51, 18, 90, 84, 164, 19, 43, 340, 231, &
          5*0, 1, 16*0, 1, 5*0, 1, 11*0, 7, 6, 7, 4, 4, 7, 7, 8, 6, &
          3, 8, 4, 6, 4, 2, 5, 5, 4, 8, 7, 6, 9, 5, 7, 2, 8, 6, 5, 7, &
          4, 3, 3, 4, 6, 8, 7, 3, 6, 8, 7, 64, 63, 65, 69, 63, 48, &
          48, 63, 63, 53, 43, 55, 66, 67, 61, 63, 66, 68, 41, 53, 37, &
          54, 52, 50, 65, 52, 70, 40, 36, 44, 54, 59, 69, 50, 62, 68, &
          39, 49, 64, 67, 5, 9, 11, 10, 58, 9, 11, 4, 14, 4, 12, 2, &
          25, 23, 19, 4, 16, 12, 12, 8, 13, 12, 8, 7, 21, 28, 13, 13, &
          22, 36, 9, 87, 5, 22, 4, 15, 4, 11, 10, 18, 7*1, 7*2, 2*3, &
          5*4, 7*1, 4*2, 3*3, 5*4, 21*0, 19*1/
      DATA NVEF/1, 1, 1, 1, 1/, INDEF/3, 4, 5, 6, 7/, INDCL/6, 7/
!
      CALL SVGLM (X, MODEL, ILT, IRT, MAXCL, NCOEF, COEF, ALGL, &
                  COV, XMEAN, CASE, GR, IADDS, ICEN=ICEN, &
                  NCLVAR=NCLVAR, INDCL=INDCL, NEF=NEF, NVEF=NVEF, &
                  INDEF=INDEF, NCLVAL=NCLVAL, CLVAL=CLVAL)
!
      CALL SCOPY (NCOL, X(1:,1), LDX, XPT(1:,1), LDXPT)
      CALL SCOPY (NCOL, X(2:,1), LDX, XPT(2:,1), LDXPT)
!
      CALL STBLE (XPT, MODEL, TIME, NPT, DELTA, IFIX, &
                  NCOEF, COEF, SPROB, XBETA, NCLVAR=NCLVAR, INDCL=INDCL,&
                  NCLVAL=NCLVAL, CLVAL=CLVAL, NEF=NEF, NVEF=NVEF,&
                  INDEF=INDEF, IPRINT=IPRINT)
!
      END
```

Output

```
                              group   1
                               xpt
  1          2          3          4          5          6
411          0          7         64          5          1

7
0

                        design vector
1          2          3          4          5          6
1          7         64          5          1          0

7          8
0          1
```

```
xbeta =        -5.57097

                         group   2
                           xpt
  1           2           3           4           5           6
126           0           6          63           9           1

7
0

                      design vector
1           2           3           4           5           6
1           6          63           9           1           0

7           8
0           1

xbeta =        -5.03551

              survival and hazard estimates
                         (sprob)
  time           s1           h1           s2           h2
 10.00       0.9626     0.003807       0.9370     0.006503
 30.00       0.8921     0.003807       0.8228     0.006503
 50.00       0.8267     0.003807       0.7224     0.006503
 70.00       0.7661     0.003807       0.6343     0.006503
 90.00       0.7099     0.003807       0.5570     0.006503
110.00       0.6579     0.003807       0.4890     0.006503
130.00       0.6096     0.003807       0.4294     0.006503
150.00       0.5649     0.003807       0.3770     0.006503
170.00       0.5235     0.003807       0.3310     0.006503
190.00       0.4852     0.003807       0.2907     0.006503
```

Note that in simple exponential models the hazard rate is constant over time.

ACTBL

Produces population and cohort life tables.

Required Arguments

IMTH — Type of life table. (Input)
IMTH = 0 indicates a population (current) table. IMTH = 1 indicates a cohort table.

N — Number of age classes. (Input)

NPOP — Population size. (Input, if IMTH = 0; not used otherwise)
For IMTH = 0, the population size at the beginning of the first age interval. The value is somewhat arbitrary. NPOP = 10000 is reasonable. Not used if IMTH = 1.

AGE — Vector of length N + 1 containing the lowest age in each age interval, and in AGE(N + 1), the end-point of the last age interval. (Input)
Negative AGE(1) indicates that the age intervals are all of length |AGE(1)| and that the initial age interval is from 0.0 to |AGE(1)|. In this case, all other elements of AGE need not be specified. AGE(N + 1) need not be specified when IMTH = 1.

A — Vector of length N containing the fraction of those dying within each interval who die before the interval midpoint. (Input)
A common choice for all A(I) is 0.5. This choice may also be specified by setting A(1) to any negative value. In this case, the remaining values of A need not be specified.

IPOP — Vector of length N containing the cohort sizes during each interval. (Input)
If IMTH = 0, then IPOP(I) contains the size of the population at the midpoint of interval I. If IMTH = 1, then IPOP(I) contains the size of the cohort at the beginning of interval I. When IMTH = 0, the population sizes in IPOP may need to be adjusted to correspond to the number of deaths in IDTH . See the "Description" section of the KAPMR routine for more information.

IDTH — Vector of length N containing the number of deaths in each age interval. (Input, if IMTH = 0; not used otherwise)
If IMTH = 1, IDTH is not used and may be dimensioned of length 1.

TABLE — N by 12 matrix containing the life table. (Output)
The rows of TABLE correspond to the age intervals.

	Description
1	Lowest age in the age interval.
2	Fraction of those dying within the interval who die before the interval midpoint.
3	Number surviving to the beginning of the interval.
4	Number of deaths in the interval.
5	Death rate in the interval. If IMTH = 1, this column is set to NaN (not a number).
6	Death rate in the interval. If IMTH = 1, this column is set to NaN (not a number).
7	Proportion dying in the interval.
8	Standard error of the proportion dying in the interval.
9	Standard error of the proportion of survivors at the beginning of the interval.
10	Expected lifetime at the beginning of the interval.
11	Standard error of the expected life at the beginning of the interval.
12	Total number of time units lived by all of the population in the interval.

Optional Arguments

IPRINT — Printing option. (Input)
If IPRINT = 1, the life table is printed. Otherwise, no printing is done.
Default: IPRINT = 0.

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, TABLE [, ...])
Specific: The specific interface names are S_ACTBL and D_ACTBL.

FORTRAN 77 Interface

Single: CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, IPRINT TABLE, LDTABL)
Double: The double precision name is DACTBL.

Description

Routine **ACTBL** computes population (current) or cohort life tables based upon the observed population sizes at the middle (**IMTH** = 0) or the beginning (**IMTH** = 1) of some userspecified age intervals. The number of deaths in each of these intervals must also be observed.

The probability of dying prior to the middle of the interval, given that death occurs somewhere in the interval, may also be specified. Often, however, this probability is taken to be 0.5. For a discussion of the probability models underlying the life table here, see the references.

Let t_i, for i = 0, 1, ..., t_n denote the time grid defining the n age intervals, and note that the length of the age intervals may vary. Following Gross and Clark (1975, page 24), let d_i denote the number of individuals dying in age interval i, where age interval i ends at time t_i. If **IMTH** = 0, the death rate at the middle of the interval is given by $r_i = d_i/(M_i h_i)$, where M_i is the number of individuals alive at the middle of the interval, and $h_i = t_i - t_{i-1}$, $t_0 = 0$. The number of individuals alive at the beginning of the interval may be estimated by $P_i = M_i + (1 - a_i)d_i$ where a_i is the probability that an individual dying in the interval dies prior to the interval midpoint. When **IMTH** = 1, P_i is input directly while the death rate in the interval, r_i, is not needed.

The probability that an individual dies during the age interval from t_{i-1} to t_i is given by $q_i = d_i/P_i$. It is assumed that all individuals alive at the beginning of the last interval die during the last interval. Thus, $q_n = 1.0$. The asymptotic variance of q_i can be estimated by

$$\sigma_i^2 = q_i\left(1 - q_i\right)P_i$$

When **IMTH** = 0, the number of individuals alive in the middle of the time interval (input in **IPOP**(**I**)) must be adjusted to correspond to the number of deaths observed in the interval. Routine **ACTBL** assumes that the number of deaths observed in interval h_i occur over a time period equal to h_i. If d_i is measured over a period u_i, where $u_i \neq d_i$, then **IPOP**(**I**) must be adjusted to correspond to d_i by multiplication by u_i/h_i, i.e., the value M_i input into **ACTBL** as **IPOP**(**I**) is computed as

$$M_i^* = M_i u_i / h_i$$

Let S_i denote the number of survivors at time t_i from a hypothetical (**IMTH** = 0) or observed (**IMTH** = 1) population. Then, S_0 = **NPOP** when **IMTH** = 0, and S_0 = **IPOP**(1) for **IMTH** = 1, and S_i is given by $S_i = S_{i-1} - \delta_{i-1}$ where $\delta_i = S_i q_i$ is the number of individuals who die in the i-th interval. The proportion of survivors in the interval is given by $V_i = S_i/S_0$ while the asymptotic variance of V_i can be estimated as follows.

$$\text{var}(V_i) = V_i^2 \sum_{j=1}^{i-1} \frac{\sigma_j^2}{(1-q_j)^2}$$

The expected lifetime at the beginning of the interval is calculated as the total lifetime remaining for all survivors alive at the beginning of the interval divided by the number of survivors at the beginning of the interval. If e_i denotes this average expected lifetime, then the variance of e_i can be estimated as (see Chiang 1968)

$$\text{var}(e_i) = \frac{\sum_{j=i}^{n-1} P_j^2 \sigma_j^2 [e_{j+1} + h_{j+1}(1-a_j)]^2}{P_j^2}$$

where var(e_n) = 0.0.

Finally, the total number of time units lived by all survivors in the time interval can be estimated as:

$$U_i = h_i \left[S_i - \delta_i \left(1 - a_i \right) \right]$$

Example

The following example is taken from Chiang (1968). The cohort life table has thirteen equally spaced intervals, so AGE(1) is set to -5.0. Similarly, the probabilities of death prior to the middle of the interval are all taken to be 0.5, so A(1) is set to -1.0. Since IPRINT = 1, the life table is printed by ACTBL.

```
      USE ACTBL_INT

      IMPLICIT   NONE

      INTEGER    IMTH, IPRINT, LDTABL, N, NPOP
      PARAMETER  (IMTH=1, IPRINT=1, N=13, NPOP=10000, LDTABL=N)
!
      INTEGER    IDTH(13), IPOP(13)
      REAL       A(1), AGE(1), TABLE(13,12)
!
      DATA AGE/-5.0/, A/-1.0/
      DATA IPOP/270, 268, 264, 261, 254, 251, 248, 232, 166, 130, 76, &
          34, 13/
!
      CALL ACTBL (IMTH, N, NPOP, AGE, A, IPOP, IDTH, TABLE, &
                  IPRINT=IPRINT)
!
      END
```

Output

```
                              Life Table
 Age Class         Age      PDHALF       Alive      Deaths  Death Rate
         1           0         0.5         270           2         NaN
         2           5         0.5         268           4         NaN
```

3	10	0.5	264	3	NaN
4	15	0.5	261	7	NaN
5	20	0.5	254	3	NaN
6	25	0.5	251	3	NaN
7	30	0.5	248	16	NaN
8	35	0.5	232	66	NaN
9	40	0.5	166	36	NaN
10	45	0.5	130	54	NaN
11	50	0.5	76	42	NaN
12	55	0.5	34	21	NaN
13	60	0.5	13	13	NaN

Age Class	P(D)	Std(P(D))	P(S)	Std(P(S))	Lifetime
1	0.007	0.00522	1.000	0.00000	43.19
2	0.015	0.00741	0.993	0.00522	38.49
3	0.011	0.00652	0.978	0.00897	34.03
4	0.027	0.01000	0.967	0.01092	29.40
5	0.012	0.00678	0.941	0.01437	25.14
6	0.012	0.00686	0.930	0.01557	20.41
7	0.065	0.01560	0.919	0.01665	15.62
8	0.284	0.02962	0.859	0.02116	11.53
9	0.217	0.03199	0.615	0.02962	10.12
10	0.415	0.04322	0.481	0.03041	7.23
11	0.553	0.05704	0.281	0.02737	5.59
12	0.618	0.08334	0.126	0.02019	4.41
13	1.000	0.00000	0.048	0.01303	2.50

Age Class	Std(Life)	Time Units
1	0.6993	1345.0
2	0.6707	1330.0
3	0.6230	1312.5
4	0.5940	1287.5
5	0.5403	1262.5
6	0.5237	1247.5
7	0.5149	1200.0
8	0.4982	995.0
9	0.4602	740.0
10	0.4328	515.0
11	0.4361	275.0
12	0.4167	117.5
13	0.0000	32.5

Multidimensional Scaling

Routines

Usage Notes

The routines described in this chapter all involve multidimensional scaling. Routine MSIDV performs computations for the individual differences metric scaling models. The utility routines are useful for associated computations as well as for programming other methods of multidimensional scaling.

The following is a brief introduction to multidimensional scaling meant to acquaint the user with the purposes of the routines described in this chapter. Also of interest is the table at the end of this section giving the notation used. A more complete description of procedures in multidimensional scaling may be found in the references, as well as in the algorithm sections for the routines.

Multidimensional Scaling Data Types

A "dissimilarity" is a subject's measure of the "distance" between two objects. For example, a subject's estimate of the distance between two cities is a dissimilarity measure that may, or may not, be the actual distance between the cities (depending upon the subjects familiarity with the two cities). Dissimilarities usually have less relationship to distance. For example, the subject may estimate, on a given scale, the difference between two smells, two tastes, two colors, two shapes, etc. As a concrete example, the subject is asked to compare two wines and indicate whether they have very similar tastes (scale value 0), or very different tastes (scale value 10), or are somewhere in between. In this case, no objective measure of "distance" is available, yet the dissimilarity may be measured. In all cases, however, the larger the difference between the objects, the larger the dissimilarity measure.

If instead the measure increases as the objects become more similar, then a "similarity" measure rather than a "dissimilarity" measure is obtained. Most routines in this chapter require dissimilarities as input so that similarities must be converted to dissimilarities before most routines in this chapter can be used. Routine MSSTN provides two common methods for performing these conversions.

In general, dissimilarities between all objects in a set are measured (yielding a matrix of dissimilarities), and the multidimensional scaling problem is to locate the objects in a Euclidean (or other) space of known dimension given the matrix of dissimilarities. The estimates of object locations should yield predicted distances between the objects that "closely approximate" the observed dissimilarities. In many multidimensional scaling methods, "closely approximates" means that a predefined measure of the discrepancy (the "stress") is minimized. The simplest stress measure is the sum of the squared differences between the observed dissimilarities and the distances predicted by the estimated object locations. This stress measure, as well as all other stress measures used in this chapter, is discussed more fully in the manual document for routine MSTRS.

Note that the predicted distances between objects may not be Euclidean distance. Indeed, in one of the more popular multidimensional scaling models, the individual differences model, weighted Euclidean distance is used. Let $\boldsymbol{\lambda}_{1k}$ and $\boldsymbol{\lambda}_{2k}$, $k = 1, \ldots, d$, be the location estimates of two objects (stimuli) in a d dimensional space. Then, the weighted Euclidean distance used in the individual difference model is given by

$$\delta_{12} = \sqrt{\sum_{k=1}^{d} w_k \left(\lambda_{1k} - \lambda_{2k} \right)^2}$$

Many other distance models are possible. The models used in this chapter are discussed in the manual document for routine **MSDST**.

A dissimilarity is a subject's estimate of the difference ("distance") between two objects. From the observed dissimilarities, a predicted distance between the objects is obtained by estimating the location of the objects in a Euclidean space of given dimension. In metric scaling, the dissimilarity may be a ratio measure (in which case a dissimilarity of zero means that the objects are in the same location) or an interval measure (in which case "distance" plus a constant is observed). When an interval measure is observed, the interval constant, c, must also be estimated in order to relate the dissimilarity to the predicted distance. For ratio measures, c is not required. A couple of methods for estimating c are used by the routines in this chapter. These methods are explained in the routines that use them.

In nonmetric scaling, the dissimilarity is an ordinal (rank) or categorical measure. In this case, the stress function need only assure that the predicted distances satisfy, as closely as possible, the ordinal or categorical relationships observed in the data. Thus, the stress should be zero if the predicted distances maintain the observed rankings in the dissimilarities in ordinal data. The meaning of a stress in categorical data is more obtuse and is discussed further below.

In ordinal data, the stress function is computed as follows: First, the dissimilarities are transformed so that they correspond as closely as possible to the predicted distances, but such that the observed ordinal relationships are maintained. The transformed dissimilarities are called "disparities", and the stress function is computed from the disparities and the predicted distances. (In ratio and interval data, disparities may be taken as the dissimilarities.) Thus, if the predicted distances preserve the observed ordinal relationships, a stress of zero will be computed. If the predicted distances do not preserve these relationships, then new estimates for the distances based upon the disparities can be computed. These can be followed by new estimates of the disparities. When the new estimates do not lead to a lower stress, convergence of the algorithm is assumed.

In categorical data, all that is observed is a category for the "distance" between the objects, and there are no known relationships between the categories. In categorical data, the disparities are such that the categories are preserved. A score minimizing the stress is found for each category. As with ordinal data, new distances are computed from this score, followed by new scores for the categories, etc., with convergence occurring when the stress cannot be lowered further. In categorical data, a stress of zero should be relatively uncommon.

The individual differences model assumes that the squared distance between stimuli i and j for subject l,

$$\delta_{ijl}^2$$

is given as

$$\delta_{ijl}^2 = \sum_{k=1}^{d} w_{lk}\left(\lambda_{ik} - \lambda_{jk}\right)^2$$

where d is the number of dimensions (always assumed to be known), $\boldsymbol{\lambda}_{lk}$ is the location of the i-th stimulus in the k-th dimension, and w_{lk} is the weight given by subject l to the k-th dimension. Let

$$\overline{\delta}_{i\bullet l}^2$$

denote the average of the squared distances in the i-th row of the dissimilarity matrix for the l-th subject, let

$$\overline{\delta}_{\bullet jl}^2$$

be similarly defined for the j-th column, and let

$$\overline{\delta}_{\bullet\bullet l}^2$$

denote the average of all squared distances for the l-th subject. Then, the product moment (double centering) transformation is given by

$$p_{ijl} = -\left(\delta_{ijl}^2 - \overline{\delta}_{i\bullet l}^2 - \overline{\delta}_{\bullet jl}^2 + \overline{\delta}_{\bullet\bullet l}^2\right)2.0$$

The advantage of the product-moment transformations is that the "product-moment" (double centered) matrices $P_l = (p_{ijl})$ can be expressed as

$$P_l = \Lambda[\mathrm{diag}(W_l)]\Lambda^{\mathrm{T}}$$

where $\Lambda = (\boldsymbol{\lambda}_{ik})$ is the configuration matrix, and where $\mathrm{diag}(W_l)$ is a diagonal matrix with the subject weights for subject l, w_{lk}, along the diagonal. If one assumes that the dissimilarities are measured without error, then the dissimilarities can be used in place of the distances, and the above relationship allows one to compute both $\mathrm{diag}(W_l)$ and Λ directly from the product-moment matrices so obtained. If error is present but small, then very good estimates of Λ and $\mathrm{diag}(W_l)$ can still be obtained (see De Leeuw and Pruzansky 1978). Routine `MSDBL` computes the product-moment matrices while `MSINI` computes the above estimates for X and $\mathrm{diag}(W_l)$.

Data Structures

The data input to a multidimensional scaling routine is, conceptually, one or more dissimilarity (or similarity) matrices where a dissimilarity matrix contains the dissimilarity measure between the *i*-th and *j*-th stimuli (objects) in position (*i*, *j*) of the matrix. In multidimensional scaling, the dissimilarity matrix need not be symmetric (asymmetric distances can also be modelled, see routine `MSDST`) but if it is, only elements above the diagonal need to be observed. Moreover, in the multidimensional "unfolding" models, the distances between all pairs of objects are not observed. Rather, all (or at least many) of the dissimilarities between one set of objects and a second set are measured. When these types of input are combined with the fact that missing values are also allowed in many multidimensional scaling routines, it is easy to see that data structures required in multidimensional scaling can be quite complicated. Three types of structures are allowed for the routines described in this chapter. These are discussed below.

Let **X** denote a matrix containing the input dissimilarities. The columns of **X** correspond to the different subjects, and a subjects dissimilarity matrix is contained within the column. Thus, **X** is a matrix containing a set of dissimilarity matrices, one dissimilarity matrix within each column. For any one problem, the form (structure) of all dissimilarity matrices input in **X** must be consistent over all subjects. The form can vary from problem to problem, however. In the following, **X** contains only one column and the index for subject is ignored to simplify the notation. The three storage forms used by the routines described in this chapter are

1. **Square symmetric**: For this form, each column of **X** contains the upper triangular part of the dissimilarity matrix, excluding the diagonal elements (which should be zero anyway). Specifically, **X**(1) contains the (1, 2) element of the dissimilarity matrix, **X**(2) contains the (1, 3) element, **X**(3) contains the (2, 3) element, etc. Let q denote the number of stimuli in the matrix. All $q(q - 1)/2$ off-diagonal elements are stored.

2. **Square asymmetric**: **X** contains all elements of each square matrix, including the diagonal elements, which are not used. The dissimilarities are stored in **X** as if **X** were dimensioned $q \times q$. The diagonal elements are ignored.

3. **Rectangular**: This corresponds to the "unfolding models" in which not all of the dissimilarities in each matrix are observed. In this storage mode, the row stimuli do not correspond to the column stimuli. Because of the form of the data, no diagonal elements are present, and the data are stored in **X** as if **X** were dimensioned $r \times s$ where r is the number of row stimuli and s is the number of column stimuli.

Missing values are also allowed. They are indicated in **X** in either of two ways: 1) The standard IMSL missing value indicator NaN (not a number) may be used to indicate missing values, or 2) negative elements of **X** are taken to be missing dissimilarities.

Table 14.1 gives some notation commonly used in this chapter. In general, an element of a matrix is denoted by the lowercase matrix name with subscripts. The notation is generally consistent, but there are some variations when variation seems appropriate.

Table 2 – Commonly Used Notation

Symbol	Fortran	Meaning
δ_{ijl}	DIST	Distance between objects *i* and *j* for subject *l*.
δ^*_{ijl}	DISP	Disparity for objects *i* and *j* for subject *l*.
X	X	The input array of dissimilarities.
D	NDIM	The number of dimensions in the solution.
W	W	The matrix of subject weights.
diag(W_l)		The diagonal matrix of subject weights for subject *l*.
π	WS	The matrix of stimulus weights.
Λ	CFL	The configuration matrix.
α_h	A	The intercept for strata *h*.
β_h	B	The slope for strata *h*.
ν_h	WT	The stratum weight for stratum *h*.
N_h	NCOM	The number nonmissing dissimilarities in stratum *h*.
P_l	P	The product-moment matrix for subject *l*.
ϕ	STRSS	The stress criterion (over all strata).
ϕ_l	STRS	The stress within stratum *l*.
P	POWER	The power to use in the stress criterion.
Q	NSTIM	The total number of stimuli.
η	NSUB	The number of matrices input.
Γ		Normalized eigenvectors.
	IFORM	Option giving the form of the dissimilarity input.
	ICNVT	Option giving the method for converting to dissimilarities.
	MODEL	Vector giving the parameters in the distance model.
	ISTRS	Option giving the stress formula to use.
	ITRANS	Option giving the transformation to use.
	IDISP	The method to be used in estimating disparities.
	EPS	Convergence tolerance.

MSIDV

more...

Performs individual-differences multidimensional scaling for metric data using alternating least squares.

Required Arguments

NSTIM — Number of stimuli in each similarity/dissimilarity matrix. (Input)

X — NSUB similarity or dissimilarity matrices in symmetric storage mode. (Input)
Each matrix must occupy consecutive memory positions, and must be stored as a column in X. X must be dimensioned as

```
DIMENSION X(NC2,NSUB)
```

where NC2 = NSTIM * (NSTIM - 1)/2. Each matrix is stored without the diagonal elements by column as upper triangular matrices. For example, a 3 by 3 matrix would be stored with the (1, 2), (1, 3), (2, 3) elements as the first three elements of the first column of X.

NDIM — Number of dimensions desired in the solution. (Input)

DIST — Vector of length NSUB * NC2, where NC2 = NSTIM * (NSTIM - 1)/2, containing the predicted distances. (Output)
DIST contains the distances as predicted by the estimated parameters in the model. DIST has the same storage mode as X and may be treated as a series of NSUB matrices in symmetric storage mode but without the diagonal elements.

CFL — Matrix of size NSTIM by NDIM containing the configuration of points obtained from the multidimensional scaling. (Output)

A — Vector of length NSUB containing the intercepts for each subject. (Output)

B — Vector of length NSUB containing the slopes for each subject. (Output)

WT — Vector of length NSUB containing the criterion function weights for each subject. (Output)

STRS — Vector of length NSUB containing the value of the weighted optimized criterion within each subject. (Output)

STRSS — Value of the weighted optimized criterion function (summed over subjects). (Output)

RESID — NSUB * NC2 vector containing the observation residuals. (Output)
Here, NC2 = NSTIM (NSTIM - 1)/2.

Optional Arguments

NSUB — Number of matrices to be used in the analysis. (Input)
Default: NSUB = size (X,2).

ICNVT — Option for converting from similarity to dissimilarity data. (Input)
If ICNVT = 0, the input data contains dissimilarities and no conversion is performed. If ICNVT = 1, the data are converted from similarity to dissimilarity data by subtracting each similarity from the largest similarity for the subject. If ICNVT = 2, the data are converted to dissimilarities by reciprocating each similarity.
Default: ICNVT = 0.

MODEL — Model option parameter. (Input)
MODEL = 0 means the Euclidean model is used, otherwise, the individual differences model is used.
Default: MODEL = 0.

ISTRS — Option giving the stress formula to be used. (Input)
Default: ISTRS = 0.
Stress formulas differ in the weighting given to each subject. The valid values of ISTRS are:

ISTRS	Weighting
0	Inverse of within-subject variance of observed dissimilarities about the predicted distances
1	Inverse of within-subject sum of squared dissimilarities
2	Inverse of within-subject variance of dissimilarities about the subject mean

See the Description section for further discussion of the stress formula weights.

ITRANS — Option giving the transformation to be used on the observed and predicted dissimilarities when computing the criterion function. (Input)
Default: ITRANS = 0.

ITRANS	Transformation
0	Squared distances
1	Distances (that is, no transformation is performed)
2	Log of the distances

See the Description section for further discussion of stress formula transformations.

IPRINT — Printing option. (Input)
Default: IPRINT = 0.

IPRINT	Action
0	No printing is performed.
1	Printing is performed, but the output is abbreviated.
2	All printing is performed.

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

W — NSUB by NDIM matrix containing the subject weights. (Output when MODEL is not zero, not referenced otherwise)
W is not used and may be a 1x1 array if MODEL = 0.

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW = size (W,1).

FORTRAN 90 Interface

Generic: CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, STRSS, RESID [, ...])

Specific: The specific interface names are S_MSIDV and D_MSIDV.

FORTRAN 77 Interface

Single: CALL MSIDV (NSTIM, NSUB, X, ICNVT, MODEL, ISTRS, ITRANS, NDIM, IPRINT, DIST, CFL, LDCFL, W, LDW, A, B, WT, STRS, STRSS, RESID)

Double: The double precision name is DMSIDV.

Description

Routine **MSIDV** performs multidimensional scaling analysis according to an alternating optimization algorithm. Input to **MSIDV** consists of symmetric dissimilarity matrices measuring distances between the row and column objects. Optionally, similarities can be input, and these can be converted to dissimilarities by use of the **ICNVT** option. In **MSIDV**, the row and column objects (stimuli) must be identical. Dissimilarities in multidimensional scaling are used to position the objects within a d = **NDIM** dimensional space, where d is specified by the user. Optionally, in the individual differences scaling model (**MODEL** ≠ 0), the weight assigned to each dimension for each subject may be changed.

The Input Data

The data input in **X** must be in a special symmetric storage form. For this storage mode, the input array **X** contains only the upper triangular part of each dissimilarity matrix and does not contain the diagonal elements (which should all be zero anyway). Storage of symmetric data in **X** is as follows: **X**(1) corresponds to the (1, 2) element in the first matrix (which is a measure of the distance between objects 1 and 2), **X**(2) corresponds to the (1, 3) element, **X**(3) corresponds to the (2, 3) element, etc., until all $t = q(q - 1)/2$ off-diagonal elements in the first matrix are stored, where q = **NSTIM**. The $t + 1$ element in **X** contains the (1, 2) element in the second matrix, and so on.

Missing values are indicated in either of two ways: 1) The standard missing value indicator NaN (not a number), specified via routine **AMACH**(6) (Reference Material) may be used to indicate missing values, or 2) Negative elements of **X** may be used to indicate missing observations. In either case, missing values are estimated as the mean dissimilarity for the subject and used as such when computing initial estimates, and they are omitted from the criterion function when optimal estimates are computed.

Routine **MSIDV** assumes a metric scaling model. When no transformation is specified (**ITRANS** = 1), then each datum (after transforming to dissimilarities) is a measure of distance plus a constant, α_m. In this case, the constant (which is always called the "intercept") is assumed to vary with subject and must first be added to the observed dissimilarities in order to obtain a metric. When a transformation is specified (**ITRANS** ≠ 1), the meaning of α_m changes (with respect to metrics). Thus, when **ITRANS** = 1, the data is assumed to be interval (see the chapter introduction) while when **ITRANS** ≠ 1 ratio data is assumed. A scaling factor, the "slope", is also always estimated for each subject.

The Criterion Function

When **ISTRS** = 1 or 2, the criterion function in **MSIDV** is given as

$$\phi = \sum_m v_m \sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right) \right)^2$$

where δ_{ijm} denotes the predicted distance between objects *i* and *j* on subject *m*,

$$\delta^*_{ijm}$$

denotes the corresponding dissimilarity (the observed distance), v_m is the subject weight, *f* is one of the transformations $f(x) = x^2$, $f(x) = x$, or $f(x) = \ln(x)$ specified by parameter **ITRANS**, α_m is the intercept added to the transformed observation within each subject, and β_m is the slope for the subject. For **ISTRS** = 0, the criterion function is given as

$$\phi = \sum_m n_m \ln\left(\sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right) \right)^2 \right)$$

where n_m is the number of nonmissing observations on the *m*-th subject. Assuming fixed weights, the first derivatives of the criterion for **ISTRS** = 0 are identical to the first derivatives of the criterion when **ISTRS** = 1 or 2, but with weights

$$v_m^{-1} = \sum_{i,j} \left(f\left(\delta^*_{ijm}\right) - \alpha_m - \beta_m f\left(\delta_{ijm}\right) \right)^2 / n_m$$

ISTRS can, thus, be thought of as changing the weighting to be used in the criterion function.

The transformation *f*(*x*) specified by parameter **ITRANS** is used to obtain constant within-subject variance of the subject dissimilarities. If the variance of the log of the observed dissimilarities (about the predicted dissimilarities) is constant within subject, then the log transformation should be used. In this case, the variance of a dissimilarity should be proportional to its magnitude. Alternatively, the within-subject variance may be constant when distances (or squared distances) are used.

The Distance Models

The distance models for δ_{ijm} available in **MSIDV** are given by:

- The Euclidian Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

- The individual-differences model:

$$\delta_{ijm}^2 = \sum_{k=1}^{d} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

where Λ denotes the configuration (**CFL**) so that λ_{ik} is the location of the *i*-th stimulus in the *k*-th dimension, where *d* is the number of dimensions, and where w_{mk} is the weight assigned by the *m*-th subject to the *k*-th dimension (**W**).

The Subject Weights

Weights that are inversely proportional to the estimated variance of the dissimilarities (about their predicted values) within each subject may be preferred because such weights lead to normal distribution theory maximum likelihood estimates (when it is assumed that the dissimilarities are independently normally distributed with constant residual variance). The estimated (conditional) variance used as the inverse of the weight ν_m for the *m*-th subject in **MSIDV** (when **ISTRS** = 0) is computed as

$$v_m^{-1} = \sum_{i,j} \frac{\left(f\left(\delta_{ijm}^* \right) - \alpha_m - \beta_m f\left(\delta_{ijm} \right) \right)^2}{n_m}$$

where the sum is over the observations for the subject, and where n_m is the number of observed nonmissing dissimilarities for the subject. These weights are used in the first derivatives of the criterion function.

When **ISTRS** = 1, the within-subject average sum of squared dissimilarities are used for the weights. They are computed as

$$v_m^{-1} = \frac{\sum_{i,j} f\left(\delta_{ijm}^* \right)^2}{n_m}$$

Finally, when **ISTRS** = 2, the within-subject variance of the dissimilarities is used for the weights. These are computed as follows

$$v_m^{-1} = \frac{\sum_{i,j} \left(f(\delta_{ijm}^*) - \overline{f(\delta_{ijk}^*)} \right)^2}{n_m}$$

where

$$\overline{f\left(\delta^*_{ijm}\right)}$$

denotes the average of the transformed dissimilarities in the stratum.

The Optimization Procedure

Initial estimates of all parameters are obtained through methods discussed in routine **MSINI**. After obtaining initial estimates, a modified Gauss-Newton algorithm is used to obtain estimates for the parameters that optimize the criterion function. The parameters are optimized sequentially as follows:

1. Optimize the configuration estimates, Λ = **CFL**.
2. If required, estimate the optimal subject weights, w_{mk} = **W**(**m**, **k**), one subject at a time.
3. Optimize the parameters α_m = A(**m**) and β_m = **B**(**m**), one subject at a time.
4. If convergence has not been reached, continue at Step 1.

An iteration is defined to be all of the Steps 1, 2, and 3. Convergence is assumed when the maximum absolute change in any parameter during an iteration is less than 10^{-4} or if there is no change in the criterion function during an iteration.

The L_p Gauss-Newton Description

A modified Gauss-Newton algorithm is used in the estimation of all parameters. This algorithm, which is discussed in detail by Merle and Spath (1974), uses iteratively reweighted least squares on a Taylor series linearization of the parameters in δ_{ijm}. During each iteration, the subject weights, which may depend upon the parameters in the model, are assumed to be fixed.

Standardization

All models available are overparameterized so the resulting parameter estimates are not uniquely defined. For example, in the Euclidean model, the columns of *X* can be translated or "rotated" (multiplied by an orthonormal matrix), and the resulting stress will not be changed. To eliminate lack of uniqueness due to translation, model estimates for the configuration are centered in all models. No attempt at eliminating the rotation problem is made, but note that rotation invariance is not a problem in many of the models given. With more general models than the Euclidean model, other kinds of overparameterization occur. Further restrictions on the parameters to eliminate this overparameterization are given below by the model transformation type specified by **ITRANS**. In the following, $w_{lk} \in W$, where *W* is the matrix of subject weights. The restrictions to be applied by model transformation type are

1. For all models:

   (*a*) $$\sum_{i=1}^{q} x_{ik} = 0$$

   where q = NSTIM. i.e., center the columns of X.

   (*b*) If W is in the model, scale the columns of W so that

   $$\sum_{i=1}^{q} x_{ik}^2 = 1$$

2. For $f(x) = x$ and $f(x) = x^2$:

   (*a*) Set $b_h = 1$ if the data are matrix conditional and W is in the model or if the data are unconditional. (Matrix conditional with one matrix is considered to be unconditional data.)

   (*b*) If W is not in the model, scale all elements in X so that

   $$\sum_{h=1}^{\eta} b_h^2 = \eta$$

   where η = NSUB is the number of matrices observed.

3. For $f(x) = \ln(x)$, substitute a_h for b_h (but set a_h to 0 instead of 1) in all restrictions in Item 2.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2IDV/DM2IDV**. The reference is:

   ```
   CALL M2IDV (NSTIM, NSUB, X, ICNVT, MODEL, ISTRS, ITRANS, NDIM, IPRINT, DIST, CFL,
      LDCFL, W, LDW, A, B, WT, STRS, STRSS, RESID, WK1, WK2, WK3, WK4, WK5, WK6, WK7,
      IWK8, WK10, WK11, WK12, WK13, ID, WKDER, DWKHES, DWKGRA, WKDDP, NCOM, DISP)
   ```

   The additional arguments are as follows:

   WK1 — Work vector of length equal to max(NSUB, NDIM * NSTIM, ND + 1)

   WK2 — Work vector of length equal to NDIM * NDIM

   WK3 — Work vector of length equal to NSTIM * NSTIM

   WK4 — Work vector of length equal to NSTIM * NSTIM

   WK5 — Work vector of length equal to NDSS * NDSS

   WK6 — Work vector of length equal to 3 * NDSS

   WK7 — Work vector of length equal to 5 * NDSS

   IWK8 — Integer work vector of length equal to NDSS

   WK10 — Work vector of length equal to NDIM * NDIM

WK11 — Work vector of length equal to NSUB * NSUB

WK12 — Work vector of length equal to NDIM * NDIM * max(NSUB, NSTIM)

WK13 — Work vector of length equal to NSTIM * NDIM

ID — Integer work vector of length equal to 4 * NDIM + 2

WKDER — Work vector of length equal to NPAR

DWKHES — Double precision work vector of length equal to NDIM * NDIM * NSTIM * NSTIM

DWKGRA — Double precision work vector of length equal to NPAR

WKDDP — Work vector of length equal to NC2

NCOM — Work vector of length equal to NSUB

DISP — Work vector of length equal to NSUB * NC2

where ND = NDIM * (NDIM + 1)/2, NC2 = NSTIM * (NSTIM - 1)/2, NDSS = max(NDIM, NSTIM, NSUB), and where NPAR = NDIM * NSTIM + 2 * NSUB when MODEL = 0; otherwise NPAR = NDIM * NSTIM + (NDIM + 2) * NSUB.

2. Informational errors

Type	Code	Description
3	1	At some point during the iterations there were too many step halvings. This is usually not serious.
4	1	The program cannot continue because a Hessian matrix is ill defined. A different model may be required. This error should only occur when there is serious numerical imprecision.
4	2	A dissimilarity matrix has every element missing.

Examples

Example 1

The following example concerns some intercity distance rankings. The data are described by Young and Lewyckyj (1979, page 83). The driving mileages between various cities in the United States are ranked, yielding a symmetric ordinal dissimilarity matrix. These rankings are used as input to MSIDV. A Euclidean model is fit. The resulting two-dimensional scaling yields results closely resembling the locations of the major cites in the U.S. Note that MSIDV assumes continuous, not ranked, data.

The original rankings are given as:

$$\begin{pmatrix} - & 4 & 22 & 8 & 34 & 6 & 10 & 35 & 36 & 3 \\ & - & 13 & 15 & 31 & 21 & 9 & 32 & 30 & 5 \\ & & - & 12 & 11 & 29 & 27 & 16 & 19 & 26 \\ & & & - & 24 & 18 & 25 & 28 & 33 & 23 \\ & & & & - & 39 & 42 & 2 & 17 & 37 \\ & & & & & - & 20 & 44 & 45 & 14 \\ & & & & & & - & 43 & 40 & 1 \\ & & & & & & & - & 7 & 41 \\ & & & & & & & & - & 38 \\ & & & & & & & & & - \end{pmatrix}$$

```
      USE PGOPT_INT
      USE MSIDV_INT

      IMPLICIT   NONE

      INTEGER    IPRINT, ISTRS, LDCFL, LDW, LNX, NDIM, NSTIM, NSUB, IPAGE
      PARAMETER  (IPRINT=2, ISTRS=1, LDCFL=10, LNX=45, NDIM=2, &
                  NSTIM=10, NSUB=1)
!
      REAL       A(1), B(1), CFL(LDCFL,NDIM), DIST(45), RESID(LNX), &
                 STRS(1), STRSS, WT(1), X(45,1)
!
      DATA X/4, 22, 13, 8, 15, 12, 34, 31, 11, 24, 6, 21, 29, 18, 39, &
          10, 9, 27, 25, 42, 20, 35, 32, 16, 28, 2, 44, 43, 36, 30, &
          19, 33, 17, 45, 40, 7, 3, 5, 26, 23, 37, 14, 1, 41, 38/
!                                Call PGOPT to set page length for
!                                the plotting
      IPAGE =50

      CALL PGOPT (-2, IPAGE)
!
      CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, &
                 STRSS, RESID, ISTRS=ISTRS, IPRINT=IPRINT)
!
      END
```

Output

```
 Initial parameter estimates.
        CFL
          1       2
 1  -0.762   0.124
 2  -0.451  -0.349
 3   0.496   0.073
 4  -0.151   0.651
 5   1.237   0.392
 6  -1.114   0.588
 7  -1.077  -0.566
 8   1.461   0.034
 9   1.321  -0.614
10  -0.961  -0.333

                       Iteration history
```

```
 Iter      Source        Stress   Stress change  Maximum Change
  1     INIT   STRSS  0.3755E-02
  1     CONFIG STRSS  0.3399E-02    0.3559E-03      0.8062E-03
  1     LINES  STRSS  0.3142E-02    0.2564E-03      0.8062E-03
  2     CONFIG STRSS  0.3068E-02    0.7382E-04      0.1022E-04
  2     LINES  STRSS  0.3047E-02    0.2156E-04      0.1022E-04

                      Plot(s) of the configuration matrix (CFL)
             :::::::+:::::::::::::X::+:::::::::::::::+:::::::::::::::+
             .                      I                               .
       0.600 X                      I                               +
             .                      I                               .
             .                      I                               .
             .                      I                               .
       0.450 +                      I                               +
             .                      I                         X     .
             .                      I                               .
             .                      I                               .
       0.300 +                      I                               +
             .                      I                               .
             .                      I                               .
 D           .                      I                               .
 i     0.150 +                      I                               +
 m           .       X              I                               .
 e           .                      I         X                     .
 n           .                      I                           X   .
 s     0.000 +------------------------------------------------------+
 i           .                      I                               .
 o           .                      I                               .
 n           .                      I                               .
      -0.150 +                      I                               +
 2           .                      I                               .
             .                      I                               .
             .                      I                               .
      -0.300 +                      I                               +
             .   X          X       I                               .
             .                      I                               .
             .                      I                               .
      -0.450 +                      I                               +
             .                      I                               .
             .                      I                               .
             . X                    I                               .
      -0.600 +                      I                               +
             :::::::+:::::::::::::::+:::::::::::::::+:::::::::::X::::+
                 -0.80           0.00            0.80            1.60

                                 Dimension 1

 Final parameter estimates.

  NCOM
    45

        CFL
         1       2
   1  -0.738   0.095
   2  -0.447  -0.337
   3   0.497   0.077
   4  -0.153   0.661
   5   1.237   0.399
```

```
      6  -1.132   0.609
      7  -1.074  -0.571
      8   1.445   0.035
      9   1.325  -0.624
     10  -0.960  -0.343

         A
    -0.04255

         B
    0.4019

         WT
    0.01248

         STRS
    0.003047

   STRESS =     3.04681E-03

                    Residuals
    Subject  Row Stimulus  Column Stimulus Residual
        1           2               1       -0.0436
        1           3               1        0.1230
        1           3               2       -0.1422
        1           4               1       -0.1318
        1           4               2       -.0697
        1           4               3       -0.0581
        1           5               1        0.0950
        1           5               2        0.0631
        1           5               3       -0.0456
        1           5               4        0.0639
        1           6               1       -0.0742
        1           6               2        0.1268
        1           6               3        0.0681
        1           6               4        0.1212
        1           6               5       -0.0495
        1           7               1       -0.0376
        1           7               2       -0.0216
        1           7               3       -0.0736
        1           7               4       -0.0119
        1           7               5        0.0464
        1           7               6        0.0558
        1           8               1       -0.1177
        1           8               2        0.0169
        1           8               3        0.0480
        1           8               4       -0.0173
        1           8               5       -0.0223
        1           8               6        0.0178
        1           8               7       -0.0047
        1           9               1       -0.0185
        1           9               2        0.0373
        1           9               3        0.0872
        1           9               4        0.0618
        1           9               5        0.0335
        1           9               6       -0.0913
        1           9               7        0.0202
        1           9               8       -0.0671
        1          10               1       -0.0415
        1          10               2       -0.0276
```

```
         1          10               3        0.0869
         1          10               4        0.1342
         1          10               5       -0.1565
         1          10               6       -0.0522
         1          10               7        0.0179
         1          10               8        0.0701
         1          10               9       -0.0191

                          Residual Plot

          0.160 :::::::+:::::::::::::::+:::::::::::::::+:::::::::::::::+::::
                .                                                         .
                .                                                         .
                .           X                                             .
                .         X                                               .
          0.120 +       X   X                                             +
                .                                                         .
                .                                                         .
                .                              X                          .
                .       X         X                                       .
          0.080 +                                                         +
                .                     X                       X           .
                .               X           X  X                          .
                .         X                                               .
                .     X                                       X           .
          0.040 +                       X                                 +
      R         .      X                                                  .
      e         .                                        X                 .
      s        X                            X                       X     .
      i         .                                                         .
      d   0.000 +-------------------------------------------------------+
      u         .                 X                                 X     .
      a         .                       X            X   X               .
      l        .X X                                                       .
      s         .                                                         .
         -0.040 +XX X                                                     +
                .   X                                    X                .
                .    X X                                                  .
                .  X                                                      .
                .  X    X            X                                    .
         -0.080 +                                                         +
                .                                                        X
                .                                                         .
                .                                                         .
                .                                                         .
         -0.120 +                                 X                       +
                .   X                                                     .
                .                                                         .
                .      X                                                  .
                .                                                         .
         -0.160 :::::::+:::::::::::::::+:::::::::::::::+::X:::::::::::::+::::
                       1               3               5               7

                                Predicted Distances
```

Example 2

The second example involves three subjects' assessment of the dissimilarity between rectangles that vary in height and width. An analysis is performed in $k = 2$ dimensions using the individual-differences scaling model. The estimated subject weights, w_{mk}, indicate how each subject weight the dimensions. The raw data are given as follows:

$$\begin{pmatrix} - & 1.00 & 1.41 & 2.24 & 2.00 & 2.24 & 1.41 & 1.00 & 1.00 \\ & - & 1.00 & 2.00 & 2.24 & 2.83 & 2.24 & 2.00 & 1.41 \\ & & - & 1.00 & 1.41 & 2.24 & 2.00 & 2.24 & 1.00 \\ & & & - & 1.00 & 2.00 & 2.24 & 2.83 & 1.41 \\ & & & & - & 1.00 & 1.41 & 2.24 & 1.00 \\ & & & & & - & 1.00 & 2.00 & 1.41 \\ & & & & & & - & 1.00 & 1.00 \\ & & & & & & & - & 1.41 \\ & & & & & & & & - \end{pmatrix}$$

$$\begin{pmatrix} - & 1.50 & 1.68 & 2.12 & 1.50 & 2.12 & 1.68 & 1.50 & 0.75 \\ & - & 0.75 & 1.50 & 2.12 & 3.35 & 3.09 & 3.00 & 1.68 \\ & & - & 0.75 & 1.68 & 3.09 & 3.00 & 3.09 & 1.50 \\ & & & - & 1.50 & 3.00 & 3.09 & 3.35 & 1.68 \\ & & & & - & 1.50 & 1.68 & 2.12 & 0.75 \\ & & & & & - & 0.75 & 1.50 & 1.68 \\ & & & & & & - & 0.75 & 1.50 \\ & & & & & & & - & 1.68 \\ & & & & & & & & - \end{pmatrix}$$

$$\begin{pmatrix} - & 0.50 & 2.06 & 4.03 & 4.00 & 4.03 & 2.06 & 0.50 & 2.00 \\ & - & 2.00 & 4.00 & 4.03 & 4.12 & 2.24 & 1.00 & 2.06 \\ & & - & 2.00 & 2.06 & 2.24 & 1.00 & 2.24 & 0.50 \\ & & & - & 0.50 & 1.00 & 2.24 & 4.12 & 2.06 \\ & & & & - & 0.50 & 2.06 & 4.03 & 2.00 \\ & & & & & - & 2.00 & 4.00 & 2.06 \\ & & & & & & - & 2.00 & 0.50 \\ & & & & & & & - & 2.06 \\ & & & & & & & & - \end{pmatrix}$$

```
      USE MSIDV_INT

      IMPLICIT   NONE
      INTEGER    ICNVT, IPRINT, ISTRS, ITRANS, LDCFL, LDW, LNX, MODEL, &
                 NDIM, NSTIM, NSUB
      PARAMETER  (IPRINT=1, LDCFL=9, LDW=3, LNX=108, MODEL=1, NDIM=2, &
                  NSTIM=9, NSUB=3)
!
      REAL       A(NSUB), B(NSUB), CFL(LDCFL,NDIM), DIST(LNX), &
```

```
                  RESID(LNX), STRS(NSUB), STRSS, W(LDW,NDIM), WT(NSUB), &
                  X(36,NSUB)
!
      DATA X/1.00, 1.41, 1.00, 2.24, 2.00, 1.00, 2.00, 2.24, 1.41, &
          1.00, 2.24, 2.83, 2.24, 2.00, 1.00, 1.41, 2.24, 2.00, 2.24, &
          1.41, 1.00, 1.00, 2.00, 2.24, 2.83, 2.24, 2.00, 1.00, 1.00, &
          1.41, 1.00, 1.41, 1.00, 1.41, 1.00, 1.41, 1.50, 1.68, 0.75, &
          2.12, 1.50, 0.75, 1.50, 2.12, 1.68, 1.50, 2.12, 3.35, 3.09, &
          3.00, 1.50, 1.68, 3.09, 3.00, 3.09, 1.68, 0.75, 1.50, 3.00, &
          3.09, 3.35, 2.12, 1.50, 0.75, 0.75, 1.68, 1.50, 1.68, 0.75, &
          1.68, 1.50, 1.68, 0.50, 2.06, 2.00, 4.03, 4.00, 2.00, 4.00, &
          4.03, 2.06, 0.50, 4.03, 4.12, 2.24, 1.00, 0.50, 2.06, 2.24, &
          1.00, 2.24, 2.06, 2.00, 0.50, 1.00, 2.24, 4.12, 4.03, 4.00, &
          2.00, 2.00, 2.06, 0.50, 2.06, 2.00, 2.06, 0.50, 2.06/
!
      CALL MSIDV (NSTIM, X, NDIM, DIST, CFL, A, B, WT, STRS, &
                  STRSS, RESID, MODEL=MODEL, IPRINT=IPRINT, W=W)
!
      END
```

Output

```
                  Iteration history
Iter      Source        Stress   Stress change  Maximum Change
  1     INIT   STRSS -0.3590E+03
  1     CONFIG STRSS -0.3590E+03    0.0000E+00      0.5708E-03
  1     SUB WT STRSS -0.3590E+03    0.0000E+00      0.1581E-02
  1     LINES  STRSS -0.3590E+03    0.0000E+00      0.2727E-02
  2     CONFIG STRSS -0.3590E+03    0.0000E+00      0.1442E-06
  2     SUB WT STRSS -0.3590E+03    0.0000E+00      0.7165E-04
  2     LINES  STRSS -0.3590E+03    0.0000E+00      0.7165E-04
Final parameter estimates.

     NCOM
  1    2    3
 36   36   36

 CFL
 1       2
 1   1.225   0.000
 2   1.225  -1.225
 3   0.000  -1.225
 4  -1.225  -1.225
 5  -1.225   0.000
 6  -1.225   1.225
 7   0.000   1.225
 8   1.225   1.225
 9   0.000   0.000

         W
         1       2
 1   1.000   1.000
 2   0.342   1.372
 3   1.411   0.089

                A
         1          2          3
 -0.002773   0.001941   0.000055
```

```
                B
          1         2         3
     0.2229    0.2587    0.2963

               WT
          1         2         3
     1000.0    1000.0    1000.0

             STRS
          1         2         3
     -119.7    -119.7    -119.7

    STRESS =     -359.018
```

MSDST

Computes distances in a multidimensional scaling model.

Required Arguments

CFL — NSTIM by NDIM matrix containing the stimulus configuration. (Input)

NSUB — Number of subjects. (Input)

DIST — Vector of length nv * NSUB, where nv = NSTIM * (NSTIM - 1)/2 if IFORM = 0, and nv = NSTIM * NSTIM otherwise. (Output)
DIST may be treated as NSUB distance matrices. Storage in DIST is such that the elements of each column of a subject's distance matrix are adjacent. Each column in the matrix is immediately followed by the elements in the next column. If IFORM = 0, then only the elements in each column above the diagonal are stored. Otherwise, all elements are stored.

Optional Arguments

NSTIM — Number of stimuli. (Input)
Default: NSTIM = size (CFL,1).

NDIM — Number of dimensions in the model. (Input)
Default: NDIM = size (CFL,2).

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

IMOD — Vector of length 3 describing the weighting to be used. (Input)
Default: IMOD = 0.

IMOD	Weight
1	Not used. Reserved for other scaling subroutines.
2	Subject weights (in W).
3	Stimulus weights (in WS).

If IMOD(*i*) is zero, then the *i*-th set of weights is not used. Otherwise, the weights are used. For the Euclidean model, set IMOD(2) = IMOD(3) = 0. For the individual differences model, IMOD(2) should not be zero. For the stimulus weighted individual differences model, both IMOD(2) and IMOD(3) are not zero.

IFORM — Form option. (Input)
If IFORM = 0, the computed distances are stored as the upper triangle of square matrices stored columnwise without the diagonal elements. Otherwise, the distances are stored as square matrices and include the diagonal elements. See argument DIST.
Default: IFORM = 0.

ITRANS — Transformation option. (Input)
ITRANS determines the output returned in DIST.
Default: ITRANS = 0.

ITRANS	Output in DIST
0	Squared distances
1	Distances
2	Log of the distances

W — NSUB by NDIM matrix of individual weights. (Input)
If IMOD(2) is zero, then W is not referenced and can be a 1 × 1 array.
Default: W is a 1x1 array and not used.

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW =size(W,1).

WS — NSTIM by NDIM matrix of stimulus weights. (Input)
If IMOD(3) is zero, then W is not referenced and can be a 1 × 1 array.
Default: WS is a 1x1 array and not used.

LDWS — Leading dimension of WS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDWS =size(WS,1)

FORTRAN 90 Interface

Generic: CALL MSDST (CFL, NSUB, DIST [, ...])

Specific: The specific interface names are S_MSDST and D_MSDST.

FORTRAN 77 Interface

Single: CALL MSDST (NSTIM, NDIM, CFL, LDCFL, NSUB, IMOD, IFORM, ITRANS, W, LDW, WS, LDWS, DIST)

Double: The double precision name is DMSDST.

Description

Routine MSDST computes squared distances, distances, or log distances for various metrics in multidimensional scaling. The "distances" are computed and stored as either square matrices or as upper triangular symmetric matrices stored columnwise without the diagonal. In both cases, the distances are output in a vector of the required length. The terminology and metrics used here are the same as those used in the ALSCAL program of Takane, Young, De Leeuw (1977).

Suppose that there are q stimuli, m subjects, and d dimensions. Let $\boldsymbol{\lambda}_{ik}$ denote the location of the i-th stimulus in the k-th dimension. If w_{ik} denotes the weight of the i-th subject on the k-th dimension (matrix W) and pi_{ik} denotes the weight for the i-th stimulus on the k-th dimension (matrix WS), then the distance models computed are the same as the distance models in MSIDV. They are given by:

Euclidean Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Individual Differences Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Stimulus-Weighted Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \pi_{ik} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

Stimulus-Weighted Individual Differences Model

$$\delta_{ijm}^2 = \sum_{k=1}^{d} \pi_{ik} w_{mk} \left(\lambda_{ik} - \lambda_{jk} \right)^2$$

where δ_{ijm} is the distance between the *i*-th and *j*-th stimuli on the *m*-th subject.

Example

The following small example illustrates the distance computations in symmetric matrices. The data are fictional.

```
      USE MSDST_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    IFORM, ITRANS, LDCFL, LDW, LDWS, NDIM, NSTIM, NSUB
      PARAMETER  (LDCFL=4, LDW=2, LDWS=4, NDIM=2, NSTIM=4, NSUB=2)
!
      INTEGER    IMOD(3), NOUT
      REAL       CFL(NSTIM,NDIM), DIST(12), W(NSUB,NDIM), WS(1,1)
!
      DATA IMOD/0, 1, 0/
!
      DATA CFL/1.0, -1.0, 1.0, -1.0, &
             1.0, 1.0, -1.0, -1.0/
!
      DATA W/1.0, 2.0, 1.0, 2.0/
!
      CALL MSDST (CFL, NSUB, DIST, IMOD=IMOD, W=W)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) DIST
      END
```

Output

```
 4.00000    4.00000    8.00000    8.00000    4.00000    4.00000    8.00000
 8.00000    16.0000    16.0000    8.00000    8.00000
```

MSSTN

Transforms dissimilarity/similarity matrices and replace missing values by estimates to obtain standardized dissimilarity matrices.

Required Arguments

NROW — Number of row stimuli in each dissimilarity/similarity matrix. (Input)

NCOL — Number of column stimuli in each dissimilarity/similarity matrix. (Input)
If `IFORM` = 0 or 1, `NCOL` must equal `NROW`, and the stimuli in the rows and columns must correspond to one another.

IFORM — Storage option indicating the storage mode for the input data in each column of `X`. (Input)
Array `X` contains `NSUB` columns, and each column of `X` contains a dissimilarity/similarity matrix stored as specified by option `IFORM`.

`IFORM`	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) In this storage mode, consecutive elements of each column of `X` contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (`NROW` - 1, `NROW`) elements of the corresponding dissimilarity/similarity matrix.
1	Square matrix in full storage mode. Consecutive elements of each column of `X` contain the (1, 1), (2, 1), (3, 1), …, (`NROW`, 1), (1, 2), (2, 2), …, (`NROW`, `NROW`) elements of the corresponding dissimilarity/similarity matrix.
2	Rectangular matrix in full storage mode. In this storage mode, the row and column stimuli input in `X` do not correspond to each other. Let *m* = `NROW`. Consecutive elements of each column of `X` contain the (1, *m* + 1), (2, *m* + 1), …, (`NROW`, *m* + 1), (1, *m* + 2), …, (`NROW`, *m* + 2), …, (`NROW`, *m* + `NCOL`) elements of the corresponding dissimilarity/similarity matrix.

X — `NSUB` similarity or dissimilarity matrices in storage mode as determined by `IFORM`. (Input)
`X` must be dimensioned as:

```
DIMENSION X (LDX,NSUB)
```

where `LDX` ≥ `NROW * NCOL` in full storage mode and `LDX` ≥ `NROW *` (`NROW -` 1)/2 in symmetric storage mode. See argument `IFORM` for the method of storage used for each storage mode. Negative elements of `X`, or elements equal to NaN (“not a number”) are presumed to be missing values and will be estimated as an appropriate average in `MSSTN`.

ICNVT — Option for converting from similarity to dissimilarity matrices. (Input)

ICNVT	Conversion
0	No conversion performed.
1	Subtracting each similarity from the largest similarity in the strata (see ISTRAT).
2	Take the reciprocal of each similarity (elements of X equal to zero are assumed to be missing).

ISTRAT — Option giving the level of stratification to be used. (Input)
If ISTRAT = 1, each dissimilarity/similarity matrix in X is considered to be in a different stratum. The data are said to be matrix conditional. If ISTRAT = 2, each column of each dissimilarity matrix is considered to be in a different stratum. (Thus, each column of array X contains NCOL strata.) For ISTRAT to be 2, IFORM must be 1 or 2. The data are said to be column conditional. If ISTRAT = 3, all of the dissimilarity/similarity matrices in X are considered to be in the same stratum. The data are said to be unconditional.

NCOM — Vector containing the number of nonmissing observations in each stratum. (Output)
The diagonal elements of each dissimilarity/similarity matrix are not counted.

ISTRAT	Length of NCOM
1	NSUB
2	NSUB * NSTIM, where NSTIM = NROW when IFORM = 0 or 1, and NSTIM = NROW + NCOL when IFORM = 2
3	1

XOUT — Vector of length NV * NSUB containing the standardized dissimilarity matrices where NV = NROW * (NROW - 1)/2 if IFORM = 0 and NV = NSTIM * NSTIM otherwise. (Output)
The value of NSTIM is as described in parameter NCOM. XOUT contains the standardized dissimilarity matrices in the same storage mode as X if IFORM = 0 or 1 and stored as square matrices when IFORM = 2. Missing values are replaced by an appropriate average dissimilarity and changed in sign. Scaling is performed as requested.

Optional Arguments

NSUB — Number of dissimilarity/similarity matrices. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ISCALE — Scaling option. (Input)
Default: ISCALE = 1.

ISCALE	Scaling
0	No scaling is performed.
1	The data in each stratum are scaled such that the sum of the squared dissimilarities equals the number of elements in the stratum.

FORTRAN 90 Interface

Generic: CALL MSSTN (NROW, NCOL, IFORM, X, ICNVT, ISTRAT, NCOM, XOUT [, ...])
Specific: The specific interface names are S_MSSTN and D_MSSTN.

FORTRAN 77 Interface

Single: CALL MSSTN (NROW, NCOL, NSUB, IFORM, X, LDX, ICNVT, ISTRAT, ISCALE, NCOM, XOUT)
Double: The double precision name is DMSSTN.

Description

Routine MSSTN standardizes dissimilarity/similarity data to be usable by other routines in the multidimensional scaling chapter. Routine MSSTN converts similarity to dissimilarity data, estimates missing values within specified strata ("conditionality groups"), scales the data, computes the number of nonmissing data elements within each stratum, and stores the data in a standard form.

The computations proceed as follows:

1. Routine MSSTN begins by expanding rectangular or symmetric storage-form data into square storage mode (the form when IFORM = 1).
2. Missing values are replaced by the average nonmissing value within the stratum, or when there is only one stratum, the average within each matrix is used. If all elements in a stratum are missing and the stratum is a column of the dissimilarity/similarity matrix, then the average of the nonmissing elements in the matrix is used as the missing value estimate. (Missing values are estimated primarily for use in routines computing estimates via "double-centering", routines MSINIand MSDBL.) Missing values are denoted in the output by changing the signs of the estimated missing elements to be negative.
3. The data are converted to dissimilarity data from similarity data according to the method specified by the parameter ICNVT.

4. The data are scaled according to the method specified by the ISCALE parameter.

Comments

1. Workspace may be explicitly provided, if desired, by use of M2STN/DM2STN. The reference is:

   CALL M2STN (NROW, NCOL, IFORM, NSUB, X, LDX, ICNVT, ISTRAT, ISCALE, NCOM, XOUT, NSTIM, XX, XMIS)

   The additional arguments are as follows:

   NSTIM — Integer scalar. NSTIM = NROW when IFORM = 0 or 1, and NSTIM = NROW + NCOL when IFORM = 2.

   XX — Work vector of length NSTIM * NSTIM.

   XMIS — Work vector of length NSTIM * NSTIM.

2. Informational errors

Type	Code	Description
3	1	At least one column in column conditional data has all elements missing.
4	2	A dissimilarity matrix has every element missing.

Example

The following example illustrates the use of MSSTN on similarity data that are converted to dissimilarity data with the ICNVT = 1 option. Standardization within each matrix is used. The input data is such that IFORM = 0. Since ICNVT = 1 and all elements of the input data are nonnegative, no missing values are estimated. The input data is given by the following two similarity matrices:

$$\begin{pmatrix} - & 4 & 0 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 0 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \quad \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 0 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

```
      USE MSSTN_INT
      USE WRIRN_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    ICNVT, IFORM, ISTRAT, LDX, NCOL, NROW, NSUB
      PARAMETER  (ICNVT=1, IFORM=0, ISTRAT=1, LDX=10, NCOL=5, &
                  NROW=5, NSUB=2)
!
      INTEGER    I, J, K, N, NCOM(NSUB), NOUT
      REAL       X(LDX,NSUB), XOUT(NROW*(NROW-1))
```

```
!
      DATA X/4.0, 0.0, 1.0, 3.0, 1.0, 0.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 0.0, 3.0, 4.0/
!
      CALL MSSTN (NROW, NCOL, IFORM, X, ICNVT, ISTRAT, &
                  NCOM, XOUT)
!
      CALL WRIRN ('NCOM', NCOM, 1, NSUB, 1)
      CALL UMACH (2, NOUT)
!
      N = 1
      DO 20  I=1, 2
         WRITE (NOUT,99998) I
         DO 10  J=1, 4
            WRITE (NOUT,99999) (XOUT(K),K=N,N+J-1)
            N = N + J
   10    CONTINUE
   20 CONTINUE
!
99998 FORMAT (///' Output matrix (in XOUT)', I2)
99999 FORMAT (1X, 4F8.3)
!
      END
```

Output

```
 NCOM
  1    2
 10   10

 Output matrix (in XOUT) 1
  0.000
  1.569   1.177
  0.392   1.177   1.569
  1.177   0.392   0.784   0.000

 Output matrix (in XOUT) 2
  1.205
  0.803   1.205
  0.402   0.803   1.205
  1.205   1.606   0.402   0.000
```

MSDBL

Obtains normalized product-moment (double centered) matrices from dissimilarity matrices.

Required Arguments

NSTIM — Number of stimuli in each dissimilarity matrix. (Input)

IFORM — Storage option for the data in each dissimilarity matrix. (Input) Each column of X contains one of the NSUB dissimilarity matrices in the storage mode specified by IFORM.

IFORM	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) In this storage mode, consecutive elements of each column of X contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (NSTIM - 1, NSTIM) elements of the corresponding dissimilarity matrix.
1	Square matrix in full storage mode. Consecutive elements of each column of X contain the (1, 1), (2, 1), (3, 1), …, (NROW, 1), (1, 2), (2, 2), …, (NSTIM, NSTIM) elements of the corresponding dissimilarity matrix.

X — NV by NSUB matrix containing the NSUB dissimilarity matrices, where NV = NSTIM * (NSTIM - 1)/2 if IFORM = 0, and NV = NSTIM * NSTIM if IFORM = 1. (Input)
Missing values (NaN, "not a number") are not allowed in X, but the position of a missing element may be indicated as a negative dissimilarity. Since MSDBL uses the absolute value of each element in X in the estimation procedure, the signs of elements in X have no effect. See Comments.

DISP — NSTIM by NSTIM by NSUB array containing the NSUB dissimilarity matrices in full storage mode. (Output)
In DISP, missing value estimates are positive, and all elements represent the square of distances.

P — NSTIM by NSTIM by NSUB array containing the standardized product-moment matrices in full storage mode. (Output)
P contains NSUB matrices, each of size NSTIM by NSTIM. If DISP is not needed, DISP and P can occupy the same storage locations.

DS — NSTIM by NSTIM array containing the sum of the NSUB matrices in P. (Output)

Optional Arguments

NSUB — Number of dissimilarity matrices. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ISCALE — Scaling option. (Input)
Default: ISCALE = 1.

ISCALE	Type of Scaling
0	No scaling
1	Scaling within each matrix
2	Scaling over all matrices

Scaling is such that the Euclidean norm of the vector of scaled data is equal to the number of elements in vector.

LDDISP — Leading and second dimension of DISP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDISP = size (DISP,1).

LDP — Leading and second dimension of P exactly as specified in the dimension statement in the calling program. (Input)
Default: LDP = size (P,1).

LDDS — Leading dimension of DS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDDS = size (DS,1).

FORTRAN 90 Interface

Generic: CALL MSDBL (NSTIM, IFORM, X, DISP, P, DS [, ...])
Specific: The specific interface names are S_MSDBL and D_MSDBL.

FORTRAN 77 Interface

Single: CALL MSDBL (NSTIM, NSUB, IFORM, X, LDX, ISCALE, DISP, LDDISP, P, LDP, DS, LDDS)
Double: The double precision name is DMSDBL.

Description

Routine **MSDBL** computes product-moment (double-centered) matrices from input dissimilarity matrices. The product-moment matrices output from **MSDBL** may be scaled either within each matrix, over all matrices input, or not at all.

The interest in product-moment matrices can be explained as follows: Let Λ denote a configuration of points in an *d*-dimensional Euclidean space with center at the origin. When the data is measured without error, the matrix $P = \Lambda\Lambda^T$ can also be written as the "double-centered" matrix (defined below) obtained from the matrix of squares of distances between the rows of

$$\Lambda\left(\delta_{ij}^2 = \sum_k \left(\lambda_{ik} - \lambda_{jk}\right)^2\right)$$

These distances are input, approximately, in the dissimilarities. Thus, an estimate for Λ can be obtained, approximately, by computing the double-centered matrix *P* from the squared dissimilarities and then computing Λ from the scaled eigenvectors of *P* (such that $P = \Lambda\Lambda^T$).

The computation in **MSDBL** proceeds as follows:

1. Each input dissimilarity matrix is transformed into a square symmetric matrix of distances. Asymmetric matrices are made symmetric by averaging the matrix of dissimilarities with its transpose.
2. Estimates for the square of the distances,

   $$\overline{\tilde{\delta}^2}$$

   are computed as the square of the estimated distances.
3. Let

   $$\overline{\tilde{\delta}^2}_{m\bullet\bullet}$$

   denote the average squared distance in a matrix *m* of squared distances, let

   $$\overline{\tilde{\delta}^2}_{mi\bullet}$$

   denote the average of the *i*-th row of estimated squared distances in matrix *m* and let

   $$\overline{\tilde{\delta}^2}_{m\bullet j}$$

   denote the average of the *j*-th column. The *m*-th product-moment matrix is computed from the *m*-th estimated squared distance matrix as

$$p_{mij} = -\left(\overline{\tilde{\delta}^2}_{mij} - \overline{\tilde{\delta}^2}_{mi\bullet} - \overline{\tilde{\delta}^2}_{m\bullet j} + \overline{\tilde{\delta}^2}_{m\bullet\bullet}\right)/2$$

The resulting matrix is said to be double-centered.

4. If the elements of P_m are to be scaled within matrix m, then the elements of P_m are divided by

$$\sqrt{\sum_{i,j} p_{\text{mij}}^2 / q^2}$$

where q = NSTIM so that q^2 is the total number of elements in the matrix. If the elements of P are to be scaled over all matrices, then the elements of each matrix are divided by

$$\sqrt{\sum_{m,i,j} p_{mij}^2 / \left(sq^2\right)}$$

where s = NSUB.

5. The matrix **DS** is computed as the sum over all subjects of the product-moment matrices, P_m.

Comments

Routine MSSTN may be used to obtain the matrix **X** with missing values estimated and changed in sign so that all estimates of missing values are negative. Routine **MSSTN** will also convert similarities to dissimilarities. Unless a ratio distance measure is observed, the user will usually call **MSSTN** prior to calling **MSDBL**.

Example

The following example illustrates the use of **MSDBL** in computing product-moment matrices for two input dissimilarity matrices. The input matrices are given as:

$$\begin{pmatrix} - & 4 & 1 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 2 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \quad \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 2 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

```
      USE MSDBL_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    IFORM, LDDISP, LDDS, LDP, LDX, NSTIM, NSUB
      PARAMETER  (IFORM=0, LDDISP=5, LDDS=5, LDP=5, LDX=10, &
                  NSTIM=5, NSUB=2)
!
```

```
      REAL       DISP(LDDISP,LDDISP,NSUB), DS(LDDS,NSTIM), &
                 P(LDP,LDP,NSUB), X(LDX,NSUB)
!
      DATA X/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 2.0, 3.0, 4.0/
!
      CALL MSDBL (NSTIM, IFORM, X, DISP, P, DS)
!
      CALL WRRRN ('The first matrix in DISP', DISP(1:,1:,1))
      CALL WRRRN ('The second matrix in DISP', DISP(1:,1:,2))
      CALL WRRRN ('The first matrix in P', P(1:,1:,1))
      CALL WRRRN ('The second matrix in P', P(1:,1:,2))
      CALL WRRRN ('DS', DS)
!
      END
```

Output

```
          The first matrix in DISP
          1       2       3       4       5
1    0.00   16.00    1.00    9.00    1.00
2   16.00    0.00    1.00    1.00    9.00
3    1.00    1.00    0.00    4.00    4.00
4    9.00    1.00    4.00    0.00   16.00
5    1.00    9.00    4.00   16.00    0.00
          The second matrix in DISP
          1       2       3       4       5
1    0.00    1.00    4.00    9.00    1.00
2    1.00    0.00    1.00    4.00    4.00
3    4.00    1.00    0.00    1.00    9.00
4    9.00    4.00    1.00    0.00   16.00
5    1.00    4.00    9.00   16.00    0.00
          The first matrix in P
          1       2       3       4       5
1   1.110  -1.931   0.274  -0.487   1.034
2  -1.931   1.110   0.274   1.034  -0.487
3   0.274   0.274  -0.182  -0.182  -0.182
4  -0.487   1.034  -0.182   1.338  -1.703
5   1.034  -0.487  -0.182  -1.703   1.338
          The second matrix in P
          1       2       3       4       5
1   0.500   0.000  -0.500  -1.000   1.000
2   0.000   0.000   0.000   0.000   0.000
3  -0.500   0.000   0.500   1.000  -1.000
4  -1.000   0.000   1.000   2.000  -2.000
5   1.000   0.000  -1.000  -2.000   2.000
                    DS
          1       2       3       4       5
1   0.805  -0.966  -0.113  -0.743   1.017
2  -0.966   0.555   0.137   0.517  -0.243
3  -0.113   0.137   0.159   0.409  -0.591
4  -0.743   0.517   0.409   1.669  -1.852
5   1.017  -0.243  -0.591  -1.852   1.669
```

MSINI

more...

Computes initial estimates in multidimensional scaling models.

Required Arguments

NSTIM — Number of stimuli in each dissimilarity matrix. (Input)

IFORM — Storage option for the data in each dissimilarity matrix. (Input)
Each column of X contains one of the NSUB dissimilarity matrices in the storage mode specified by IFORM.

IFORM	Data Storage Mode
0	Symmetric storage mode without the diagonal elements. (Upper triangular matrix stored columnwise.) Consecutive elements of each column of X contain the (1, 2), (1, 3), (2, 3), (1, 4), (2, 4), (3, 4), …, (NSTIM - 1, NSTIM) elements of the dissimilarity matrix.
1	Square matrix in full storage mode. Consecutive elements of each column of X contain the (1, 1), (2, 1), (3, 1), …, (NSTIM, 1), (1, 2), (2, 2), …, (NSTIM, NSTIM) elements of the dissimilarity matrix.

X — NV by NSUB matrix containing the NSUB dissimilarity matrices, where NV = NSTIM * (NSTIM - 1)/2 if IFORM = 0, and NV = NSTIM * NSTIM if IFORM = 1. (Input)
If IFORM = 0, then the input data is assumed to be symmetric, and the elements below and on the diagonal are not input. If IFORM = 1, all elements of each column of X are input, and the data for the column need not form a symmetric matrix. Missing values (NaN, "not a number") are not allowed in X, but the position of a missing element may be indicated as a negative dissimilarity. Since MSINI uses the absolute value of each element in X as the dissimilarity to be used in the estimation procedure, the sign of an element in X has no effect. See Comment 3.

IMOD — Vector of length 3 giving the model parameters to be estimated. (Input)
IMOD also gives the method of initialization to be used for each set of parameters. Each element of IMOD corresponds to a different parameter matrix. The correspondence is given as:

Parameter Matrix	
1	CFL–The configuration
2	W–The subject weights
3	WS–The stimulus weights

The value used for each element of **IMOD** tells how the parameter matrix is to be initialized.

Effect on Parameter Matrix	
0	The parameter matrix is not used.
1	The parameter matrix is input and its values are fixed. The parameter matrix may be standardized.
2	Initial estimates are input, but they may be changed by MSINI.
3	MSINI calculates the initial estimates.

IMOD(1) must be nonzero. **IFORM** must not be 0 if **IMOD**(3) is not zero. If **IMOD**(2) or **IMOD**(3) is 1, **IMOD**(1) must be 1. If **IMOD**(3) is 1, **IMOD**(2) must not be 2 or 3.

CFL — NSTIM by **NDIM** matrix containing the estimated stimulus coordinates. (Input/Output, if **IMOD**(1) = 1 or 2; Output, otherwise)

W — NSUB by **NDIM** matrix of subject weights. (Input/Output, if **IMOD**(2) = 1 or 2, output, if **IMOD**(2) = 3, not referenced if **IMOD**(2) = 0)
W is not referenced and can be dimensioned as a 1 by 1 matrix if **IMOD**(2) = 0.

WS — NSTIM by **NDIM** matrix of stimulus weights. (Input/Output, if **IMOD**(3) = 1 or 2; Output, if **IMOD**(3) = 3; not referenced if **IMOD**(3) = 0)
WS is not referenced and can be dimensioned as a 1 by 1 matrix if **IMOD**(3) = 0.

WMIN — Minimum weight in **W** prior to adjustment. (Output, if **IMOD**(2) = 2 or 3; not referenced if **IMOD**(2) = 0 or 1)
If **WMIN** is negative, the weights in **W** are adjusted such that all weights are positive by subtracting **WMIN** from each element in **W**.

WSMIN — Minimum weight in **WS** prior to adjustment. (Output, if **IMOD**(3) = 2 or 3; not referenced if **IMOD**(3) = 0 or 1)
If **WSMIN** is negative, the weights in **WS** are adjusted such that all weights are positive by subtracting **WSMIN** from each element in **WS**.

Optional Arguments

NSUB — Number of dissimilarity matrices to be used in the analysis. (Input)
Default: NSUB = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

NDIM — Number of dimensions in the solution. (Input)
Default: NDIM = size (CFL,2).

LDCFL — Leading dimension of CFL exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCFL = size (CFL,1).

LDW — Leading dimension of W exactly as specified in the dimension statement in the calling program. (Input)
Default: LDW = size (W,1).

LDWS — Leading dimension of WS exactly as specified in the dimension statement in the calling program. (Input)
Default: LDWS = size (WS,1).

FORTRAN 90 Interface

Generic: CALL MSINI (NSTIM, IFORM, X, IMOD, CFL, W, WS, WMIN, WSMIN [, ...])

Specific: The specific interface names are S_MSINI and D_MSINI.

FORTRAN 77 Interface

Single: CALL MSINI (NSTIM, NSUB, IFORM, X, LDX, IMOD, NDIM, CFL, LDCFL, W, LDW, WS, LDWS, WMIN, WSMIN)

Double: The double precision name is DMSINI.

Description

Routine MSINI computes initial estimates for the stimulus configuration (Λ = CFL), subject weights (W = W), and stimulus weights (Π = WS) in multidimensional scaling models. The number of dimensions in the solution must also be input. Routine MSINI requires complete (i.e., no missing values) dissimilarity matrices as input. Consequently, missing data must be replaced by an estimate (often an average of other dissimilarities). Because the

absolute values of dissimilarities are used, missing dissimilarities may be denoted by changing their sign to be negative. Estimation of missing values, and further standardization, can be performed through the use of routine MSSTN.

In some cases, **MSINI** can use values input in parameter matrices **CFL**, **W**, or **WS** in order to compute initial estimates for other parameter matrices. For example, values input in matrix **CFL** may be used in the estimation of initial estimates for **W** or **WS**. Because of the method of estimation, values input for some parameter matrices will not effect the estimate computed for other matrices. In particular, values input in **W** will not effect the estimation of **CFL**, and values input in **WS** will not effect the estimation of either **CFL** or **W**. Note that some combinations of input and estimated matrices are not even allowed (see the option parameter **IMOD**). Also, note that when the configuration matrix **CFL** is input and fixed (except for standardization), computed estimates for all weights **W** and **WS** are arbitrarily taken as 1.

Let

$$\tilde{\delta}^2_{ijl}$$

denote the squared distance between stimulus *i* and stimulus *j* for matrix (subject) *l*, let

$$\overline{\delta}^2_{i\bullet l}$$

denote the average of the squared distances in the *i*-th row for the *l*-th subject, let

$$\overline{\delta}^2_{\bullet jl}$$

be similarly defined, and let

$$\overline{\delta}^2_{\bullet\bullet l}$$

denote the average of all squared distances for the *l*-th subject. If each dissimilarity input in **X** is measured without error, then the dissimilarities and the distances are identical. In **MSINI**, the errors observed in the dissimilarities,

$$\tilde{\delta}^2_{ijl}$$

are assumed to be small so that good estimates for the squared distances may be computed by squaring each dissimilarity (after first subtracting the constant obtained in Step 1 below). The computations proceed as follows:

1. The squared distance matrices are double-centered using the product moment transformation

$$p_{ijl} = -\left(\tilde{\delta}^2_{ijl} - \overline{\tilde{\delta}^2}_{i\bullet l} - \overline{\tilde{\delta}^2}_{\bullet jl} + \overline{\tilde{\delta}^2}_{\bullet\bullet l}\right)/2$$

The matrix formed by averaging the product moment matrices P_l (over subjects) is computed as $\overline{P}$.

2. If the configuration has been input and cannot be modified (i.e., if **IMOD**(1) is 1), then all weights to be estimated are taken as 1, and the computations continue in Step 8 below.

3. If the configuration matrix has not been input, then a preliminary estimate is obtained by first computing the eigenvectors (**Γ**) corresponding to the *d*-largest eigenvalues of $\overline{P}$.

   The configuration is then estimated as $\mathbf{\Gamma\Delta}^{1/2}$ where **Δ** is the square matrix containing the eigenvalues along the diagonal and zeros off the diagonal.

4. If the subject weights **W** are to be estimated, or if they can be modified (i.e., if **IMOD**(2) is 2 or greater), then a SUMSCALE procedure (De Leeuw and Pruzansky, 1978) is used to estimate the weights (regardless of the values input) and to "rotate" the configuration estimates. This is done as follows:

   A. The matrices

   $$C_l = \Phi^{-1}\Lambda^T P_l \Lambda \Phi^{-1}$$

   are computed, where **Φ** = **Δ** if **Δ** has been computed, and where the diagonal elements of **Φ** are the diagonal elements of $\mathbf{\Lambda}^T\mathbf{\Lambda}$ otherwise (the off-diagonal elements of **Φ** are always zero).

   B. An orthogonal matrix *Q* is found such that the sum of the squared off-diagonal elements of $Q^T C_l Q$ is minimized over all matrices *C*. (See IMSL routine **KPRIN**, in *Chapter 9*.

   C. A new configuration estimate is obtained by "rotating" the current estimate, i.e., $\mathbf{\Lambda}_n = \mathbf{\Lambda}$.

   D. The subject weights for subject *l* are taken as the diagonal elements of $Q^T C_l Q$.

5. If the subject weights have been computed and the minimum weight in **W** is negative, add its absolute value to all elements in **W** to ensure that all estimated stimulus weights are nonnegative.

6. If the stimulus weights are to be estimated (i.e., if (**IMOD**(3) is 2 or 3), then least-squares estimates are used. The least-squares model is obtained by substituting predicted distance for actual distance in the multidimensional scaling model specified by **IMOD** (see the chapter introduction for a discussion of the models available). Least-squares fitting is then performed over the **NSUB** subjects.

7. If the stimulus weights have been computed and the minimum weight in **WS** is negative, its absolute value is added to all elements in **WS** to ensure that all estimated stimulus weights are nonnegative.

8. The estimates are standardized (even when **IMOD**(*i*) = 2) as follows:

   A. If **IMOD**(2) is not zero, then let

   $$r_i = \lambda_i^T \lambda_i$$

   where λ_i is the *i*-th column of the configuration matrix. Let w_i denote the *i*-th column of the subject weight matrix. Standardize Λ such that the diagonal elements of $\Lambda^T\Lambda$ are 1. Multiply w_i by r_i.

   B. If **IMOD**(2) = 0 but **IMOD**(3) is not zero, then compute r_i and standardize the configuration matrix as above. Multiply the *i*-th column of **WS** by r_i.

   C. If both **IMOD**(2) and **IMOD**(3) are nonzero, then compute

   $$s_i = \sqrt{w_i^T w_i}$$

   and standardize **W** such that W^TW is an identity matrix. Multiply the *i*-th column of **WS** by c_i.

Comments

1. Workspace may be explicitly provided, if desired, by use of **M2INI/DM2INI**. The reference is:

   ```
   CALL M2INI (NSTIM, NSUB, IFORM, X, LDX, IMOD, NDIM, CFL, LDCFL, W, LDW, WS, LDWS,
       WMIN, WSMIN, TR, XX, DISP, DS, EWK1, EWK2, IEWK, C)
   ```

   The additional arguments are as follows:

   TR — Real work vector of length max(**NDIM** + 1, **NSUB**, **NSTIM**).

   XX — Real work vector of length **NSTIM** * **NSTIM**.

   DISP — Real work vector of length **NSTIM** * **NSTIM** * **NSUB**.

   DS — Real work vector of length **NSTIM** * **NSTIM**.

   EWK1 — Real work vector of length 3 * **NSTIM**.

   EWK2 — Real work vector of length max(5 * **NSTIM**, 4 * **NSUB**).

   IEWK — Integer work vector of length **NSTIM**.

   C — Real work vector of length **NDIM** * **NDIM** * **NSUB**.

2. Informational error

Type	Code	Description
4	1	The sum of the product moment matrices for the data input in X has less than NDIM positive eigenvalues. Rerun with NDIM = number of positive eigenvalues or less or provide initial estimates for the configuration matrix CFL.

3. Routine MSSTN may be used to obtain the matrix X with missing values estimated and changed in sign so that all estimates of missing values are negative. Routine MSSTN will also convert similarities to dissimilarities. Unless a ratio distance measure is observed, the user will usually call MSSTN prior to calling MSINI.

Example

The following example illustrates the use of MSINI to obtain initial estimates for an individual differences model when symmetric dissimilarities matrices obtained from two subjects are input. The input matrices are given as:

$$\begin{pmatrix} - & 4 & 1 & 3 & 1 \\ & - & 1 & 1 & 3 \\ & & - & 2 & 2 \\ & & & - & 4 \\ & & & & - \end{pmatrix} \quad \begin{pmatrix} - & 1 & 2 & 3 & 1 \\ & - & 1 & 2 & 2 \\ & & - & 1 & 3 \\ & & & - & 4 \\ & & & & - \end{pmatrix}$$

Estimates obtained from MSINI are not optimal. Usually an optimizing multidimensional scaling routine will be called with the initial estimates computed in MSINI.

```
      USE UMACH_INT
      USE MSINI_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    IFORM, LDCFL, LDW, LDWS, LDX, NDIM, NSTIM, NSUB
      PARAMETER  (IFORM=0, LDCFL=5, LDW=2, LDWS=5, LDX=10, NDIM=2, &
                 NSTIM=5, NSUB=2)
!
      INTEGER    IMOD(3), NOUT
      REAL       CFL(LDCFL,NDIM), W(LDW,NDIM), WMIN, WS(LDWS,NDIM), &
                 WSMIN, X(LDX,NSUB)
!
      DATA X/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0, 1.0, &
          2.0, 1.0, 3.0, 2.0, 1.0, 1.0, 2.0, 3.0, 4.0/
      DATA IMOD/3, 3, 0/
!
      CALL UMACH (2, NOUT)
!
      CALL MSINI (NSTIM, IFORM, X, IMOD, CFL, W, WS, WMIN, WSMIN)
!
      CALL WRRRN ('The Configuration', CFL)
      CALL WRRRN ('Subject weights', W)
      WRITE (NOUT,99999) WMIN
```

```
      !
      99999 FORMAT (/, ' WMIN = ', F12.4)
      !
            END
```

Output

```
      The Configuration
                1        2
      1   0.2279   0.6854
      2  -0.0808  -0.6584
      3  -0.1728  -0.0090
      4  -0.6621  -0.2287
      5   0.6879   0.2107
      Subject weights
              1       2
      1   7.078   8.533
      2   9.615   0.000
      WMIN =       0.0000
```

MSTRS

Computes various stress criteria in multidimensional scaling.

Required Arguments

DIST — Vector of length **N** containing the distances. (Input)
Missing values are not allowed in **DIST**.

DISP — Vector of length **N** containing the disparities. (Input)

A — The intercept. (Input)
If **INTCEP** = 0, **A** is not used.

B — The slope. (Input)
If **ISLOPE** = 0, **B** is not used.

POWER — Power to use in the stress function. (Input)
POWER must be greater than or equal to 1.

STRSS — The computed stress criterion. (Output)

WT — The weight used in computing the stress. (Output)
If the weight is too large, a maximum weight is used. See the Description section.

Optional Arguments

N — Number of distances and disparities. (Input)
Default: **N** = size (**DIST**,1).

INTCEP — Intercept option parameter. (Input)
If **INTCEP** = 0, the intercept is not used in the model. If **INTCEP** = 1, the intercept is used in the model.
Default: **INTCEP** = 1.

ISLOPE — Slope option parameter. (Input)
If **ISLOPE** = 0, the slope **B** is not used. If **ISLOPE** = 1, the slope is used.
Default: **ISLOPE** = 1.

ISTRS — Stress option parameter. (Input)
Default: **ISTRS** = 1.

ISTRS	Stress Criterion Used
0	Log stress
1	Stress weighted by the inverse of the sum of the squared disparities
2	Stress weighted by the inverse of the sum of the centered squared disparities

FORTRAN 90 Interface

Generic: CALL MSTRS (DIST, DISP, A, B, POWER, STRSS, WT [, ...])
Specific: The specific interface names are S_MSTRS and D_MSTRS.

FORTRAN 77 Interface

Single: CALL MSTRS (N, DIST, DISP, INTCEP, A, ISLOPE, B, POWER, ISTRS, STRSS, WT)
Double: The double precision name is DMSTRS.

Description

Routine MSTRS computes the value of stress criteria commonly used in multidimensional scaling. Routine MSTRS allows transformed values of the disparities and distances to be input and will compute the stress on the transformed values. Additionally, the user can input a slope and/or an intercept to be used in the stress computations, and the stress can be computed using an arbitrary L_p norm as well as the squared error norm in which $p = 2$.

Let

$$\delta_i^*$$

denote a disparity, δ_i denote the corresponding distance, α denote the intercept, and let β denote the slope. If INTCEP = 0, then set $\alpha = 0$. If ISLOPE = 0, then set $\beta = 1$.

Set $\varepsilon = 0.001$, and let

$$\tau = \sum_{i=1}^{n} |\delta_i^* - \alpha - \beta\delta_i|^p$$

When ISTRS = 0, the stress is computed as

$$\phi_0 = n\ln\left[\max\left(n\varepsilon,\tau\right)\right]$$

where n is the number of nonmissing disparities, and p = **POWER** is the power to be used. This stress formula, when optimized, can lead to to normal distribution theory maximum likelihood estimation. It can not be used in nonmetric scaling. The weight is computed as $n/\max(n\boldsymbol{\varepsilon},\ \tau)$.

When **ISTRS** is 1, the stress is computed as

$$\phi_1 = \frac{\tau}{\max\left(\varepsilon\tau,\ \sum_{i=1}^{n}\left|\ \delta_i^*\ \right|^P\right)}$$

and the weight returned is given as

$$1 / \max\left(\varepsilon\tau, \sum_{i=1}^{n} |\delta_i^*|^p\right)$$

Takane, Young, and de Leeuw (1977) recommend using this formula when the data is not column conditional (i.e., whenever the stress is computed over one or more dissimilarity matrices rather than over one column in a single matrix). When **ISTRS** = 2, the stress is given by

$$\phi_2 = \frac{\tau}{\max\left(\varepsilon\tau,\ \sum_{i=1}^{n}\left|\ \delta_i^* - \overline{\delta}_\bullet^*\ \right|^P\right)}$$

where

$$\overline{\delta}_\bullet^* = \sum_{i=1}^{n} \delta_i^*$$

is the average of the nonmissing disparities. The weight is computed as

$$1 / \max\left(\varepsilon\tau, \sum_{i=1}^{n} |\delta_i^* - \overline{\delta}_\bullet^*|^p\right)$$

Takane, Young, and de Leeuw (1977) recommend this stress for column conditional data.

Missing values (NaN) are not allowed in **DIST** while missing disparities in **DISP** are not used in the computations. If all disparities are missing, the stress criteria is set to 0, and the weight (**WT**) is set to missing (NaN).

In general, a single call to **MSTRS** would be made for each strata ("conditionality group") in the data.

Example

The following example illustrates the computation of stress when the log of the distances and disparities are input. For this example, `ISTRS` is 1 and `POWER` is 2.

```
      USE MSTRS_INT
      USE UMACH_INT
      USE SDOT_INT

      IMPLICIT   NONE
      INTEGER    INTCEP, ISLOPE, ISTRS, N
      REAL       A, POWER
      PARAMETER  (A=0.0, INTCEP=0, N=10, POWER=2.0)
!
      INTEGER    I, NOUT
      REAL       ALOG, B, DISP(N), DIST(N), STRSS, WT
      INTRINSIC  ALOG
!
      DATA DIST/4.0, 1.5, 1.25, 3.0, 1.75, 2.0, 1.0, 3.5, 2.5, 3.75/
      DATA DISP/4.0, 1.0, 1.0, 3.0, 1.0, 2.0, 1.0, 3.0, 2.0, 4.0/
!                                Transform the data
      DO 10  I=1, N
         DIST(I) = ALOG(DIST(I))
         DISP(I) = ALOG(DISP(I))
   10 CONTINUE
!                                Compute a slope
      B = SDOT(N,DISP,1,DIST,1)/SDOT(N,DIST,1,DIST,1)
!                                Compute the stress
      CALL MSTRS (DIST, DISP, A, B, POWER, STRSS, WT, INTCEP=INTCEP)
!                                Print results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) STRSS, WT
!
99999 FORMAT (' STRSS = ', F12.4, '    WT = ', F12.4)
      END
```

Output

```
 STRSS =       0.0720    WT =       0.1385
```

Density and Hazard Estimation

Routines

Usage Notes

The routines described in this chapter compute estimates for smoothing parameters and estimates in models for estimating density and hazard functions. For density estimation, the penalized likelihood method of Scott (1976) may be used to obtain smooth estimates for arbitrary (smooth) densities. Alternatively, the routines DESKN and DNFFT obtain density estimates by the kernel method for a given window width and kernel function. Routine DNFFT uses a Gaussian kernel, while for routine DESKN, the kernel is provided by the user. Finally, routine DESPT finds linear or quasi-cubic interpolated estimates of a density. Tapia and Thompson (1978) discuss all of these methods.

For hazard estimation, the methods of Tanner and Wong (1984) are used to obtain estimates of the smoothing parameters in a modified likelihood. These methods are implemented in routines HAZRD and HAZEZ, the difference between the routines is in the ease of use and the options offered. For given smoothing parameters, the routine HAZST may be used to obtain estimates for the hazard function.

DESPL

more...

Performs nonparametric probability density function estimation by the penalized likelihood method.

Required Arguments

X — Vector of length **NOBS** containing the random sample of responses. (Input)

NODE — Number of mesh nodes for the discrete probability density estimate. (Input)
NODE must be an odd integer greater than 4.

BNDS — Vector of length 2 containing the upper and lower endpoints for the interval of support of the density. (Input)
The node values are taken as **BNDS**(1), **BNDS**(1) + *h*, …, **BNDS**(2), where
h = (**BNDS**(2) **-** **BNDS**(1))/(**NODE** **-** 1). All observations in vector **X** should be in the support interval.

DENS — Vector of length **NODE** containing the estimated values of the discrete pdf at the **NODE** equally spaced mesh nodes. (Input/Output, if **INIT** ≠ 0; Output, otherwise)
If **INIT** is not zero, then **DENS**(1) through **DENS**(**NODE**) contain the (positive) initial estimates on input. The sum of these estimates times the window width *h* (see **BNDS**) must equal 1.0, i.e., the integral of the density must be 1.

STAT — Vector of length 4 containing output statistics. (Output)
STAT(1) and **STAT**(2) contain the log-likelihood and the log-penalty terms, respectively. **STAT**(3) and **STAT**(4) contain the estimated mean and variance for the estimated density.

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOBS** = size (**X**,1).

INIT — Initialization option. (Input)
INIT = 0 means that a bootstrap procedure is used to obtain initial estimates for the density. Otherwise, user-supplied initial estimates are contained in DENS on entry into DESPL.
Default: INIT = 0.

ALPHA — Penalty-weighting factor that controls the smoothness of the estimate. (Input)
For standard normal data, ALPHA = 10.0 works well. Other values that might be tried are 1.0 and 100.0. ALPHA must be greater than 0.0.
Default: ALPHA = 10.0.

MAXIT — Maximum number of iterations allowed in the iterative procedure. (Input)
MAXIT = 30 is typical.
Default: MAXIT = 30.

EPS — Convergence criterion. (Input)
When the Euclidean norm of the changes to DENS is less than EPS, convergence is assumed. EPS = 0.0001 is typical.
Default: EPS = 0.0001.

NMISS — Number of missing values in X. (Output)

FORTRAN 90 Interface

Generic: CALL DESPL (X, NODE, BNDS, DENS, STAT [, ...])
Specific: The specific interface names are S_DESPL and D_DESPL.

FORTRAN 77 Interface

Single: CALL DESPL (NOBS, X, NODE, BNDS, INIT, ALPHA, MAXIT, EPS, DENS, STAT, NMISS)
Double: The double precision name is DDESPL.

Description

Routine DESPL computes piecewise linear estimates of a one-dimensional density function for a given random sample of observations. These estimates are discussed in detail in Scott et al. (1980), and in Tapia and Thompson (1978, Chapter 5). The estimator of the density function is piecewise linear over the finite interval (BNDS(1) to BNDS(2)), is nonnegative, and integrates to one. A penalty method is used to ensure "smooth" behavior of the estimator. The criterion function to be maximized is a discrete approximation to

$$\Phi = \prod_{i=1}^{n} f(x_i) \exp\left(-\alpha \int \left| \frac{d^2 f(t)}{dt^2} \right|^2 dt\right)$$

where n = **NOBS** and $f(t)$ is a density function. Let m = **NODE**. The discrete approximation is as follows: The density f is estimated at each of the equally spaced grid points t_j, for $j = 1, \ldots, m$, with restriction $f(t_1) = f(t_m) = 0.0$; the density at each data point x_i is then estimated using linear interpolation. The integral of the second derivative of the square of f is approximated using the piecewise linear function defined by the estimates of f at the grid points t_j.

Because ln **Φ** is actually maximized, the criterion can be separated into a likelihood term (returned in **STAT**(1)) and a penalty term (returned in **STAT**(2)).

The parameter **α** (= **ALPHA**) determines the amount of "smoothness" in the estimate. The larger the value of **α**, the smoother the resulting estimator for f. In practice, the user should pick **α** as small as possible such that there is not excessive bumpiness in the estimator. One way of doing this is to try several values of **α** that differ by factors of 10. The resulting estimators can then be graphically displayed and examined for bumpiness. **α** could then be chosen from the displayed density estimates. IMSL routines can be used to produce line printer plots (**PLOTP**) of the estimated density. For a random sample from the standard normal distribution, **α** = 10.0 works well. Note that **α** changes with scale. If x is multiplied by a factor **β**, **α** should be multiplied by a factor β^5.

The second choice to be made in using **DESPL** is the mesh for the estimator. The mesh interval (**BNDS**(1), **BNDS**(2)) should be picked as narrow as possible since a narrow mesh will speed algorithm convergence. Note, however, that points outside the interval (**BNDS**(1), **BNDS**(2)) are not included in the likelihood. Because of this fact, **DESPL** actually estimates a density that is conditional on the mesh interval (**BNDS**(1), **BNDS**(2)). The number of mesh nodes, **NODE**, should be as small as possible, but large enough to exhibit the "fine" structure of the density. One possible method for determining **NODE** is to use **NODE** = 21 initially. With **NODE** = 21, find an acceptable value for **α**. When an acceptable value for **α** has been found, increase or decrease **NODE** as required.

STAT(3) and **STAT**(4) contain "exact" estimates of the mean and variance when the estimated piecewise linear density is used in the required integrals. Routine DESPT may be used to find interpolated estimates for the density at any point x given the **NODE** estimates of the density returned in **DENS**.

Comments

1. Workspace may be explicitly provided, if desired, by use of **D2SPL/DD2SPL**. The reference is:

   ```
   CALL D2SPL (NOBS, X, NODE, BNDS, INIT, ALPHA, MAXIT, EPS, DENS, STAT, NMISS,
       HESS, LDHESS, ILOHI, DENEST, B, IPVT, WK2, XWK)
   ```

   The additional arguments are as follows:

   HESS — Work vector of length 7 * (**NODE** – 2).

LDHESS — Leading dimension of HESS exactly as specified in the dimension statement in the calling program. (Input)
The leading dimension must be set to 7.

ILOHI — Integer work vector of length 2 * NODE.

DENEST — Work vector of length 3 * NODE.

B — Work vector of length NODE.

IPVT — Integer work vector of length NODE - 2.

WK2 — Work vector of length NODE – 2.

XWK — Work vector of length NOBS. If X is sorted with all missing (NaN, not a number) values at the end, then XWK is not needed. If X is not needed, X and XWK can share the same storage location.

2. Informational error

Type	Code	Description
3	1	The maximum number of iterations is exceeded.

3 Routine DESPT may be used after the estimates DENS have been obtained in order to obtain an interpolated estimate of the density at new points. Use AMESH = BNDS in calling DESPT.

Example

An estimate for a density function of unknown form using a random sample of size 10 and 13 mesh points with α = 10 is estimated as follows:

```
      USE DESPL_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS, NODE
      PARAMETER  (NOBS=10, NODE=13)
!
      INTEGER    NOUT
      REAL       BNDS(2), DENS(NODE), STAT(4), X(NOBS)
!
      DATA BNDS/-3., 3./
      DATA X/-.9471, -.7065, -.2933, -.1169, .2217, .4425, .4919, &
          .5752, 1.1439, 1.3589/
!
      CALL DESPL (X, NODE, BNDS, DENS, STAT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' DENS = '',9F7.4, /, 9X, 4F7.4)') DENS
      WRITE (NOUT,'('' Log-likelihood term = '', F7.3, /, &
                  '' Log-penalty term    = '', F7.3, /, &
                 '' Mean                 = '', F7.3, /, &
                 '' Variance             = '', F7.3)') STAT
      END
```

Output

```
DENS =  0.0000 0.0014 0.0356 0.1111 0.2132 0.3040 0.3575 0.3565 0.2947
        0.1986 0.0986 0.0288 0.0000
Log-likelihood term = -11.968
Log-penalty term    =  -1.303
Mean                =   0.217
Variance            =   1.042
```

The following figure shows the affect of various choices of α. For larger α, the density estimate is smoother.

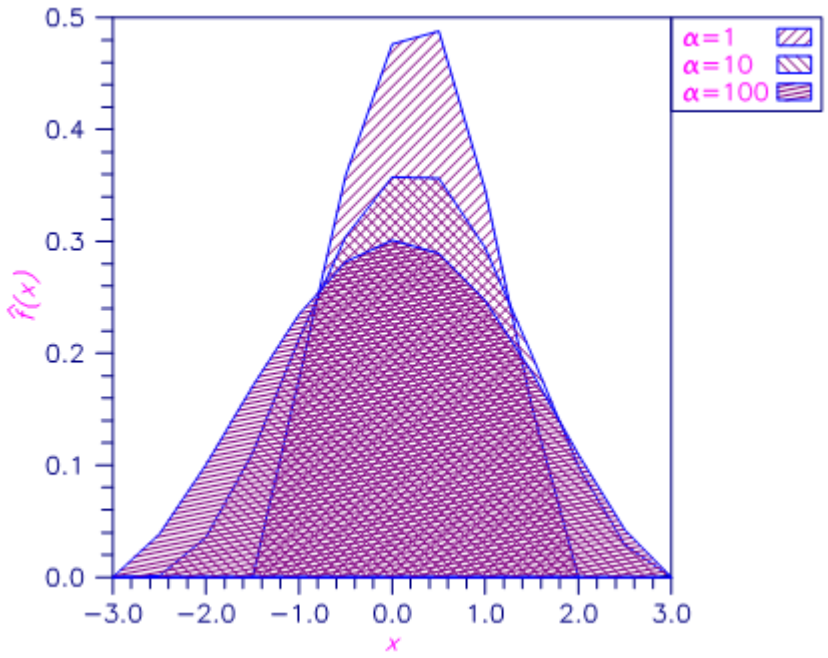


Figure 20, Density Estimates Using α = 1, 10, 100

DESKN

Performs nonparametric probability density function estimation by the kernel method.

Required Arguments

XKER — User-supplied FUNCTION to compute the kernel at any point on the real line. The form is XKER(Y), where:

Y — Point at which the kernel is to be evaluated.

XKER — Value of the kernel at point Y.

X — Vector of length NOBS containing the random sample of observations. (Input)

WINDOW — Window width for the kernel function. (Input)
Generally, several different values of WINDOW should be tried.

XMAX — Cutoff value such that XKER(Y) = 0.0 for all |Y| greater than XMAX. (Input)
If XMAX exists, then the kernel function is 0.0 for all Y greater in absolute value than XMAX, and the efficiency of the computations is enhanced. If no such XMAX exists or the user does not wish to make use of XMAX, then XMAX should be assigned any nonpositive value.

XPT — Vector of length NXPT containing the values at which a density estimate is desired. (Input)
If XMAX is greater than zero, then XPT must be sorted from smallest to largest.

DENS — Vector of length NXPT containing the density estimates at the points specified in XPT. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

NXPT — Number of points at which a density estimate is desired. (Input)
Default: NXPT = size (XPT,1).

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: CALL DESKN (XKER, X, WINDOW, XMAX, XPT, DENS [, ...])

Specific: The specific interface names are S_DESKN and D_DESKN.

FORTRAN 77 Interface

Single: CALL DESKN (XKER, NOBS, X, WINDOW, XMAX, NXPT, XPT, DENS, NMISS)

Double: The double precision name is DDESKN.

Description

Routine DESKN computes kernel estimates of the density function for a random sample of (scalar-valued) observations. The kernel estimate of the density at the point *y* is given by.

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n} K[(y - x_i)/h]$$

where

$$\hat{f}(y)$$

is the estimated density at *y*, *K* is the kernel function, x_i denotes the *i*-th observation, *n* is the number of observations, and *h* is a fixed constant (called the "window width") supplied by the user.

One is usually interested in computing the density estimates using several values of the window width *h*. Tapia and Thompson (1978), Chapter 2, give some considerations relevant to the choice of *h*. Some common kernel functions (see Tapia and Thompson 1978, page 60) are given as follows.

Name	Function
Uniform	$K(y) = \begin{cases} 0.5 & \text{for } \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Triangular	$K(y) = \begin{cases} 1 - \lvert y \rvert & \text{for } \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Biweight	$K(y) = \begin{cases} 15(1 - y^2)^2/16 & \text{for } \lvert y \rvert < 1 \\ 0 & \text{elsewhere} \end{cases}$
Normal	$K(y) = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} \quad -\infty < y < \infty$

The computation can be made much more efficient when the kernel is nonzero over a finite range since observations outside this range can be ignored in the computation of the density. In this case, the array XPT is assumed to be sorted.

Comments

1. Informational error

Type	Code	Description
4	7	Negative kernel functions are not allowed.

2. Routine may be used to obtain interpolated density estimates from the **NXPT** density estimates returned in **DENS**. Array **AMESH** in **DESPT** corresponds to array **XPT** in **DESKN**.

Example

In this example, the standard normal density function is estimated at 13 points using a random sample of 10 points from a standard normal distribution. The biweight kernel function is used. The actual density for the standard normal density is also reported in the output for comparison. The random sample is generated using routines **RNSET** and **RNNOR** in Chapter 18, "Random Number Generation".

```
      USE RNSET_INT
      USE RNNOR_INT
      USE DESKN_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS, NXPT
      REAL       C1, WINDOW, XMAX
      PARAMETER  (C1=0.3989423, NOBS=10, NXPT=13, WINDOW=2.0, XMAX=1.0)
!
      INTEGER    I, NMISS, NOUT
      REAL       DENS(NXPT), EXP, X(NOBS), XKER, XPT(NXPT)
      INTRINSIC  EXP
      EXTERNAL   XKER
!
      DATA XPT/-3.0, -2.5, -2.0, -1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5, &
          2.0, 2.5, 3.0/
!
      CALL RNSET (1234457)

      CALL RNNOR (X)
      CALL DESKN (XKER, X, WINDOW, XMAX, XPT, DENS, NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'('' NMISS = '', I1)') NMISS
      WRITE (NOUT,'('' DENS Estimate = '', 10F6.4,/,8X,3F6.4)') DENS
      WRITE (NOUT,'('' DENS Exact    = '',10F6.4,/,8X,3F6.4)') &
                           (C1*EXP(-XPT(I)*XPT(I)/2.0),I=1,NXPT)
      END
      REAL FUNCTION XKER (Y)
      REAL       Y
!
      REAL       ABS
      INTRINSIC  ABS
!
```

```
          IF (ABS(Y) .LT. 1.0) THEN
             XKER = 15.0*(1.0-Y*Y)*(1.0-Y*Y)/16.0
          ELSE
             XKER = 0.0
          END IF
          RETURN
          END
```

Output

```
    NMISS = 0
    DENS Estimate = 0.00000.01180.07900.16980.26780.34670.36870.31840.22340.1391
            0.06120.01350.0005
    DENS Exact    = 0.00440.01750.05400.12950.24200.35210.39890.35210.24200.1295
            0.05400.01750.0044
```

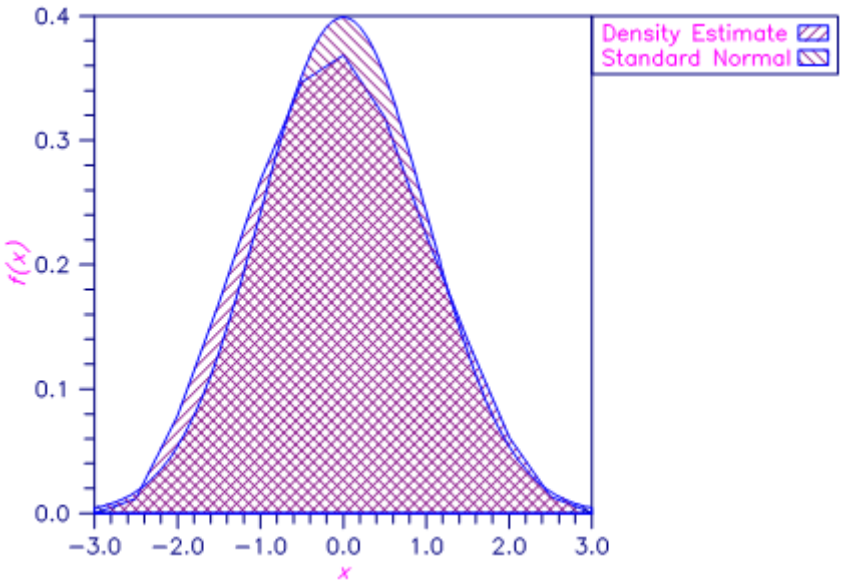


Figure 21, Density Estimate and Standard Normal Density

DNFFT

more...

Computes Gaussian kernel estimates of a univariate density via the fast Fourier transform over a fixed interval.

Required Arguments

X — Vector of length NOBS containing the data for which a univariate density estimate is desired. (Input)
X is not referenced and may be dimensioned of length 1 in the calling program if IFFT = 1.

BNDS — Vector of length 2 containing the minimum and maximum values of X at which the density is to be estimated. (Input)
Observations less than BNDS(1) or greater than BNDS(2) are ignored. If either range of the hypothesized density is infinite, a value equal to the smallest observation minus 3 * WINDOW is a good choice for BNDS(1), and a value equal to the largest observation plus 3 * WINDOW is a good choice for BNDS(2). Let STEP = (BNDS(2) - BNDS(1))/(NXPT - 1), and note that the density is estimated at the points BNDS(1) + *i* STEP where *i* = 0, 1, …, NXPT - 1. The density is assumed constant over the interval from BNDS(1) + *i* * STEP to BNDS(1) + (*i* + 1) * STEP.

WINDOW — Window width for the kernel function. (Input)
Generally, several different values for WINDOW should be tried. When several different values are tried, use the IFFT option.

COEF — Vector of length NXPT containing the Fourier coefficients. (Input, if IFFT = 1; output, otherwise)

DENS — Vector of length NXPT containing the density estimates. (Output)
The density is estimated at the points BNDS(1) + *i* * STEP, *i* = 0, 1, …, NXPT - 1, where STEP = (BNDS (2) - BNDS(1))/(NXPT - 1).

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

FRQ — Vector of length **NOBS** containing the frequency of the corresponding element of **X**. (Input)
If **FRQ**(1) is **-** 1.0, then the vector **FRQ** is not used and all frequencies are taken to be one. **FRQ** is also not used if **IFFT** = 1. In either case, **FRQ** may be dimensioned of length 1 in the calling program.
Default: **FRQ**(1) = –1.0.

IFFT — Fourier transform option parameter. (Input)
If **IFFT** = 1, then **COEF** contains the Fourier coefficients on input, and the coefficients are not computed. Otherwise, the coefficients are computed. This option is used when several different values for **WINDOW** are to be tried. On the first call to **DNFFT**, **IFFT** = 0 and the coefficients **COEF** are computed. On subsequent calls, these coefficients do not need to be recomputed (but only if **NXPT** also remains fixed).
Default: **IFFT** = 0.

NXPT — Number of equally-spaced points points at which the density is to be estimated. (Input)
Routine **DNFFT** is most efficient when **NXPT** is a power of 2. Little efficiency is lost if **NXPT** is a product of small primes. Because of the method of estimation, **NXPT** should be large, say greater than 64.
Default: **NXPT** = 128.

NRMISS — Number of rows of data that contain missing values in **X** or **FRQ**. (Output) **NRMISS** is not referenced if **IFFT** = 1.

FORTRAN 90 Interface

Generic: **CALL DNFFT** (**X**, **BNDS**, **WINDOW**, **COEF**, **DENS** [, ...])
Specific: The specific interface names are **S_DNFFT** and **D_DNFFT**.

FORTRAN 77 Interface

Single: **CALL DNFFT** (**NOBS**, **X**, **FRQ**, **BNDS**, **WINDOW**, **IFFT**, **NXPT**, **COEF**, **DENS**, **NRMISS**)
Double: The double precision name is **DDNFFT**.

Description

Routine **DNFFT** computes Gaussian kernel estimates of the density function for a random sample of (scalar-valued) observations using a Gaussian kernel (normal density). The computations are comparatively fast because they are performed through the use of the fast Fourier transform. Routine **DESKN** should be used in place of **DNFFT** if a kernel other than the Gaussian kernel is to be used, if a irregular grid is desired, or if the approximations in **DNFFT** are not acceptable. Because of its speed, **DNFFT** will usually be preferred to **DESKN**.

A Gaussian kernel estimate of the density at the point *y* is given by:

$$\hat{f}(y) = \frac{1}{nh}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{y-x_i}{h}\right)^2\right]$$

where

$$\hat{f}(y)$$

is the estimated density at y, x_i denotes the i-th observation, n is the number of observations, and h is a fixed constant called the *window width* supplied by the user. If density estimates for several different window sizes are to be computed, then **DNFFT** performs a fast Fourier transform on the data only during the first call (when **IFFT** is zero). On subsequent calls (with **IFFT** set at 1), the Fourier transform of the data need not be recomputed.

If the same value of **NXPT** is to be used with several different input vectors **X**, then the computations can be made faster by the use of **D2FFT**. In **D2FFT**, it is assumed that some constants required by the Fourier transform and its inverse have already been computed via routine **FFTRI** (IMSL MATH/LIBRARY) in work array **WFFTR**. If **D2FFT** is called repeatedly with the same value of **NXPT**, **WFTTR** need only be computed once.

Routine **DNFFT** is an implementation of *Applied Statistics* algorithm AS 176 (Silverman 1982) using IMSL routines for the fast Fourier transforms. Modification to algorithm AS 176, as discussed in Silverman (1986, pages 61–66), gives the details of the computational method. The basic idea is to partition the support of the density into **NXPT** equally-sized nonoverlapping intervals. The frequency of the observations within each interval is then computed, and the Fourier transform of the frequencies obtained. Since the kernel density estimate is the convolution of the frequencies with the Gaussian kernel (for given window size), the Fourier coefficients for the Gaussian kernel density estimates are computed as the product of the coefficients obtained for the frequencies, times the Fourier coefficients for the Gaussian kernel function. The discrete Fourier coefficients for the Gaussian kernel may be estimated from the continuous transform. The inverse transform is then used to to obtain the density estimates.

Because the fast Fourier transform is used in computing

$$\hat{f}(y)$$

the computations are relatively fast (providing that **NXPT** is a product of small primes). To maintain precision, a large number of intervals, say 256, is usually recommended. Tapia and Thompson (1978), Chapter 2, give some considerations relevant to the choice of the window size parameter **WINDOW**. Generally, several different window sizes should be tried in order to obtain the best value for this parameter.

Comments

1. Workspace may be explicitly provided, if desired, by use of **D2FFT/DD2FFT**. The reference is:

   CALL D2FFT (NOBS, X, FRQ, BNDS, WINDOW, IFFT, NXPT, COEF, DENS, NRMISS, WFFTR)

The additional argument is:

WFFTR – Work vector of length 2 * NXPT + 15. See Comment 3. (Input)

2. Informational errors

Type	Code	Description
4	1	The sum of the frequencies must be positive.
4	2	Each frequency must be nonnegative.
4	3	There are no valid observations remaining after all missing values are eliminated.

3. WFFTR is computed in DNFFT. If D2FFT is to be called, WFFTR must first be computed via the following FORTRAN statement:

```
CALL FFTRI (NXPT, WFFTR)
```

If DD2FFT is used, call DFFTRI instead of FFTRI. WFFTR need not be recomputed between successive calls to D2FFT if NXPT does not change.

Example

In this example, the density function is estimated at 64 points using a random sample of 150 points from a standard normal distribution. The actual density for the standard normal density is also reported in the output for comparison. The random sample is generated using routines RNSET and RNNOR in Chapter 18, "Random Number Generation".

```
      USE RNSET_INT
      USE RNNOR_INT
      USE DNFFT_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOBS, NXPT
      REAL       CONS, WINDOW
      PARAMETER  (CONS=0.39894228, NOBS=150, NXPT=64, WINDOW=0.25)
!
      INTEGER    I, NOUT
      REAL       BNDS(2), COEF(NXPT), DENS(NXPT), EXP, STEP, X(NOBS), XX
      INTRINSIC  EXP
!
      DATA BNDS/-4.0, 3.875/
!
      CALL RNSET (123457)
      CALL RNNOR (X)
!
      CALL DNFFT (X, BNDS, WINDOW, COEF, DENS, NXPT=NXPT)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998)
99998 FORMAT ('   X  DENSITY POPULATION')
```

```
      STEP = (BNDS(2)-BNDS(1))/(NXPT-1)
      XX   = BNDS(1)
      DO 10  I=1, NXPT, 2
         WRITE (NOUT,99999) XX, DENS(I), CONS*EXP(-XX*XX/2.0)
99999    FORMAT (F6.2, 2F8.4)
         XX = XX + STEP*2.0
   10 CONTINUE
!
      END
```

Output

```
 X  DENSITY POPULATION
-4.00  0.0000  0.0001
-3.75  0.0000  0.0004
-3.50  0.0000  0.0009
-3.25  0.0000  0.0020
-3.00  0.0001  0.0044
-2.75  0.0011  0.0091
-2.50  0.0089  0.0175
-2.25  0.0345  0.0317
-2.00  0.0772  0.0540
-1.75  0.1204  0.0863
-1.50  0.1573  0.1295
-1.25  0.2076  0.1826
-1.00  0.2682  0.2420
-0.75  0.2987  0.3011
-0.50  0.2976  0.3521
-0.25  0.3072  0.3867
 0.00  0.3336  0.3989
 0.25  0.3458  0.3867
 0.50  0.3169  0.3521
 0.75  0.2834  0.3011
 1.00  0.2683  0.2420
 1.25  0.2242  0.1826
 1.50  0.1557  0.1295
 1.75  0.1182  0.0863
 2.00  0.0946  0.0540
 2.25  0.0569  0.0317
 2.50  0.0199  0.0175
 2.75  0.0033  0.0091
 3.00  0.0002  0.0044
 3.25  0.0000  0.0020
 3.50  0.0000  0.0009
 3.75  0.0000  0.0004
```

DESPT

Estimates a probability density function at specified points using linear or cubic interpolation.

Required Arguments

XPT — Vector of length NODE containing the points at which an estimate of the probability density is desired. (Input)

AMESH — Vector of length NORD for IOPT = 2 or 4, and of length 2 for IOPT = 1 or 3. (Input)
If IOPT = 2 or 4, AMESH(I) contains the abscissas corresponding to each density estimate in DENS(I). In this case, the abscissas must be specified in increasing order. If IOPT = 1 or 3 (i.e., for an equally spaced mesh), then the lower and upper ends of the mesh are specified by AMESH(1) and AMESH(2), respectively, with the increment between mesh points given by (AMESH(2) - AMESH(1))/(NORD - 1).

DENS — Vector of length NORD containing the density function values corresponding to each of the NORD abscissa values. (Input)

DENEST — Vector of length NODE containing the density function estimates for the points in XPT. (Output)

Optional Arguments

NODE — Number of points at which the density is desired. (Input)
Default: NODE = size (XPT,1).

IOPT — Interpolation option parameter. (Input)
Default: IOPT = 1.

IOPT	Method of interpolation
1	Linear on equally spaced points
2	Linear with unequal spacing
3	Cubic on equally spaced points
4	Cubic with unequal spacing

NORD — Number of ordinates supplied. (Input)

NORD must be greater than one for linear interpolation, and greater than three for cubic interpolation.

Default: NORD = size (DENS,1).

FORTRAN 90 Interface

Generic: CALL DESPT (XPT, AMESH, DENS, DENEST [, ...])

Specific: The specific interface names are S_DESPT and D_DESPT.

FORTRAN 77 Interface

Single: CALL DESPT (NODE, XPT, IOPT, NORD, AMESH, DENS, DENEST)

Double: The double precision name is DDESPT.

Description

Routine DESPT computes an estimate of a density function using either linear or cubic spline interpolation on a set $\{(X_i, F_i)$, for $i = 1, \ldots, N\}$, where F_i = DENS(i), N = NODE, and where the values of the the grid points X_i can be obtained from the vector AMESH. The value of IOPT indicates the type of interpolation (linear or cubic) to be performed and whether the mesh values are equally spaced. When IOPT is 1 or 3, then an equally spaced mesh is used with mesh values given by

$$\text{AMESH}(1) + i * \text{DELTA}$$

for $i = 0, 1, \ldots, N - 1$, where

$$\text{DELTA} = (\text{AMESH}(2) - \text{AMESH}(1))/(\text{NORD} - 1)$$

IOPT = 2 or 4 yields an unequally spaced mesh with all mesh values contained in the vector AMESH.

The Akima cubic spline method of interpolation (Akima 1970) is used for the cubic interpolation.

Comments

1. Workspace may be explicitly provided, if desired, by use of D2SPT/DD2SPT . The reference is:

   CALL D2SPT (NODE, XPT, IOPT, NORD, AMESH, DENS, DENEST, CF, X, BREAK)

   The additional arguments are as follows:

   CF — Work vector of length 4 * NORD for IOPT = 3 or 4. CF is not used for other values of IOPT and may be dimensioned of length 1.

X — Work vector of length NORD for IOPT = 3 or 4. X is not used for other values of IOPT and may be dimensioned of length 1.

BREAK — Work vector of length NORD for IOPT = 3 or 4. BREAK is not used for other values of IOPT and may be dimensioned of length 1.

2. Array AMESH is the same as array BNDS in DESPL when IOPT is 1 or 3, and the same as array XPT in DESKN when IOPT is 2 or 4.

Example

The standard normal density is to be estimated via a grid of points over which the density is provided. Grid points are given by (0.0, 0.5, 1.0, 1.5, 2.0) while the density is to be estimated (via linear interpolation) at the four points (0.25, 0.75, 1.25, 1.75). For comparison, both the exact and the estimated density values at each of the four points are printed.

```
      USE DESPT_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NODE, NORD
      PARAMETER  (NODE=4, NORD=5)
!
      INTEGER    I, NOUT
      REAL       AMESH(2), DENEST(NODE), DENS(NORD), EXP, F, H, X, X0, &
                 XPT(NODE)
      INTRINSIC  EXP
!
      DATA XPT/0.25, 0.75, 1.25, 1.75/
      DATA AMESH/0, 2/
!
      F(X) = 0.3989423*EXP(-X*X/2.0)
!                                Get the grid values
      H  = (AMESH(2)-AMESH(1))/(NORD-1)
      X0 = AMESH(1)
      DO 10  I=1, NORD
         DENS(I) = F(X0)
         X0      = X0 + H
   10 CONTINUE
!                                Get the density estimates
      CALL DESPT (XPT, AMESH, DENS, DENEST)
!                                Print the results
      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(''    X       DENEST      EXACT'')')
      DO 20  I=1, NODE
         WRITE (NOUT,'(F5.2, 2F12.5)') XPT(I), DENEST(I), F(XPT(I))
   20 CONTINUE
      END
```

Output

```
     X       DENEST      EXACT
  0.25     0.37550     0.38667
```

```
    0.75     0.29702     0.30114
    1.25     0.18574     0.18265
    1.75     0.09175     0.08628
```

HAZRD

Performs nonparametric hazard rate estimation using kernel functions and quasi-likelihoods.

Required Arguments

X — NOBS by *m* matrix containing the raw data, where *m* = 1 if ICEN = 0, and *m* = 2 otherwise. (Input)

IRT — Column number in X of the event times. (Input)

KMIN — Minimum number for parameter *k*. (Input)
Parameter *k* is the number of nearest neighbors to be used in computing the *k*-th nearest neighbor distance.

INK — Increment between successive values of parameter *k*. (Input)

NK — Number of values of *k* to be considered. (Input)
HAZRD finds the optimal value of *k* over the grid given by: KMIN + (*j* - 1) * INK, for *j* = 1, ..., NK.

ST — Vector of length NOBS containing the times of occurrence of the events, sorted from smallest to largest. (Output)
Vector ST is obtained from the matrix X and should be used as input to routine HAZST.

JCEN — Vector of length NOBS containing the sorted censor codes. (Output)
Censor codes are sorted corresponding to the events ST(*i*), with censored observations preceding tied failures. Vector JCEN is obtained from the censor codes in X, if present, and is used as input to routine HAZST.

ALPHA — Optimal estimate for the parameter α. (Output)

BTA — Optimal estimate for the parameter β. (Output)

K — Optimal estimate for the parameter *k*. (Output)

VML — Optimum value of the criterion function. (Output)

H — Vector of length NOBS * 5 containing constants needed to compute the *k*-th nearest failure distances, and the observation weights. (Output)
H is used as input to routine HAZST.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICEN — Censoring option. (Input)
If ICEN = 0, then all of the data is treated as exact data with no censoring. For ICEN > 0, column ICEN of X contains the censoring codes. A censoring code of 0 means an exact event (failure). A censoring code of 1 means that the observation was right censored at the event time.
Default: ICEN = 0.

IWTO — Weight option. (Input)
If IWTO = 1, then weight ln(1 + 1/(NOBS - *i* + 1)) is used for the *i*-th smallest observation. Otherwise, weight 1/(NOBS - *i* + 1) is used.
Default: IWTO = 0.

NGRID — Grid option. (Input)
If NGRID = 0, a default grid is used to locate an initial starting value for parameter BTA. For NGRID > 0, a user-defined grid is used. This grid is defined as BSTART + (*j* - 1) * GINC, for *j* = 1, ..., NGRID, where BSTART, GINC, and NGRID are input.
Default: NGRID = 0.

BSTART — First value to be used in the user-defined grid. (Input)
Not used if NGRID = 0.

GINC — For a user-defined grid, the increment between successive grid values of BTA. (Input)
Not used if NGRID = 0.

IPRINT — Printing option. (Input)
If IPRINT = 1, the grid estimates and the optimized estimates are printed for each value of *k*. Otherwise, no printing is performed.
Default: IPRINT = 0.

ISORT — Sorting option. (Input)
If ISORT = 1, then the event times are not automatically sorted by HAZRD. Otherwise, sorting is performed with exact failure times following tied right-censored times.
Default: ISORT = 0.

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: CALL HAZRD (X, IRT, KMIN, INK, NK, ST, JCEN, ALPHA, BTA, K, VML, H [, ...])

Specific: The specific interface names are S_HAZRD and D_HAZRD.

FORTRAN 77 Interface

Single: CALL HAZRD (NOBS, X, LDX, IRT, ICEN, IWTO, NGRID, BSTART, GINC, KMIN, INK, NK, IPRINT, ISORT, ST, JCEN, ALPHA, BTA, K, VML, H, NMISS)

Double: The double precision name is DHAZRD.

Description

Routine HAZRD is an implementation of the methods discussed by Tanner and Wong (1984) for estimating the hazard rate in survival or reliability data with right censoring. It uses the biweight kernel,

$$K(x) = \begin{cases} \frac{15}{16}(1-x^2)^2 & \text{for } |x| < 1 \\ 0 & \text{elsewhere} \end{cases}$$

and a modified likelihood to obtain data-based estimates of the smoothing parameters α, β, and *k* needed in the estimation of the hazard rate. For kernel $K(x)$, define the "smoothed" kernel $K_S(x - x_{(j)})$ as follows:

$$K_S\left(x - x_{(j)}\right) = \frac{1}{\alpha d_{jk}} K\left(\frac{x - x_{(j)}}{\beta d_{jk}}\right)$$

where d_{jk} is the distance to the *k*-th nearest failure from $x_{(j)}$, and $x_{(j)}$ is the *j*-th ordered observation (from smallest to largest). For given α and β, the hazard at point *x* is then

$$h(x) = \sum_{i=1}^{N} \left\{ \left(1 - \delta_i\right) w_i K_s\left(x - x_{(i)}\right) \right\}$$

where *N* = NOBS, δ_i is the *i*-th observation's censor code (1 = censored, 0 = failed), and w_i is the *i*-th ordered observation's weight, which may be chosen as either $1/(N - i + 1)$, or $\ln(1 + 1/(N - i + 1))$. After the smoothing parameters have been obtained, the hazard may be estimated via HAZST.

Let

$$H(x) = \int_o^x h(s)ds$$

The likelihood is given by

$$L = \prod_{i=1}^{N} \{h(x_i)^{(1-\delta_i)} \exp(-H(x_{(i)}))\}$$

where **Π** denotes product. Since the likelihood leads to degenerate estimates, Tanner and Wong (1984) suggest the use of a modified likelihood. The modification consists of deleting observation x_i in the calculation of $h(x_i)$ and $H(x_i)$ when the likelihood term for x_i is computed using the usual optimization techniques. **α** and **β** for given *k* can then be estimated.

Estimates for **α** and **β** are computed as follows: for given **β**, a closed form solution is available for **α**. The problem is thus reduced to the estimation of **β**. A grid search for **β** is first performed. Experience indicates that if the initial estimate of **β** from this grid search is greater than, say, e^6, then the modified likelihood is degenerate because the hazard rate does not change with time. In this situation, **β** should be taken to be infinite, and an estimate of **α** corresponding to infinite **β** should be directly computed. When the estimate of **β** from the grid search is less than e^6, a secant algorithm is used to optimize the modified likelihood. The secant algorithm iteration stops when the change in **β** from one iteration to the next is less than 10^{-5}. Alternatively, the iterations may cease when the value of **β** becomes greater than e^6, at which point an infinite **β** with a degenerate likelihood is assumed.

To find the optimum value of the likelihood with respect to *k*, a user-specified grid of *k*-values is used. For each grid value, the modified likelihood is optimized with respect to **α** and **β**. That grid point, which leads to the smallest likelihood, is taken to be the optimal *k*.

Comments

1. Informational Errors

Type	Code	Description
4	18	All observations are missing (NaN, not a number) values.

2. In the optimization routines, the parameterization is changed to β^* and α^*, where $\beta^* = -\ln(\beta)$ and $\alpha^* = -\ln(\alpha)$. The default grid uses –8, –4, –3, –2.5, –2, –1.5, –1, –0.5, and 0.5 for β^*. This corresponds to a grid in **β** of 2981, 54.6, 20.08, 12.18, 7.39, 4.48, 2.72, 1.64, and .61. The grid **β** that maximizes the modified "likelihood" is used as the starting point for the iterations.

3. If the initial estimate of β as determined from the grid or as given by the user is greater than 400 (actually e^6), then infinite β is assumed, and an analytic estimate of α based upon infinite β is used. In the optimization, if it is determined that β must be greater than 1000, then an infinite β is assumed. Infinite β corresponds to a "flat" hazard rate.

Programming Notes

1. The routine HAZST may be used to estimate the hazard on a grid of points once the optimal values for α, β and k have been found. The user should also consider using the "easy-to-use" version of HAZRD, routine HAZEZ.

2. If sorting of the data is performed by HAZRD, then the sorted array will be such that all censored observations at a given time precede all failures at that time. To specify an arbitrary pattern of censored/failed observations at a given time point, the **ISORT** = 1 option must be used. In this case, it is assumed that the times have already been sorted from smallest to largest.

3. The smallest value of k must be greater than the largest number of tied failures since d_{jk} must be positive for all j. (Censored observations are not counted.) Similarly, the largest value of k must be less than the total number of failures. If the grid specified for k includes values outside the allowable range, then a warning error is issued; but k is still optimized over the allowable grid values.

4. The secant algorithm iterates on the transformed parameter $\beta^* = \exp(-\beta)$. This assures a positive β, and it also seems to lead to a more desirable grid search. All results returned to the user are in the original parameterization, however.

5. Since local minimums have been observed in the modified likelihood, it is recommended that more than one grid of initial values for α and β be used.

Example

The following example is taken from Tanner and Wong (1984). The data are from Stablein, Carter, and Novak (1981) and involve the survival times of individuals with nonresectable gastric carcinoma. Individuals treated with radiation and chemotherapy are used. For each value of k from 18 to 22 with increment of 2, the default grid search for β is performed. Using the optimal value of β in the grid, the optimal parameter estimates of α and β are computed for each value of k. The final solution is the parameter estimates for the value of k which optimizes the modified likelihood (**VML**). Because the **IPRINT** = 1 option is in effect, **HAZRD** prints all of the results in the output.

```
      USE HAZRD_INT
      USE UMACH_INT
      USE WRRRN_INT
      USE WRIRN_INT
```

```
      IMPLICIT   NONE
      INTEGER    ICEN, INK, IPRINT, IRT, ISORT, KMIN, LDX, &
                 NK, NOBS
      PARAMETER  (ICEN=2, INK=2, IPRINT=1, IRT=1, ISORT=1, &
                  KMIN=18, LDX=45, NK=3, NOBS=45)
!
      INTEGER    JCEN(NOBS), K, NMISS, NOUT
      REAL       ALPHA, BTA, H(5*NOBS), ST(NOBS), VML, X(NOBS,2)
!
      DATA X/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
          170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
          401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
          892, 1031, 1033, 1306, 1335, 1366, 1452, 1472, 36*0, 9*1/
!
      CALL HAZRD (X, IRT, KMIN, INK, NK, ST, JCEN, ALPHA, BTA, &
                  K, VML, H, ICEN=ICEN, IPRINT=IPRINT, ISORT=ISORT, &
                  NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) NMISS
99999 FORMAT (/' NMISS = ', I4/)
      CALL WRRRN ('ST', ST, 1, NOBS, 1)
      CALL WRIRN ('JCEN', JCEN, 1, NOBS, 1)
      CALL WRRRN ('H', H, NOBS, 5, NOBS)
      END
```

Output

```
                 *** GRID SEARCH FOR K =    18 ***
          ALPHA                 BETA                  VML
       4.578322           2980.958008          -266.804504
       4.543117             54.598148          -266.619690
       4.336464             20.085537          -265.541168
       4.019334             12.182494          -264.001404
       3.542742              7.389056          -262.540100
       2.990577              4.481689          -262.511810
       2.351537              2.718282          -262.633911
       1.584173              1.648721          -262.158264
       0.966332              1.000000          -262.868408

                 *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                 BETA                  VML
       1.695147              1.769263          -262.118530

                 *** GRID SEARCH FOR K =    20 ***
          ALPHA                 BETA                  VML
       4.053934           2980.958008          -266.525970
       4.032835             54.598148          -266.401428
       3.905046             20.085537          -265.648315
       3.687815             12.182494          -264.401672
       3.304344              7.389056          -262.665924
       2.822716              4.481689          -262.080078
       2.252759              2.718282          -262.445251
       1.555777              1.648721          -261.772278
       0.955586              1.000000          -262.617645

                 *** OPTIMAL PARAMETER ESTIMATES ***
```

```
              ALPHA                  BETA                   VML
              1.540533               1.631551            -261.771484

                    *** GRID SEARCH FOR K =    22 ***
              ALPHA                  BETA                   VML
              3.656405            2980.958008            -267.595337
              3.641593              54.598148            -267.498596
              3.550560              20.085537            -266.903870
              3.388752              12.182494            -265.859131
              3.071474               7.389056            -264.066040
              2.645036               4.481689            -263.038696
              2.137399               2.718282            -263.334717
              1.512606               1.648721            -262.639740
              0.936368               1.000000            -262.682739

                    *** OPTIMAL PARAMETER ESTIMATES ***
              ALPHA                  BETA                   VML
              1.342176               1.450016            -262.561188

                 *** THE FINAL SOLUTION     (K =    20) ***
              ALPHA                  BETA                   VML
              1.540533               1.631551            -261.771484
    NMISS =    0

                                       ST
         1        2        3        4        5        6        7        8
      17.0     42.0     44.0     48.0     60.0     72.0     74.0     95.0
         9       10       11       12       13       14       15       16
     103.0    108.0    122.0    144.0    167.0    170.0    183.0    185.0
        17       18       19       20       21       22       23       24
     193.0    195.0    197.0    208.0    234.0    235.0    254.0    307.0
        25       26       27       28       29       30       31       32
     315.0    401.0    445.0    464.0    484.0    528.0    542.0    567.0
        33       34       35       36       37       38       39       40
     577.0    580.0    795.0    855.0    882.0    892.0   1031.0   1033.0
        41       42       43       44       45
    1306.0   1335.0   1366.0   1452.0   1472.0
                                      JCEN
     1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
     0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
     0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1
    41  42  43  44  45
     1   1   1   1   1
                         H
            1        2        3        4        5
 1      217.0    218.0      1.0     21.0      1.0
 2      192.0    193.0      1.0     21.0      0.5
 3      190.0    191.0      1.0     21.0      0.3
 4      186.0    187.0      1.0     21.0      0.2
 5      174.0    175.0      1.0     21.0      0.2
 6      162.0    163.0      1.0     21.0      0.2
 7      160.0    161.0      1.0     21.0      0.1
 8      139.0    140.0      1.0     21.0      0.1
 9      131.0    132.0      1.0     21.0      0.1
10      126.0    127.0      1.0     21.0      0.1
11      112.0    113.0      1.0     21.0      0.1
12      102.0    110.0      2.0     22.0      0.1
13      123.0    125.0      3.0     23.0      0.1
14      126.0    128.0      3.0     23.0      0.1
```

15	132.0	135.0	5.0	25.0	0.1
16	130.0	137.0	5.0	25.0	0.1
17	133.0	145.0	5.0	25.0	0.1
18	135.0	147.0	5.0	25.0	0.1
19	137.0	149.0	5.0	25.0	0.1
20	148.0	160.0	5.0	25.0	0.1
21	167.0	174.0	6.0	26.0	0.0
22	166.0	175.0	6.0	26.0	0.0
23	182.0	191.0	6.0	26.0	0.0
24	204.0	212.0	9.0	29.0	0.0
25	212.0	213.0	9.0	29.0	0.0
26	231.0	234.0	14.0	34.0	0.0
27	275.0	278.0	14.0	34.0	0.0
28	294.0	297.0	14.0	34.0	0.0
29	311.0	314.0	15.0	35.0	0.0
30	343.0	345.0	16.0	36.0	0.0
31	357.0	359.0	16.0	38.0	0.0
32	382.0	384.0	16.0	38.0	0.0
33	392.0	394.0	16.0	38.0	0.0
34	395.0	397.0	16.0	38.0	0.0
35	610.0	612.0	16.0	43.0	0.0
36	670.0	672.0	16.0	45.0	0.0
37	689.0	697.0	17.0	45.0	0.0
38	699.0	707.0	17.0	45.0	0.0
39	838.0	846.0	17.0	45.0	0.0
40	840.0	848.0	17.0	45.0	0.0
41	1113.0	1121.0	17.0	45.0	0.0
42	1142.0	1150.0	17.0	45.0	0.0
43	1173.0	1181.0	17.0	45.0	0.0
44	1259.0	1267.0	17.0	45.0	0.0
45	1279.0	1287.0	17.0	45.0	0.0

HAZEZ

Performs nonparametric hazard rate estimation using kernel functions. Easy-to-use version of HAZRD.

Required Arguments

X — NOBS by *m* matrix containing the raw data, where *m* = 1 if ICEN = 0, and *m* = 2 otherwise. (Input)

IRT — Column number in X containing the times of occurrence of the events. (Input)

ST — Vector of length NOBS containing the times of occurrence of the events, sorted from smallest to largest. (Output)
Vector ST is obtained from matrix X and is used as input to routine HAZST.

JCEN — Vector of length NOBS containing the sorted censor codes. (Output)
Censor codes are sorted corresponding to the events ST(*i*), with censored observations preceding tied failures. Vector JCEN is obtained from the censor codes in X and is used as input to routine HAZST.

ALPHA — Optimal estimate for the parameter α. (Output)

BTA — Optimal estimate for the parameter β. (Output)

K — Optimal estimate for the parameter *k*. (Output)

VML — Optimal value of the criterion function. (Output)
VML is the "modified likelihood".

H — Vector of length 5 * NOBS containing the constants needed to compute the *k*-th nearest failure distance and the observation weights. (Output)
H is used as input to routine HAZST.

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICEN — Censoring option. (Input)
If ICEN = 0, then all of the data is treated as exact data with no censoring. For ICEN > 0, column ICEN of X contains the censoring codes. A censoring code of 0 means an exact event (failure). A censoring code of 1 means that the observation was right censored at the event time.
Default: ICEN = 0.

IPRINT — Printing option. (Input)
If IPRINT = 1, the grid estimates and the optimized estimates are printed for each value of *k*. Otherwise, no printing is performed.
Default : IPRINT = 0.

NMISS — Number of missing (NaN, not a number) values in X. (Output)

FORTRAN 90 Interface

Generic: CALL HAZEZ (X, IRT, ST, JCEN, ALPHA, BTA, K, VML, H [, …])
Specific: The specific interface names are S_HAZEZ and D_HAZEZ.

FORTRAN 77 Interface

Single: CALL HAZEZ (NOBS, X, LDX, IRT, ICEN, IPRINT, ST, JCEN, ALPHA, BTA, K, VML, H, NMISS)
Double: The double precision name is DHAZEZ.

Description

Routine HAZEZ is an implementation of the methods discussed by Tanner and Wong (1984) for estimating the hazard rate in survival or reliability data with right censoring. It uses the biweight kernel,

$$K(x) = \begin{cases} \frac{15}{16}(1-x^2)^2 & \text{for } |x| < 1 \\ 0 & \text{elsewhere} \end{cases}$$

and a modified likelihood to obtain data-based estimates of the smoothing parameters α, β, and k needed in the estimation of the hazard rate. For kernel $K(x)$, define the "smoothed" kernel
$K_S(x - x_{(j)})$ as follows:

$$K_s\left(x - x_{(j)}\right) = \frac{1}{\alpha d_{jk}} K\left(\frac{x - x_{(j)}}{\beta d_{jk}}\right)$$

where d_{jk} is the distance to the k-th nearest failure from $x_{(j)}$, and $x_{(j)}$is the j-th ordered observation (from smallest to largest). For given **α** and **β**, the hazard at point x is given by:

$$h\left(x\right) = \sum_{i=1}^{N} \left\{ \left(1 - \delta_i\right) w_i K_s\left(x - x_{(i)}\right) \right\}$$

where N = **NOBS**, δ_i is the censor code (0 = failed, 1 = censored) for the i-th ordered observation, and w_i is the weight of the i-th ordered observation (given by $1/(N - i + 1)$). The hazard may be estimated via routine after the smoothing parameters have been obtained

Let

$$H(x) = \int_0^x h(s)ds$$

The likelihood is given by:

$$L = \prod_{i=1}^{N} \{h(x_i)^{(1-\delta_i)} \exp(-H(x_i))\}$$

where **Π** denotes product. Since the likelihood, as specified, will lead to degenerate estimates, Tanner and Wong (1984) suggest the use of a modified likelihood. The modification consists of deleting the observation x_i in the calculation of $h(x_i)$ and $H(x_i)$ when the likelihood term for x_i is computed. For a given k, **α** and **β** can then be estimated via the usual optimization techniques.

Estimates for **α** and **β** are computed as follows. For a given **β**, a closed form solution is available for **α**. The problem is thus reduced to the estimation of **β**. To estimate **α** and **β**, a grid search is first performed. Experience indicates that if the initial estimate of **β** from this grid search is greater than exp(6), then the modified likelihood is degenerate because the hazard rate does not change with time. In this situation, **β** should be taken to be infinite, and an estimate of **α** corresponding to infinite **β** is computed directly. When the estimate of **β** from the grid search is less than exp(6) (approximately 400), a secant algorithm is used to optimize the modified likelihood. The secant algorithm is said to have converged when the change in **β** from one iteration to the next is less than 0.00001. Additionally, convergence is assumed when the value of **β** becomes greater than exp(6). This corresponds to an infinite **β** with a degenerate likelihood.

A grid of k-values is used to find the optimum value of the likelihood with respect to k. The grid is determined by **HAZEZ** and consists of at most 10 points. The starting value in the grid is the smallest possible value of k. An increment of 2 is then used to obtain the remaining grid points.

For each grid value, the modified likelihood is optimized with respect to α and β. That grid point, which leads to the smallest likelihood, is taken to be the optimal *k*.

Comments

1. Informational errors

Type	Code	Description
4	6	All observations are missing (NaN, not a number) values.
4	7	There are not enough failing observations in X to continue.

2. The grid values in the initial grid search are given as follows: Let β^* = – 8, – 4, – 2, – 1, – 0.5,0.5,1, and 2, and

$$\beta = e^{-\beta^*}$$

For each value of β, VML is computed at the optimizing β. The maximizing β is used to initiate the iterations.

3. If the initial β^* is determined from the grid search to be less than –6, then it is presumed that β is infinite, and an analytic estimate of α based upon infinite β is used. Infinite β corresponds to a flat hazard rate.

Programming Notes

1. Routine HAZST may be used to estimate the hazard on a grid of points once the optimal values for α, β and *k* have been found. (The user should also consider using routine HAZRD, which allows for more options than HAZEZ.)

2. Routine HAZEZ assumes that censored observations precede failed observations at tied failure/censoring times.

3. The secant algorithm iterates on the transformed parameter β^* = exp(–β). This assures a positive β, and it also seems to lead to a more desirable grid search. All results returned to the user are in the original parameterization.

Example

The following example is illustrated in Tanner and Wong (1984), and the data are taken from Stablein, Carter, and Novak (1981). It involves the survival times of individuals with nonresectable gastric carcinoma. Only those individuals treated with radiation and chemotherapy are used.

```
      USE HAZEZ_INT
      USE UMACH_INT
      USE WRRRN_INT
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    ICEN, IPRINT, IRT, LDX, NOBS
      PARAMETER  (ICEN=2, IPRINT=1, IRT=1, LDX=45, NOBS=45)
!
      INTEGER    JCEN(NOBS), K, NMISS, NOUT
      REAL       ALPHA, BTA, H(5*NOBS), ST(NOBS), VML, X(NOBS,2)
!
      DATA X/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
          170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
          401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
          892, 1031, 1033, 1306, 1335, 1366, 1452, 1472, 36*0, 9*1/
!
      CALL HAZEZ (X, IRT, ST, JCEN, ALPHA, BTA, K, VML, H, ICEN=ICEN, &
                  IPRINT=IPRINT, NMISS=NMISS)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) NMISS
99999 FORMAT (/' NMISS = ', I4/)
      CALL WRRRN ('ST', ST, 1, NOBS, 1)
      CALL WRIRN ('JCEN', JCEN, 1, NOBS, 1)
      CALL WRRRN ('H', H, NOBS, 5, NOBS)
      END
```

Output

```
                 *** GRID SEARCH FOR K =     2 ***
          ALPHA                  BETA                   VML
       65.157967            2980.958008          -266.543945
       32.434208              54.598148          -262.551147
       17.100269              20.085537          -263.100769
       11.402525              12.182494          -264.410187
        7.263529               7.389056          -267.502014
        4.452315               4.481689          -270.548523
        2.689497               2.718282          -335.407288
        1.633702               1.648721          -338.566162
        0.995799               1.000000          -519.939514

                 *** OPTIMAL PARAMETER ESTIMATES ***
          ALPHA                  BETA                   VML
       32.219337              53.968315          -262.550781

                 *** GRID SEARCH FOR K =     4 ***
          ALPHA                  BETA                   VML
       25.596716            2980.958008          -266.471558
       20.476425              54.598148          -262.893860
       13.995192              20.085537          -262.792755
       10.109113              12.182494          -262.573212
        6.883837               7.389056          -263.030121
        4.407142               4.481689          -265.238647
        2.690131               2.718282          -265.606293
        1.633339               1.648721          -266.822693
```

```
 0.993371              1.000000            -271.831390

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
 8.530729              9.683726            -262.545593

       *** GRID SEARCH FOR K =     6 ***
 ALPHA                  BETA                   VML
16.828691           2980.958008            -266.729248
14.840095             54.598148            -264.019409
11.215133             20.085537            -262.844360
 9.013870             12.182494            -263.663391
 6.557755              7.389056            -263.283752
 4.330785              4.481689            -263.732697
 2.691744              2.718282            -264.613800
 1.633932              1.648721            -265.381866
 0.990891              1.000000            -266.242767

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
12.553377             28.178671            -262.529877

       *** GRID SEARCH FOR K =     8 ***
 ALPHA                  BETA                   VML
11.377748           2980.958008            -266.746185
10.773529             54.598148            -265.469299
 8.766835             20.085537            -262.476807
 7.427887             12.182494            -263.109009
 5.916299              7.389056            -264.492432
 4.184323              4.481689            -264.289886
 2.656351              2.718282            -264.807617
 1.623750              1.648721            -265.270691
 0.989442              1.000000            -264.738403

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
 8.522110             18.281288            -262.438568

       *** GRID SEARCH FOR K =    10 ***
 ALPHA                  BETA                   VML
 8.689023           2980.958008            -267.026093
 8.412854             54.598148            -266.250366
 7.196551             20.085537            -263.192688
 6.207793             12.182494            -262.648376
 5.143391              7.389056            -264.274384
 3.934601              4.481689            -264.523193
 2.630993              2.718282            -264.877869
 1.611710              1.648721            -264.332581
 0.984530              1.000000            -263.920013

       *** OPTIMAL PARAMETER ESTIMATES ***
 ALPHA                  BETA                   VML
 6.483376             13.956067            -262.589661

       *** GRID SEARCH FOR K =    12 ***
 ALPHA                  BETA                   VML
 6.669007           2980.958008            -266.778259
 6.551508             54.598148            -266.347595
 5.933167             20.085537            -264.141174
 5.252526             12.182494            -262.516205
```

```
4.471936              7.389056           -262.691589
3.598284              4.481689           -263.914032
2.557817              2.718282           -263.390106
1.588307              1.648721           -263.879578
0.973723              1.000000           -263.361908

      *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                  VML
4.923379              9.819798           -262.336670

      *** GRID SEARCH FOR K =    14 ***
ALPHA                  BETA                  VML
5.668086           2980.958008           -266.747559
5.595870             54.598148           -266.436584
5.195685             20.085537           -264.737946
4.685275             12.182494           -262.971497
4.044650              7.389056           -262.288147
3.335586              4.481689           -263.126434
2.496436              2.718282           -262.891663
1.585763              1.648721           -263.418976
0.969140              1.000000           -263.164032

      *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                  VML
4.145060              7.966486           -262.260559

      *** GRID SEARCH FOR K =    16 ***
ALPHA                  BETA                  VML
4.970138           2980.958008           -266.419281
4.924928             54.598148           -266.199646
4.663393             20.085537           -264.938660
4.280633             12.182494           -263.266602
3.741570              7.389056           -262.020355
3.132969              4.481689           -262.401733
2.421248              2.718282           -262.555817
1.586469              1.648721           -262.426025
0.967658              1.000000           -263.135101

      *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                  VML
3.639074              6.767537           -261.987305

      *** GRID SEARCH FOR K =    18 ***
ALPHA                  BETA                  VML
4.578322           2980.958008           -266.804504
4.543117             54.598148           -266.619690
4.336464             20.085537           -265.541168
4.019334             12.182494           -264.001404
3.542742              7.389056           -262.540100
2.990577              4.481689           -262.511810
2.351537              2.718282           -262.633911
1.584173              1.648721           -262.158264
0.966332              1.000000           -262.868408

      *** OPTIMAL PARAMETER ESTIMATES ***
ALPHA                  BETA                  VML
1.695147              1.769263           -262.118530

      *** GRID SEARCH FOR K =    20 ***
ALPHA                  BETA                  VML
```

```
             4.053934            2980.958008            -266.525970
             4.032835              54.598148            -266.401428
             3.905046              20.085537            -265.648315
             3.687815              12.182494            -264.401672
             3.304344               7.389056            -262.665924
             2.822716               4.481689            -262.080078
             2.252759               2.718282            -262.445251
             1.555777               1.648721            -261.772278
             0.955586               1.000000            -262.617645
                    *** OPTIMAL PARAMETER ESTIMATES ***
             ALPHA                   BETA                   VML
             1.540533               1.631551            -261.771484

                *** THE FINAL SOLUTION     (K =    20) ***
             ALPHA                   BETA                   VML
             1.540533               1.631551            -261.771484

NMISS =    0

                                ST
     1        2        3        4        5        6        7        8
  17.0     42.0     44.0     48.0     60.0     72.0     74.0     95.0

     9       10       11       12       13       14       15       16
 103.0    108.0    122.0    144.0    167.0    170.0    183.0    185.0

    17       18       19       20       21       22       23       24
 193.0    195.0    197.0    208.0    234.0    235.0    254.0    307.0

    25       26       27       28       29       30       31       32
 315.0    401.0    445.0    464.0    484.0    528.0    542.0    567.0

    33       34       35       36       37       38       39       40
 577.0    580.0    795.0    855.0    882.0    892.0   1031.0   1033.0

    41       42       43       44       45
1306.0   1335.0   1366.0   1452.0   1472.0

                                     JCEN
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0

21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   1

41  42  43  44  45
 1   1   1   1   1

                       H
          1        2        3        4        5
 1    217.0    218.0      1.0     21.0      1.0
 2    192.0    193.0      1.0     21.0      0.5
 3    190.0    191.0      1.0     21.0      0.3
                     .
                     .
                     .
43   1173.0   1181.0     17.0     45.0      0.0
44   1259.0   1267.0     17.0     45.0      0.0
45   1279.0   1287.0     17.0     45.0      0.0
```

HAZST

Performs hazard rate estimation over a grid of points using a kernel function.

Required Arguments

ST — Vector of length NOBS containing the event times, sorted in ascending order. (Input)
ST may not contain missing values.

JCEN — Vector of length NOBS containing the censor codes. (Input)
JCEN(*i*) = 1 means that event *i* was (right) censored at time ST(*i*), *i* = 1, ..., NOBS. JCEN(*i*) = 0 means that event *i* was a failure at time ST(*i*).

NGRID — Number of grid points at which to compute the hazard. (Input)

GSTRT — First grid value. (Input)

GINC — Increment between grid values. (Input)

ALPHA — Value for parameter α. (Input)

BTA — Value for parameter β. (Input)

K — Value for parameter *k*. (Input)

H — Vector of length 5 * NOBS containing the constants used in computing the *k*-th failure distance. (Input, if IHCOMP = 1; Output, otherwise)

HAZ — Vector of length NGRID containing the estimated hazard rates. (Output)

Optional Arguments

NOBS — Number of observations. (Input)
If HAZRD or HAZEZ is called prior to this routine and the original data contained missing values, then NOBS in HAZST must be adjusted for the number of missing values from the value used in HAZRD or HAZEZ. That is, NOBS in HAZST is NOBS minus NMISS from HAZRD or HAZEZ.
Default: NOBS = size (ST,1).

IWTO — Weighting option. (Input)
IWTO = 1 means use weights ln(1 + 1/(NOBS - *i* + 1)). IWTO = 0 means use weights 1/(NOBS - *i* + 1). Not used if IHCOMP = 1.
Default: IWTO = 0.

IHCOMP — Option parameter. (Input)
If **IHCOMP** = 0, **H** is computed. If **IHCOMP** = 1, **H** has already been computed (generally by **HAZRD** or **HAZEZ**).
Default: **IHCOMP** = 0.

FORTRAN 90 Interface

Generic: **CALL HAZST** (**ST**, **JCEN**, **NGRID**, **GSTRT**, **GINC**, **ALPHA**, **BTA**, **K**, **H**, **HAZ** [, ...])
Specific: The specific interface names are **S_HAZST** and **D_HAZST**.

FORTRAN 77 Interface

Single: **CALL HAZST** (**NOBS**, **ST**, **JCEN**, **IWTO**, **NGRID**, **GSTRT**, **GINC**, **ALPHA**, **BTA**, **K**, **IHCOMP**, **H**, **HAZ**)
Double: The double precision name is **DHAZST**.

Description

Routine **HAZST** estimates the hazard function by use of the biweight kernel,

$$K(x) = \frac{15}{16}(1 - x^2)^2$$

Because a "smoothed" estimate is computed, one generally would use either routine HAZRD or HAZEZ routine to obtain maximum (modified) likelihood estimates of the smoothing parameters **α**, **β**, and *k*. Maximum (modified) likelihood estimates of these parameters are not required, however. A user-specified grid of points is generated. For each point, the hazard estimate is computed as

$$h(x) = \sum_{i=1}^{n} (1 - \delta_i) w_i K_s\left(x - x_{(i)}\right)$$

where *n* = **NOBS**, δ_i is the *i*-th observation's censoring code (0 = failed, 1 = censored), w_i is the *i*-th observation's weight (either 1/(*n* - *i* + 1) or ln(1 + 1/(*n* - *i* + 1)) depending upon **IWTO**), and $K_s(x - x_{(i)})$, the "smoothed kernel", is as follows:

$$K_s\left(x - x_{(i)}\right) = \frac{1}{\alpha d_{ik}} K\left(\frac{x - x_{(i)}}{\beta d_{ik}}\right)$$

Here, d_{ik} is the distance to the *k*-th nearest failure from the *i*-th observation. Because of the d_{ik}, **HAZST** requires the computation of matrix *H*, which contains constants needed to quickly compute d_{ik}. Often, *H* will have been computed in routine **HAZRD** or **HAZEZ**. In this case, the parameter **IHCOMP** should be set to zero and *H* should be input to **HAZST**. If *H* must be computed by **HAZST**, set **IHCOMP** = 1.

Comments

1. Informational error

Type	Code	Description
4	13	At least one missing (NaN, not a number) value was found in ST. Missing values are not allowed in this routine.

2. The user-defined grid is given by **GSTRT** + *j* * **GINC**, *j* = 0, ..., **NGRID** - 1.
3. Routine **HAZST** assumes that the grid points are new data points.

Example

The following example is a continuation of the example from **HAZRD**. The data are from Stablein, Carter, and Novak (1981), and involve the survival times of individuals with nonresectable gastric carcinoma. Only those individuals treated with both radiation and chemotherapy are used.

```
          USE HAZST_INT
          USE WRRRN_INT

          IMPLICIT   NONE
          INTEGER    K, NGRID, NOBS
          REAL       ALPHA, BTA, GINC, GSTRT
          PARAMETER  (ALPHA=1.540537, BTA=1.631553, GINC=10, GSTRT=0.0, &
                      K=20, NGRID=100, NOBS=45)
    !
          INTEGER    JCEN(NOBS), NOUT
          REAL       H(5*NOBS), HAZ(NGRID), ST(NOBS)
    !
          DATA ST/17, 42, 44, 48, 60, 72, 74, 95, 103, 108, 122, 144, 167, &
              170, 183, 185, 193, 195, 197, 208, 234, 235, 254, 307, 315, &
              401, 445, 464, 484, 528, 542, 567, 577, 580, 795, 855, 882, &
              892, 1031, 1033, 1306, 1335, 1366, 1452, 1472/
          DATA JCEN/36*0, 9*1/
    !
          CALL HAZST (ST, JCEN, NGRID, GSTRT, GINC, ALPHA, &
                      BTA, K, H, HAZ)
```

```
!
      CALL WRRRN ('Ten elements of HAZ', HAZ, 1, 10, 1)
      CALL WRRRN ('The first 10 rows of H', H, 10, 5, NOBS)
      END
```

Output

```
                          Ten elements of HAZ
        1          2          3          4          5          6          7
 0.000962   0.001111   0.001276   0.001451   0.001634   0.001819   0.002004

        8          9         10
 0.002185   0.002359   0.002523

           The first 10 rows of H
          1       2       3       4       5
 1   217.0   218.0     1.0    21.0     1.0
 2   192.0   193.0     1.0    21.0     0.5
 3   190.0   191.0     1.0    21.0     0.3
 4   186.0   187.0     1.0    21.0     0.2
 5   174.0   175.0     1.0    21.0     0.2
 6   162.0   163.0     1.0    21.0     0.2
 7   160.0   161.0     1.0    21.0     0.1
 8   139.0   140.0     1.0    21.0     0.1
 9   131.0   132.0     1.0    21.0     0.1
10   126.0   127.0     1.0    21.0     0.1
```

Line Printer Graphics

Routines

Usage Notes

The routine names in this chapter end with the letter "**P**" to indicate line printer plotting and every routine starts printing at the beginning of a new page.

Depending on the nature of plots, some routines allow the user to change page width and/or length. This capability is specified in each routine and, if allowed, can be done by calling the routine **PGOPT** (see Chapter 19, "Utilities") in advance. To change the page width, the user should make the following call to **PGOPT**:

```
CALL PGOPT(–1, IPAGEW)
```

where **IPAGEW** indicates the page width in columns. To change the page length, the user should make the following call to **PGOPT**:

```
CALL PGOPT(-2, IPAGEL)
```

where **IPAGEL** indicates the page length in rows. See the **PGOPT** document for more information.

VHSTP

Prints a vertical histogram.

Required Arguments

FRQ — Vector of length NBAR containing the frequencies or counts. (Input)
Elements of FRQ must be nonnegative.

TITLE — CHARACTER string containing main title. (Input)

Optional Arguments

NBAR – Number of bars. (Input)
If NBAR exceeds 100/(ISP + 1), then NBAR = 100/(ISP + 1) is used. NBAR must be positive.
Default: NBAR = size (FRQ,1).

ISP — Spacing between histogram bars. (Input)
ISP may be 0, 1, or 4.
Default: ISP= 4.

FORTRAN 90 Interface

Generic: CALL VHSTP (FRQ, TITLE [, ...])
Specific: The specific interface names are S_VHSTP and D_VHSTP.

FORTRAN 77 Interface

Single: CALL VHSTP (NBAR, FRQ, ISP, TITLE)
Double: The double precision name is DVHSTP.

Description

VHSTP prints a vertical histogram on not more than one printer page using not more than 50 vertical and 100 horizontal print positions. Spacing control is allowed on the horizontal axis.

Given a vector containing positive counts, **VHSTP** determines the maximum count T_{max}. Vertical printing position depends on K defined by $K = 1 + (T_{max} - 1)/50$. If a frequency is greater than K, then a character is printed on the corresponding position of the first horizontal line from above. Henceforth, K is reduced by $K/50$ for each horizontal line, and frequencies are compared to the new K.

Comments

1. Informational errors

Type	Code	Description
3	1	`ISP` is out of range. `ISP` = 0 is used.
3	3	`NBAR` * (`ISP` + 1) is less than 1 or greater than 100. The width of the histogram is set to 100, and 100/(`ISP` + 1) bars are printed. The number of class intervals will be printed completely if `ISP` ≠ 0 and will always be printed up to and including 100/(`ISP` + 1) even though the histogram body is only 100 spaces wide.
3	5	The maximum value in the vector `FRQ` is less than 1; therefore, the body of the histogram is blank.
3	6	`TITLE` is too long. `TITLE` was truncated from the right side.

2. Output is written to the unit specified by the routine **UMACH** (see the Reference Material section of this manual;).

3. `TITLE` is centered and placed at the top of the plot. The plot starts on a new page.

Example

Consider the data set in Example 1of the routine **OWFRQ** (see Chapter 1, "Basic Statistics".) This data set consists of the measurements (in inches) of precipitaion in Minneapolis/St. Paul during the month of March for 30 consecutive years. We use the routine **OWFRQ** to create a one-way frequency table. A vertical histogram is then generated using **VHSTP**. A horizontal histogram for the same data set can be found in the document example for the routine .

```
      USE UMACH_INT
      USE OWFRQ_INT
      USE VHSTP_INT

      IMPLICIT   NONE
      INTEGER    NBAR, NOBS
      PARAMETER  (NBAR=10, NOBS=30)
!
      INTEGER    IOPT, NOUT
      REAL       DIV(NBAR), TABLE(NBAR), X(NOBS), XHI, XLO
!
```

```
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37, &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!                                 Get output unit number
      CALL UMACH (2, NOUT)
!                                 Create a one-way frequency table from
!                                 a given data set using intervals of
!                                 equal length and user-supplied values
!                                 of XLO and XHI
      IOPT = 1
      XLO  = 0.5
      XHI  = 4.5
      CALL OWFRQ (X, NBAR, TABLE, IOPT=IOPT, XLO=XLO, XHI=XHI, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT (' Midpoints:  ', 10F6.2, /, '    Counts:  ', 10F6.0)
!                                 Create the horizontal histogram
      CALL VHSTP (TABLE, 'Plot of VHSTP')
      END
```

Output

```
 Midpoints:    0.25  0.75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75
    Counts:      2.    7.    6.    6.    4.    2.    2.    0.    0.    1.

                            Plot of VHSTP
Frequency-------------------------------------------------------
   7  *          I                                              *
   6  *          I    I    I                                    *
   5  *          I    I    I                                    *
   4  *          I    I    I    I                               *
   3  *          I    I    I    I                               *
   2  *     I    I    I    I    I    I    I                     *
   1  *     I    I    I    I    I    I    I              I      *
-----------------------------------------------------------------
Class       1    2    3    4    5    6    7    8    9   10
```

VHS2P

Prints a vertical histogram with every bar subdivided into two parts.

Required Arguments

FRQX — Vector of length NBAR. (Input)
FRQX contains the frequencies or counts, and the elements of FRQX must be nonnegative.

FRQY — Vector of length NBAR. (Input)
FRQY contains the second frequencies or counts, and the elements of FRQY must be nonnegative.

TITLE — CHARACTER string containing the title. (Input)

Optional Arguments

NBAR — Number of bars. (Input)
NBAR must be positive.
Default: NBAR = size (FRQX,1).

ISP —Spacing between histogram bars. (Input)
ISP = 0, 1 or 4 is allowed.
Default: ISP = 4.

FORTRAN 90 Interface

Generic: CALL VHS2P (FRQX, FRQY, TITLE [, ...])
Specific: The specific interface names are S_VHS2P and D_VHS2P.

FORTRAN 77 Interface

Single: CALL VHS2P (FRQX, FRQY, TITLE, NBAR, ISP)
Double: The double precision name is DVHS2P.

Description

The routine VHS2P prints a vertical histogram on one or more pages, using not more than 50 vertical and 100 horizontal print positions. Spacing control is allowed on the horizontal axis. Given two vectors containing positive counts, VHS2P determines the maximum count of the combined vectors T_{max}. Vertical printing position depends on K defined by $K = 1 + (T_{max} - 1)/50$. If a frequency is greater than K, then a character is printed on the first line. Henceforth, K is reduced by $K/50$ for each position, and frequencies are compared to the new K.

Comments

1. Workspace may be explicitly provided, if desired, by use of V2S2P/DV2S2P. The reference is:

   CALL V2S2P (NBAR, FRQX, FRQY, ISP, TITLE, WK)

   The additional argument is

   WK — Work vector of length 2 * NBAR.

2. Informational errors

Type	Code	Description
3	2	NBAR * (ISP + 1) is less than 1 or greater than 100. The width of the histogram is set to 100 and 100/(ISP + 1) bars are printed.
3	3	ISP as specified is not valid. The zero option is used.
3	4	TITLE is too long. TITLE was truncated from the right side.

3. If NBAR exceeds 100/(ISP + 1), then only 100/(ISP + 1) bars are printed.
4. If the maximum frequency is greater than 9999, the frequency column contains on some lines.
5. Output is written to the unit specified by the routine UMACH (see the Reference Material section of this manual).
6. TITLE is automatically centered and plot starts on a new page.

Example

Let X = FRQX contain 12 months of projected income figures and let Y = FRQY contain the actual income figures for the same 12 months. VHS2P produces a histogram that allows projected versus actual figures to be graphically compared.

```
      USE VHS2P_INT
      USE UMACH_INT

      IMPLICIT   NONE
```

```
      INTEGER    NBAR
      PARAMETER  (NBAR=12)
!
      INTEGER    ISP, NOUT
      REAL       FRQX(NBAR), FRQY(NBAR)
!
      DATA FRQX/11., 4., 4., 8., 4., 3., 10., 14., 4., 20., 4., 3./
      DATA FRQY/10., 6., 4., 12., 3., 4., 8., 18., 6., 18., 3., 7./
!
      CALL VHS2P (FRQX, FRQY, 'Plot of VHS2P')
!                                Get output unit number
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999)
99999 FORMAT (/, 3X, 'Twelve months projected sales versus actual ', &
             'sales, in thousands of dollars.', /, 11X, 'A positive ', &
             'sign (+) implies projected exceeded actual.', /, 11X, &
             'A negative sign (-) implies actual exceeded projected.')
!
      END
```

Output

```
                                Plot of VHS2P
Frequency-------------------------------------------------------------------
  20                                                    -
  19                                                    -
  18                                          +         I
  17                                          +         I
  16                                          +         I
  15                                          +         I
  14                                          I         I
  13                                          I         I
  12                    +                     I         I
  11       -            +                     I         I
  10       I            +              -      I         I
   9       I            +              -      I         I
   8       I            I              I      I         I
   7       I            I              I      I         I         +
   6       I    +       I              I      I    +    I         +
   5       I    +       I              I      I    +    I         +
   4       I    I    I  I    -    +    I      I    I    I    -    +
   3       I    I    I  I    I    I    I      I    I    I    I    I
   2       I    I    I  I    I    I    I      I    I    I    I    I
   1       I    I    I  I    I    I    I      I    I    I    I    I
-------------------------------------------------------------------------
Class      1    2    3    4    5    6    7    8    9   10   11   12

  Twelve months projected sales versus actual sales, in thousands of dollars.
           A positive sign (+) implies projected exceeded actual.
           A negative sign (-) implies actual exceeded projected.
```

HHSTP

Prints a horizontal histogram.

Required Arguments

FRQ —Vector of length NBAR containing the frequencies or counts. (Input)
Elements of FRQ must be nonnegative.

IBEG — Indicates the beginning setting of the plot. (Input)
If IBEG = 0, HHSTP skips to a new page before printing the first line. If IBEG ≠ 0, HHSTP skips two spaces and begins printing on the same page.

TITLE — CHARACTER string containing the title of the histogram. (Input)

Optional Arguments

NBAR — Number of bars. (Input)
NBAR must be positive.
Default: NBAR = size (FRQ,1).

ISPACE — Indicates spaces between horizontal histogram lines. (Input)
ISPACE = 0, 1, or 2 is allowed.
Default: ISPACE = 1.

LENGTH — Indicates the upper limit of the number of lines to print within the histogram per page. (Input)
After that number of lines is printed, the routine skips to a new page to continue printing. If LENGTH = 0; then the maximum number of lines coincides with the standard printer page, which is 60.
Default: LENGTH = 0.

IREP — Determines the repeating appearance for the class line (top) and frequency line (bottom) when multiple pages are required. (Input)
If IREP = 0, the class line and the frequency line are printed on the first and last page of the histogram, respectively. If IREP ≠ 0, both class and frequency line are printed on every page.
Default: IREP = 0.

IOPT — Page width option. (Input)
IOPT = 0 will cause a full (horizontal) page histogram. IOPT = 1 will limit the width to 80 columns. Default: IOPT = 1.

FORTRAN 90 Interface

Generic: CALL HHSTP (FRQ, IBEG, TITLE [, ...])
Specific: The specific interface names are S_HHSTP and D_HHSTP.

FORTRAN 77 Interface

Single: CALL HHSTP (NBAR, FRQ, IBEG, ISPACE, LENGTH, IREP, IOPT, TITLE)
Double: The double precision name is DHHSTP.

Description

The routine HHSTP prints a horizontal histogram on one or more pages. Given a vector containing frequencies or counts, HHSTP determines the maximum count T_{max}. Horizontal printing position depends on K defined by

$K = 1 + (T_{max} - 1)/60$ for 72 characters

$K = 1 + (T_{max} - 1)/120$ for 132 characters

If a frequency is greater than K, then a character is printed in the first position. Henceforth, K is increased by $K/60$ or $K/120$ for each position, and frequencies are compared to the resulting K.

Comments

Informational Errors

Type	Code	Description
3	3	ISPACE is not 0, 1, or 2. The zero option is used for ISPACE.
3	6	IOPT is not 0 or 1. The zero option is used for IOPT.
3	7	TITLE is too long and is truncated from the right side.

Example

Consider the data set in Example 1of the routine OWFRQ (see Chapter 1, "Basic Statistics"). We use the routine OWFRQ to create a one-way frequency table. A horizontal histogram is then generated using HHSTP. The user may find a vertical histogram for the same data set in the routine. Note that classes are listed from left to right in VHSTP.

```
      USE UMACH_INT
      USE OWFRQ_INT
      USE HHSTP_INT

      IMPLICIT   NONE
      INTEGER    NBAR, NOBS
      PARAMETER  (NBAR=10, NOBS=30)
!
      INTEGER    IBEG, IOPT, NOUT
      REAL       CLHW, DIV(NBAR), TABLE(NBAR), X(NOBS), XHI, XLO
!
      DATA X/0.77, 1.74, 0.81, 1.20, 1.95, 1.20, 0.47, 1.43, 3.37,  &
          2.20, 3.00, 3.09, 1.51, 2.10, 0.52, 1.62, 1.31, 0.32, 0.59, &
          0.81, 2.81, 1.87, 1.18, 1.35, 4.75, 2.48, 0.96, 1.89, 0.90, &
          2.05/
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Create a one-way frequency table from
!                                a given data set with intervals of
!                                equal length and user-supplied values
!                                of XLO and XHI
      IOPT = 1
      XLO  = 0.5
      XHI  = 4.5
      CALL OWFRQ (X, NBAR, TABLE, IOPT=IOPT, XLO=XLO, XHI=XHI, DIV=DIV)
      WRITE (NOUT,99999) DIV, TABLE
99999 FORMAT ('  Midpoints:  ', 10F6.2, /, '     Counts:  ', 10F6.0)
!                                Create the horizontal histogram
      IBEG   = 1
      IOPT   = 0
      CALL HHSTP (TABLE, IBEG, 'Histogram', IOPT=IOPT)
      END
```

Output

```
  Midpoints:     .25   .75  1.25  1.75  2.25  2.75  3.25  3.75  4.25  4.75
     Counts:      2.    7.    6.    6.    4.    2.    2.    0.    0.    1.
```

```
        Histogram
    Class ----------
      10  *I       *
          *        *
       9  *        *
          *        *
       8  *        *
          *        *
       7  *II      *
          *        *
       6  *II      *
          *        *
       5  *IIII    *
          *        *
       4  *IIIIII  *
          *        *
       3  *IIIIII  *
          *        *
       2  *IIIIIII *
          *        *
       1  *II      *
    -----------------
    Frequency  5
               One frequency unit is equal to 1 count unit(s).
```

SCTP

Prints a scatter plot of several groups of data.

Required Arguments

A —NOBS by NVAR matrix containing the data. (Input)

ICOL — Vector of length NVAR representing the nature of each column of matrix A. (Input)
The I-th column of A is the independent variable vector if ICOL(I) = 1. The I-th column of A is a dependent variable vector if ICOL(I) = 2. The I-th column of A is ignored otherwise.

RANGE — Vector of length four specifying minimum *x*, maximum *x*, minimum *y* and maximum *y*. (Input)
SCTP will calculate the range of the axis if the minimum of that range is greater than or equal to the maximum of that range.

SYMBOL — CHARACTER string of length NVAR. (Input)
SYMBOL (I : I) is the character used to plot the data set represented by column I. SYMBOL(I : I) is ignored if ICOL(I) ≠ 2.

XTITLE — CHARACTER string containing the *x*-axis title. (Input)

YTITLE — CHARACTER string containing the *y*-axis title. (Input)

TITLE — CHARACTER string containing the plot title. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: NOBS = size (A,1).

NVAR — Number of variables. (Input)
Default: NVAR = size (A,2).

LDA —Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

FORTRAN 90 Interface

Generic: CALL SCTP (A, ICOL, RANGE, SYMBOL, XTITLE, YTITLE, TITLE [, ...])

Specific: The specific interface names are **S_SCTP** and **D_SCTP**.

FORTRAN 77 Interface

Single: **CALL SCTP** (**NOBS**, **NVAR**, **A**, **LDA**, **ICOL**, **RANGE**, **SYMBOL**, **XTITLE**, **YTITLE**, **TITLE**)

Double: The double precision name is **DSCTP**.

Description

Routine **SCTP** prints a scatter plot of one variable on the *x*-axis against several variables on the *y*-axis. For multiple points, 2, 3, ..., 9 are used to denote the number of points at a location. The character "**M**" is used when the number of points is greater than 9. Any entry of the matrix **A** containing NaN (not a number) is ignored. See **AMACH** in "Machine-Dependent Constants".

Comments

1. Informational errors

Type	Code	Description
3	10	XTITLE is too long to fit into the page width determined by the routine PGOPT. XTITLE is truncated from the right side.
3	11	YTITLE is too long to fit into the page width determined by the routine PGOPT. YTITLE is truncated from the right side.
3	12	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

2. Integers 2, ..., 9 indicate two through nine points occupying the same plot position, respectively, and the character "**M**" indicates 10 or more multiple points. Consequently, it is recommended not to use any one of the above characters for **SYMBOL**.
3. One and only one column of **A** can be the independent variable vector.
4. A point is ignored if either the independent or the dependent variable contains NaN (not a number).
5. Output is written to the unit number specified by the routine **UMACH** (see the Reference Material section in this manual).
6. Default page width and length are 78 and 60, respectively. The user may change them by calling the routine **PGOPT** (see Chapter 19, "Utilities") in advance.

Example

This example prints a scatter plot of width against length for 150 iris petals. The routine **GDATA** (see Chapter 19, "Utilities") is used to retrieve the Fisher iris data.

```
      USE GDATA_INT
      USE SCTP_INT
      USE PGOPT_INT

      IMPLICIT   NONE
      INTEGER    ICOL(5), IDATA, IPAGE, LDA, NDA, NOBS, NVAR
      REAL       A(150,5), RANGE(4)
      CHARACTER  SYMBOL*5
!
      DATA ICOL/5*0/
      DATA RANGE/4*0./
      DATA SYMBOL/'    *'/
!
      IDATA  = 3
!                                Get Fisher Iris Data
      CALL GDATA (IDATA, A, NOBS, NVAR)
!                                Plot petal width against
!                                petal length
      ICOL(4) = 1
      ICOL(5) = 2
!                                Set page width and length
      IPAGE = 78
      CALL PGOPT (-1, IPAGE)
      IPAGE = 40
      CALL PGOPT (-2, IPAGE)

      CALL SCTP (A, ICOL, RANGE, SYMBOL, 'Petal length', &
                 'Petal width', 'Fisher Iris Data')
!
      END
```

Output

```
                                Fisher Iris Data
            :
            :                                          *  **
        2.4 -:                                      *    2
            :                                       ****  * * *       *
            :                                            * *       *
            :
            :                                         **** *      *
            :                                     ****           *  *
            :                                      *2 *       *
            :                                   32 *   2* * *  *
            :
            :                                  *    *
    P   1.6 -:                                 * *   *      *
    e       :                                *  5** 2**
    t       :                              *    2 *2*       *
    a       :
    l       :                           *   3222***
            :                              ** * *  *
```

```
    w          :                     *       **
    i          :                        2 2 *  **
    d          :
    t          :
    h     0.8 -:
               :
               :      *
               :
               :       *
               :   * 3** *
               :   23* *
               :* 24875* *
               :
               : *  22
          0.0 -:
               :.....................................................................
                :         :         :         :         :         :         :
                1.        2.        3.        4.        5.        6.        7.

                                       Petal length
```

BOXP

Prints boxplots for one or more samples.

Required Arguments

NI — Vector of length NGROUP. (Input)
NI(I) is the number of observations in the I-th group.

X —Vector of length NI(1) + NI(2) + ... + NI(NGROUP). (Input)
The first NI(1) positions contain the observations for the first group. The next NI(2) positions contain the observations for the second group, and so on.

TITLE — CHARACTER string containing the title of the plot. (Input)

Optional Arguments

NGROUP — The total number of groups of samples. (Input)
Default: NGROUP = size (NI,1).

FORTRAN 90 Interface

Generic: CALL BOXP (NI, X, TITLE [, ...])
Specific: The specific interface names are S_BOXP and D_BOXP.

FORTRAN 77 Interface

Single: CALL BOXP (NGROUP, NI, X, TITLE)
Double: The double precision name is DBOXP.

Description

BOXP prints NGROUP boxplots. The minimum and maximum of X are printed. The median of each data group is marked by "*" and the upper and lower hinges by "I". The "H-spread" is the distance between the upper and lower hinges. The observation farthest from the median that still remains within one step (1.5 H-spread) from each hinge also is marked by "+". The values in the second step (between 1.5 and 3 H-spreads from the hinges) are marked by the letter "O" and the values beyond the second step are marked by "X". If there are fewer than five

data points, each data point is plotted with an "X." If multiple data points occur at positions marked "X" or "0", the number of multiple points is noted. More information on boxplots can be found in Chapter 2 of Chambers et al. (1983).

Comments

1. Workspace may be explicitly provided, if desired, by use of B2XP/DB2XP. The reference is:

   CALL B2XP (NGROUP, NI, X, TITLE, WKSP)

   The additional argument is:

   WKSP — Workspace of length NI(1) + ... + NI(NGROUP). (Input)
   The first NI(1) positions contain the sorted data from the first NI(1) positions of X. The next NI(2) positions contain sorted data from the next NI(2) positions of X, and so on.

2. Informational error

Type	Code	Description
3	5	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

3. TITLE is centered and placed at the top of the plot. The plot starts on a new page and the default page width is 78. The user may change the width by calling the routine PGOPT (see Chapter 19, "Utilities") in advance.

Example

This example prints boxplots of three batches of data containing 5, 16 and 7 observations, respectively.

```
      USE PGOPT_INT
      USE BOXP_INT

      IMPLICIT   NONE
      INTEGER    IPAGE, NGROUP, I
      PARAMETER  (NGROUP=3)
!
      INTEGER    NI(NGROUP)
      REAL       X(28)
!
      DATA (NI(I),I=1,3)/5, 16, 7/
      DATA (X(I),I=1,5)/7., 9., 3., 1., 1./
      DATA (X(I),I=6,21)/25., 0., 1., 0., 5., 4., 3., 5., 5., 5., 5., &
          5., 5., 25., 15., 9./
      DATA (X(I),I=22,28)/10., 15., 20., 25., 2., 9., 12./
!                                Set page width.
      IPAGE = 70
      CALL PGOPT (-1, IPAGE)
      CALL BOXP (NI, X, 'Plot of BOXP')
!
```

```
END
```

Output

```
                              Plot of BOXP

      X    X          X    X
      2

              I--------I
    +---------I   *    I-----+              0                          X
              I--------I                                              2

                           I--------------------I
         +-----------------I     *              I------------------+
                           I--------------------I

    +.............................+.............................+
    0.0                          12.5                         25.0
```

STMLP

Prints a stem-and-leaf plot.

Required Arguments

X —Array of length **NOBS** containing the data. (Input)

UNITS — Size of the increment on the stem. (Input)
If **UNITS** is set so small that the length of the stem is more than 60 lines, **STMLP** will use a **UNITS** such that the stem will be no longer than 60 lines. However, if **UNITS** is a negative integer, **STMLP** will use the absolute value of **UNITS**, even if the stem would become very long. A common value for **UNITS** is 10.

TITLE — **CHARACTER** string containing the plot title. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: **NOB** = size (**X**,1).

FORTRAN 90 Interface

Generic: **CALL STMLP** (**X**, **UNITS**, **TITLE** [, ...])
Specific: The specific interface names are **S_STMLP** and **D_STMLP**.

FORTRAN 77 Interface

Single: **CALL STMLP** (**NOBS**, **X**, **UNITS**, **TITLE**)
Double: The double precision name is **DSTMLP**.

Description

Routine **STMLP** prints a stem-and-leaf display. The user can specify that the plot be longer than one page, but the default maximum is 60 lines. A plus sign (+) at the end of a line indicates that there are too many data points to fit within the width specifications. A scale marked in units of 10 is printed below the stem-and-leaf display.

Comments

1. Workspace may be explicitly provided, if desired, by use of S2MLP/DS2MLP. The reference is:

   CALL S2MLP (NOBS, X, UNITS, TITLE, MAXWID, IWK, WK)

   The additional arguments are as follows:

   MAXWID — Page width. (Input)
   MAXWID = 78 when STMLP is called.

   IWK — Work vector of length MAXWID.

   WK — Vector of length NOBS. (Output)
   WK contains the sorted data from X.

2. Informational error

Type	Code	Description
3	4	TITLE is too long to fit into the page width determined by the routine PGOPT. TITLE is truncated from the right side.

3. Default page width is 78. The user may change it by calling the routine PGOPT (see Chapter 19, "Utilities") in advance.

Example

This example prints a stem-and-leaf plot consisting of 27 data points ranging from –21.8 to 106.5.

```
      USE STMLP_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=27)
!
      REAL       UNITS, X(NOBS)
!
      DATA X/6.0, 106.5, 34.0, 88.1, 89.0, 0.3, 0.7, 4.0, 4.0, 5.0, &
          56.0, 62.8, 99.0, 4.0, 15.0, 76.0, 7.6, 101.5, 33.0, 91.0, &
          91.0, -6.3, -21.8, 0.0, 8.99, 5.5, 6.9/
!
      UNITS = 10.
      CALL STMLP (X, UNITS, 'Stem and leaf plot')
!
      END
```

Output

```
                            Stem and leaf plot

 Each line on the stem represents 1.0  unit(s).
 For example:  1 25
               2 2
```

```
    represents the data 12., 15., and 22.
          -2.2
          -1
          -0 6
           0 001444566789
           1 5
           2
           3 34
           4
           5 6
           6 3
           7 6
           8 89
           9 119
          10 27
```

CDFP

Prints a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.

Required Arguments

CDF — User-supplied `FUNCTION` to compute the cumulative distribution function. The form is `CDF(P)`, where
P — Sample point. (Input)
CDF — Theoretical probability at the point `P` or integral of the probability density function at the point `P`. (Output)

X —Vector of length `NOBS` containing the sample. (Input)

Optional Arguments

NOBS — Number of observations. (Input)
Default: `NOBS` = size (`X`,1).

N12 —Confidence band option. (Input)
If `N12` = 0, then no confidence bands are printed. If `N12` = 1, then positive or upper one-sided confidence band information is printed. If `N12` = –1, then negative or lower one-sided confidence band information is printed. If `N12` = 2, then two-sided confidence band information is printed.
Default: `N12` = 2.

N95 —Confidence band option. (Input)
If `N95` = 95, the 95-percent band is desired. Otherwise, the 99-percent band is desired.
Default: `N95` = 95.

IPRINT — Print option. (Input)
If `IPRINT` = 1, then `CDFP` prints the sample CDF, the theoretical CDF, and the confidence band on the CDF. If `IPRINT` = 0, then the above information will not be printed.
Default: `IPRINT` = 1.

FORTRAN 90 Interface

Generic: `CALL CDFP (CDF, X [, ...])`
Specific: The specific interface names are `S_CDFP` and `D_CDFP`.

FORTRAN 77 Interface

Single: CALL CDFP (CDF, NOBS, X, N12, N95, IPRINT)

Double: The double precision name is DCDFP.

Description

When IPRINT = 1, CDFP prints the sample cumulative distribution function (CDF), the theoretical CDF, and confidence bands on the CDF. The theoretical CDF will be plotted with or without the confidence band information. The sample CDF is calculated. The theoretical CDF is calculated by calling the user supplied FUNCTION subprogram CDF. Asymptotic critical values are used (from the Smirnov tables) for confidence interval calculations.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2FP/DC2FP. The reference is:

   CALL C2FP (CDF, NOBS, X, N12, N95, IPRINT, WKX, WK)

   The additional arguments are as follows:

   WKX — Vector of length NOBS containing the sorted data X in ascending order. (Output)

   WK — Vector of length 4 * NOBS containing confidence band values. (Output)
   WK may be dimensioned 3 * NOBS instead of 4 * NOBS for a lower or upper confidence band.

2. Note that sample CDFs are step functions.

3. Confidence bands are plotted around the sample CDF.

4. Output is written to the unit specified by the routine UMACH (see the Reference Material section in this manual).

5. Printing starts on a new page with default page width 78 columns and default page length 60 rows. The user may change these values by calling the routine PGOPT in advance.

Example

This example prints and plots the sample CDF, the theoretical CDF, and the two-sided 95 percent band information using 70 observations. Routines RNSET and RNUN are called to generate these uniform (0, 1) random numbers.

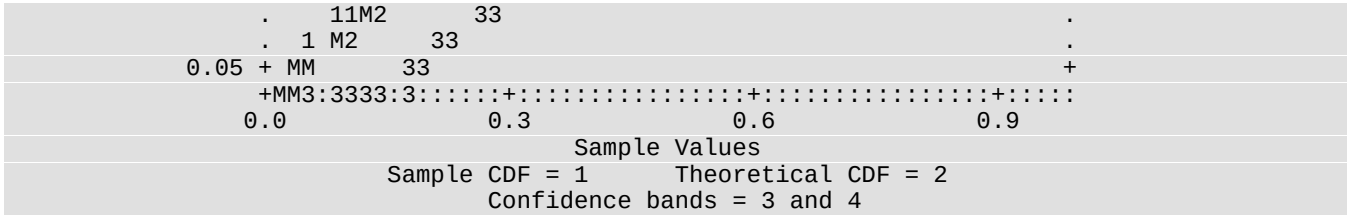
```
      USE PGOPT_INT
      USE RNSET_INT
      USE RNUN_INT
      USE CDFP_INT
```

```
      IMPLICIT   NONE
      INTEGER    IPAGE, ISEED, NOBS
      PARAMETER  (NOBS=70)
      REAL       CDF, X(NOBS)
      EXTERNAL   CDF
!
      ISEED = 123457
!                                Two-sided confidence band option.
!                                95-percent band option.
!                                Set page width and length.
      IPAGE =  78
      CALL PGOPT (-1, IPAGE)
      IPAGE=40
      CALL PGOPT (-2, IPAGE)
!                                Initialize the seed.
      CALL RNSET (ISEED)
!                                Generate pseudo-random numbers from
!                                a uniform (0,1) distribution.
      CALL RNUN (X)
!                                Plot
      CALL CDFP (CDF, X, IPRINT=0)
      END
!
      REAL FUNCTION CDF (X)
      REAL       X
!
      CDF = X
      RETURN
      END
```

Output

```
                        Cumulative Sample and Theoretical CDFs

               +:::::::::::::::::+:::::::::::::::::+:::::::::::::::4444:4MM
          0.95 +                                                    44    M2+
               .                                                   44  MM   .
               .                                                44 4  2M1    .
               .                                              44 4   22 1    33
          0.80 +                                             44     2 11    33+
               .                                           4 4   22   1  33 3 .
               .                                        4    22  1 11  33    .
     P         .                                     44    2 1 1      3     .
     r    0.65 +                               4 4    2  1         33     +
     o         .                             4     22 1 1         3       .
     b         .                           4    22 1 1         33        .
     a         .                        44  4   2  11         33          .
     b    0.50 +                    4 4    22   11      33               +
     i         .                 4  4      221 1         33               .
     l         .             4 4            11      3 3                  .
     i         .         444              2M  11      33                 .
     t    0.35 +       4 4           2 1          33                     +
     y         .      44        1  M1        3 3 3                      .
               .    44       1 MM           33                          .
               .  4        112             33                          .
          0.20 + 4        122           33                              +
               .4       M 2         33 3                                 .
```


. 11M2 33 .
. 1 M2 33 .
0.05 + MM 33 +
+MM3:3333:3:::::+:::::::::::::::+:::::::::::::::+:::::
0.0 0.3 0.6 0.9
Sample Values
Sample CDF = 1 Theoretical CDF = 2
Confidence bands = 3 and 4

CDF2P

Prints a plot of two sample cumulative distribution functions.

Required Arguments

NOBS1 — Size of sample one. (Input)

NOBS2 — Size of sample two. (Input)

X — Vector of length NOBS1 + NOBS2. (Input)
X contains sample one followed by sample two.

FORTRAN 90 Interface

Generic: CALL CDF2P (NOBS1, NOBS2, X)
Specific: The specific interface names are S_CDF2P and D_CDF2P.

FORTRAN 77 Interface

Single: CALL CDF2P (NOBS1, NOBS2, X)
Double: The double precision name is DCDF2P.

Description

Routine CDF2P plots two sample cumulative probability distribution functions (CDFs). Two samples are first merged and then sorted. The cumulative distribution functions are then calculated. On the plots, the characters "1" and "2" indicate the first and second samples, respectively, and the character "M" indicates multiple points.

Comments

1. Workspace may be explicitly provided, if desired, by use of C2F2P/DC2F2P. The reference is:

   ```
   CALL C2F2P (NOBS1, NOBS2, X, WK, IWK)
   ```

   The additional arguments are as follows:

   WK — Work vector of length 3 * (NOBS1 + NOBS2).
   IWK — Work vector of length NOBS1 + NOBS2.

2. Printing starts on a new page with default page width 78 and default page length 60. The user may change page width and length by calling the routine **PGOPT** in advance.

Example

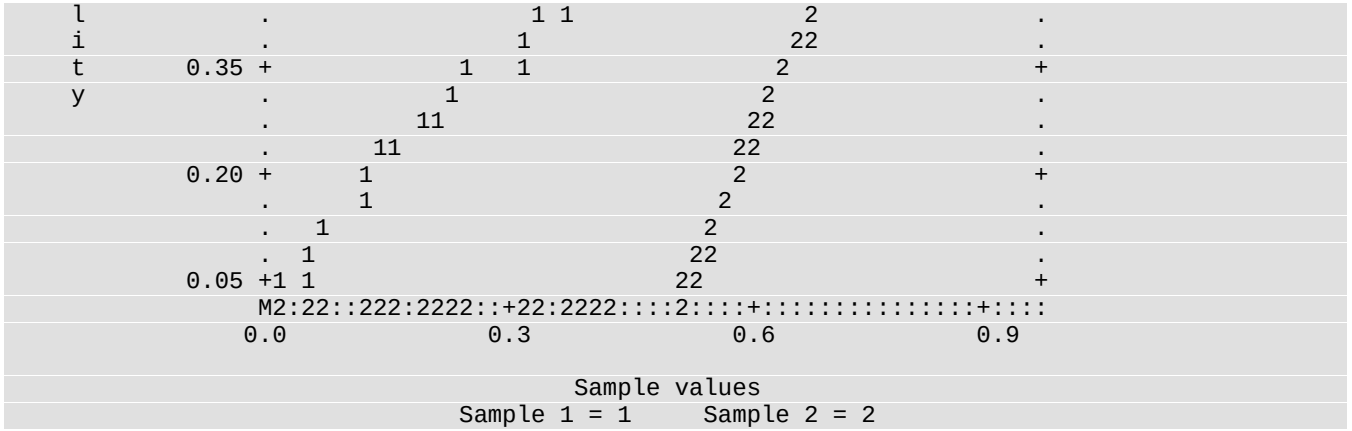
The first sample consists of pseudo-random numbers from a uniform (0, 1) distribution. Routines **RNSET** and **RNUN** (see Chapter 18, "Random Number Generation") are used to generate this sample. The second sample consists of points of the standard normal (Gaussian) distribution function generated by the routine **ANORDF** (see Chapter 17, "Probability Distribution Functions and Inverses").

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    I, IPAGE
      REAL       VAL, X(100)
!                                Initialize the seed.
      CALL RNSET (1234567)
!                                Generate pseudo-random numbers from
!                                a uniform (0,1) distribution.
      CALL RNUN (X)
!                                Second sample consists of 50 points of
!                                the std normal distribution function.
      VAL = 0.
      DO 10  I=1, 50
         VAL     = VAL + .02
         X(I+50) = ANORDF(VAL)
   10 CONTINUE
!                                Set page width and length.
      IPAGE =  78
      CALL PGOPT (-1, IPAGE)
      IPAGE =  40
      CALL PGOPT (-2, IPAGE)
      CALL CDF2P (50, 50, X)
      END
```

Output

```
                    Cumulative Sample Distribution Functions
             +:::::::::::::::::+:::::::::::::::+:::::::::::22:2+22:M
        0.95 +                                              2     1 +
             .                                             22  1 1  .
             .                                             2 1 1    .
             .                                            2111      .
        0.80 +                                            1         +
             .                                           M2         .
             .                                         11M2         .
P            .                                        11 2          .
r       0.65 +                                      111 2           +
o            .                                  11111   22          .
b            .                        1    11111      22          .
a            .                      1               2             .
b       0.50 +                     1               2              +
i            .                   11              22               .
```


l
i
t
y
0.35
0.20
0.05
0.0
0.3
0.6
0.9
Sample values
Sample 1 = 1 Sample 2 = 2

PROBP

Prints a probability plot.

Required Arguments

NOBS — Total number of observations in uncensored sample. (Input)

N1 — The rank number of the smallest observation in the sample X, if ranked in the complete sample. (Input)
In other words, the number of observations that have been censored from below is N1 - 1.

N2 — The rank number of the largest observation in the sample X, if ranked in the complete sample. (Input)
In other words, the number of observations that have been censored from above is NOBS - N2.

X — Vector of length N2 - N1 + 1. (Input)
X contains the data, possibly a censored data set from a complete sample of size NOBS.

IDIST — Distribution option. (Input)

IDIST = 1, normal distribution.

IDIST = 2, lognormal distribution.

IDIST = 3, half-normal distribution.

IDIST = 4, exponential distribution.

IDIST = 5, Weibull distribution.

IDIST = 6, extreme value distribution.

FORTRAN 90 Interface

Generic: CALL PROBP (NOBS, N1, N2, X, IDIST)
Specific: The specific interface names are S_PROBP and D_PROBP.

FORTRAN 77 Interface

Single: CALL PROBP (NOBS, N1, N2, X, IDIST)
Double: The double precision name is DPROBP.

Description

Routine **PROBP** sorts a data set and plots the observed values along the vertical axis and the ranks along the horizontal axis. In the case of the lognormal and Weibull distributions, the vertical axis has a log scale. The horizontal axis has the appropriate cumulative distribution function scale. Let *M* = **NOBS** denote the total number of observations in an uncensored sample. For normal and lognormal distributions, the horizontal plotting distance for the observation with rank *I* (out of *M*) is proportional to the inverse normal cumulative distribution function evaluated at (3 * **I** - 1)/(3 * *M* + 1). For the half-normal plot, the corresponding horizontal distance is proportional to the inverse normal cumulative distribution function evaluated at (3 * *M* + 3 * *I* - 1)/(6 * *M* + 1). For other plots, the horizontal distances are proportional to the respective inverse cumulative distribution functions evaluated at (*I* - .5)/*M*.

Let N_1 = **N1** and N_2 = **N2**. In **PROBP** it is assumed that the N_1 - 1 smallest observations and the *M* - N_2 largest observations have been censored. If there has been no censoring, N_1 should be set to 1 and N_2 set to *M*. The smallest observation is plotted against the expected value (or the approximated expected value) of the N_1-th order statistic from a sample of size *M*; the next smallest observation is plotted as if it were the (N_1 + 1)-th sample order statistic, and so on.

PROBP does not do any shifting of location of the observation in the data set. If any observations fall outside of the range of the distribution (that is, if any observations are nonpositive when the distribution specified is lognormal or Weibull), those observations are censored and N_1 or N_1 is modified to reflect the number censored. In this case an error message of type 3 is generated. A plot which is a straight line provides evidence that the sample is from the distribution specified.

Comments

1. Workspace may be explicitly provided, if desired, by use of **P2OBP/DP2OBP**. The reference is:

   ```
   CALL P2OBP (NOBS, N1, N2, X, IDIST, M1, M2, WK)
   ```

   The additional arguments are as follows:

   M1 — Rank of the smallest observation actually used. (Output)

   M2 — Rank of the largest observation actually used. (output)

   WK — Work space of length 2 * **NOBS**.

2. Informational error

Type	Code	Description
3	7	It is necessary to delete some items from the plotting because those items do not satisfy properties of the distribution.

3. NOBS must be greater than or equal to N2 - N1 + 1. If there is no censoring, then N1 = 1 and N2 = NOBS.

4. Output is written to the unit specified by the routine UMACH (see the Reference Material section in this manual).

5. Printing starts on a new page with default page width 78. The user may change it by calling the routine PGOPT (see Chapter 19, "Utilities") in advance.

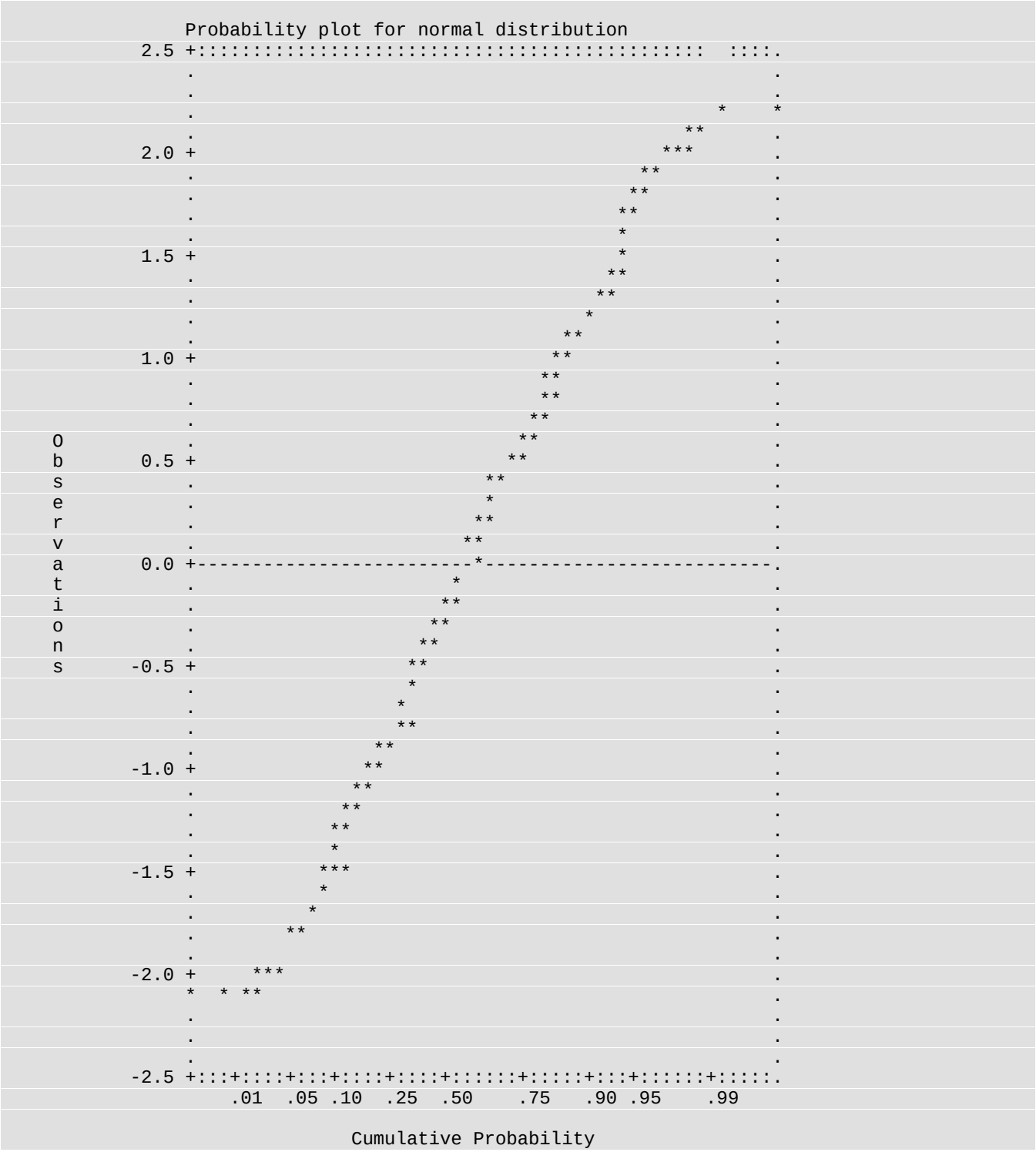
Example

In this example, a sample of size 250 (artificially generated from a normal distribution by routines RNSET and RNNOR in Chapter 18, "Random Number Generation") is plotted by PROBP against a normal distribution function. The generally straight line produced is an indication that the sample is from a normal distribution.

```
      USE RNSET_INT
      USE RNNOR_INT
      USE PROBP_INT

      IMPLICIT   NONE
      INTEGER    NOBS
      PARAMETER  (NOBS=250)
!
      INTEGER    IDIST, N1, N2
      REAL       X(NOBS)
!
      IDIST = 1
!                                No censoring
      N1 = 1
      N2 = 250
!                                Initialize the seed
      CALL RNSET (123457)
      CALL RNNOR (X)
!
      CALL PROBP (NOBS, N1, N2, X, IDIST)
      END
```

Output


Probability plot for normal distribution
Observations
Cumulative Probability
2.5
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
.01 .05 .10 .25 .50 .75 .90 .95 .99

PLOTP

Prints a plot of up to 10 sets of points.

Required Arguments

X — Vector of length NDATA containing the values of the independent variable. (Input)

A — Matrix of dimension NDATA by NFUN containing the NFUN sets of dependent variable values. (Input)

SYMBOL — CHARACTER string of length NFUN. (Input)
SYMBOL (I : I) is the symbol used to plot function I.

XTITLE — CHARACTER string used to label the *x*-axis. (Input)

YTITLE — CHARACTER string used to label the *y*-axis. (Input)

TITLE — CHARACTER string used to label the plot. (Input)

Optional Arguments

NDATA — Number of independent variable data points. (Input)
Default: NDATA = size (X,1).

NFUN — Number of sets of points. (Input)
NFUN must be less than or equal to 10.
Default: NFUN = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

INC — Increment between elements of the data to be used. (Input)
PLOTP plots X(1 + (I – 1) * INC) for I = 1, 2, ..., NDATA.
Default: INC = 1.

RANGE — Vector of length four specifying minimum *x*, maximum *x*, minimum *y* and maximum *y*. (Input)
PLOTP will calculate the range of the axis if the minimum and maximum of that range are equal.
Default: RANGE = 1.0.

FORTRAN 90 Interface

Generic: CALL PLOTP (X, A, SYMBOL, XTITLE, YTITLE, TITLE [, ...])

Specific: The specific interface names are S_PLOTP and D_PLOTP.

FORTRAN 77 Interface

Single: CALL PLOTP (NDATA, NFUN, X, A, LDA, INC, RANGE, SYMBOL, XTITLE, YTITLE, TITLE)

Double: The double precision name is DPLOTP.

Description

Routine PLOTP produces a line printer plot of up to ten sets of points superimposed upon the same plot. A character "M" is printed to indicate multiple points. The user may specify the *x* and *y*-axis plot ranges and plotting symbols. Plot width and length may be reset in advance by calling PGOPT.

Comments

1. Informational errors

Type	Code	Description
3	7	NFUN is greater than 10. Only the first 10 functions are plotted.
3	8	TITLE is too long. TITLE is truncated from the right side.
3	9	YTITLE is too long. YTITLE is truncated from the right side.
3	10	XTITLE is too long. XTITLE is truncated from the right side. The maximum number of characters allowed depends on the page width and the page length. See Comment 5 below for more information.

2. YTITLE and TITLE are automatically centered.
3. For multiple plots, the character M is used if the same print position is shared by two or more data sets.
4. Output is written to the unit specified by routine UMACH (see the "Reference Material").

Default page width is 78 and default page length is 60. They may be changed by calling PGOPT in advance.

Example

This example plots the sine and cosine functions from – 3.5 to + 3.5 and sets page width and length to 78 and 40, respectively, by calling PGOPT (see Chapter 19, "Utilities" in advance.

```
      USE PGOPT_INT
      USE PLOTP_INT

      IMPLICIT   NONE
      INTEGER    I, IPAGE
      REAL       A(200,2), DELX, PI, RANGE(4), X(200)
      CHARACTER  SYMBOL*2
      INTRINSIC  COS, SIN
!
      DATA SYMBOL/'SC'/
      DATA RANGE/-3.5, 3.5, -1.2, 1.2/
!
      PI     = 3.14159
      DELX   = 2.*PI/199.
      DO 10  I= 1, 200
         X(I)   = -PI + FLOAT(I-1) * DELX
         A(I,1) = SIN(X(I))
         A(I,2) = COS(X(I))
   10 CONTINUE
!                                Set page width and length
      IPAGE =  78
      CALL PGOPT (-1, IPAGE)
      IPAGE =  40
      CALL PGOPT (-2, IPAGE)
      CALL PLOTP (X, A, SYMBOL,'X AXIS', 'Y AXIS', 'C = COS, S = SIN')
!
      END
```

Output

```
                                C = COS,   S = SIN
           1.2 ::::+:::::::::::::::+:::::::::::::::+:::::::::::::::+::::
               .                          I                           .
               .                          I                           .
               .                       CCCCCCC     SSSSSSSS           .
               .                     CC  I  CC   SS      SS           .
           0.8 +                    C    I    C SS        SS          +
               .                   C     I     MS          SS         .
               .                  C      I    SSC           SS        .
               .                 CC      I   SS CC           SS       .
               .                CC       I   S   CC           S       .
           0.4 +                C        I  S     C           S      +
               .               C         I SS      C           SS     .
     Y         .              CC         I S       CC           S     .
               .              C          IS         C           S     .
     A         .             C           SS          C           SS   .
     X    0.0 +--S-----------CC-----------S-----------CC-----------S--+
     I         .  SS         CC           SS           CC            .
     S         .   S         C            SI            C            .
               .    S       CC           S I            CC           .
               .    SS      C           SS I             C           .
          -0.4 +     S     C            S  I              C          +
```

```
                .      S   CC           S   I               CC          .
                .      SS CC          SS   I                CC         .
                .       SSC           SS   I                 C         .
                .        MS           SS   I                 C         .
           -0.8 +       C SS          SS   I                  C        +
                .     CC   SS       SS     I                  CC       .
                .  CCCC     SSSSSSSS       I                   CCCC    .
                .  C                       I                      C    .
                .                          I                           .
           -1.2 ::::+:::::::::::::::+:::::::::::::::+:::::::::::::::+::::
                   -3              -1               1               3
                                    X AXIS
```

TREEP

Prints a binary tree.

Required Arguments

ICLSON — Vector of length NODE – 1 containing the left son nodes. (Input)
Node number NODE + K has left son given by ICLSON(K) for K = 1, ..., NODE - 1.

ICRSON — Vector of length NODE – 1 containing the right son nodes. (Input)
Node number NODE + K has right son given by ICRSON(K) for K = 1, ..., NODE - 1.

CLEVEL — Vector of length NODE - 1 containing the level used in merging or splitting the son nodes. (Input)
CLEVEL(K) specifies the scale to be used on the vertical (IMETH = 1 or 2) or horizontal (IMETH = 3) axis for node NODE + K, for K = 1, 2, ..., NODE - 1.

NSCALE — Number of horizontal slices of tree. (Input)
NSCALE must be positive.

SCALE — Vector of length two giving the interval on the CLEVEL axis which should be used to plot the tree. (Input)
SCALE(1) is the location for printing the terminal nodes. The root node is printed at SCALE(2).

NODENM — CHARACTER*(*) vector of length NODE containing the terminal node labels. (Input)
If terminal node labels are to be 1, 2, 3, ..., then NODENM(1) should be "DEFAULT" and the remaining elements of NODENM are not used. The length of each label is M, where M is determined by the user.

Optional Arguments

NODE — Initial number of observations or nodes. (Input)
NODE must be greater than 2.
Default: NODE = size (ICLSON,1) + 1.

IMETH — Method to be used for printing the binary tree. (Input)
Default: IMETH = 1.

IMETH	Method
1	Horizontal tree

IMETH	Method
2	Horizontal *I*-tree
3	Vertical tree

IROOT — Subtree specification. (Input)
IROOT specifies the root node of the subtree to be printed. If IROOT = 2 * NODE - 1 (or zero for the default), the entire tree is printed. IROOT must be in the range NODE + 1 to 2 * NODE - 1.
Default: IROOT = 0.

NFILL — The number of filler lines printed between horizontal or vertical node lines. (Input)
NFILL = 1 is usually sufficient. NFILL must be nonnegative.
Default: NFILL = 1.

FORTRAN 90 Interface

Generic: CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM [, ...])
Specific: The specific interface names are S_TREEP and D_TREEP.

FORTRAN 77 Interface

Single: CALL TREEP (NODE, ICLSON, ICRSON, IMETH, CLEVEL, IROOT, NSCALE, NFILL, SCALE, NODENM)
Double: The double precision name is DTREEP.

Description

Routine TREEP prints a binary tree which may represent results of hierarchical clustering algorithm such as the routine CLINK.

Let M = NODE indicate the number of nodes. A binary tree is composed of M terminal nodes and M - 1 nonterminal nodes uniquely numbered 1 to M and M + 1 to M + (M - 1), respectively. Each nonterminal node joins together two son nodes which may or may not be terminal. Nonterminal nodes $M + K$ are printed on the vertical scale interval $[S_1, S_2]$ at the level given in C_K, for K = 1, 2, ..., M - 1, where S_1 = SCALE(1), S_2 = SCALE(2), and C_K = CLEVEL(K).

Comments

1. Workspace may be explicitly provided, if desired, by use of T2EEP/DT2EEP. The reference is:

CALL T2EEP (NODE, ICLSON, ICRSON, IMETH, CLEVEL, IROOT, NSCALE, NFILL, SCALE, NODENM, IDTREE, ISTREE, IOTREE, INTREE, TLTREE)

The additional arguments are as follows:

IDTREE — Work vector of length IROOT. IDTREE is used to store the distance of each node from the vertical axis in vertical tree.

ISTREE — Work vector of length IROOT used to store all the nodes. IROOT is the first element of the array.

IOTREE — Work vector of length IROOT + 1 used to store the index of each node as TLTREE is sorted.

INTREE — Work vector of length IROOT.

TLTREE — Work vector of length IROOT + 1 used to store the level of each node in descending order in a vertical tree. It is used to store the distance of each node from the top of the horizontal line in ascending order in a horizontal tree.

2. Printing starts on a new page with default page width 78. The user may change it by calling the routine PGOPT in advance.

Example

```
      USE PGOPT_INT
      USE TREEP_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NODE
      PARAMETER  (NODE=5)
!
      INTEGER    ICLSON(NODE-1), ICRSON(NODE-1), IMETH, IPAGE, NOUT, NSCALE
      REAL       CLEVEL(NODE-1), SCALE(2)
      CHARACTER  NODENM(NODE)*7
!
      DATA ICLSON/5, 6, 4, 7/
      DATA ICRSON/3, 1, 2, 8/
      DATA NODENM/'DEFAULT', ' ', ' ', ' ', ' '/
      DATA CLEVEL/1., 2., 3., 4./
      DATA SCALE/0., 5./
!                                Set page width
      IPAGE = 70
      CALL PGOPT (-1, IPAGE)
      NSCALE = 1
!                                Horizontal tree
      IMETH = 1
      CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM, &
                  IMETH=IMETH)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999)
99999 FORMAT (1X, //////)
!                                Horizontal I-tree
      IMETH = 2
      CALL TREEP (ICLSON, ICRSON, CLEVEL, NSCALE, SCALE, NODENM, &
                  IMETH=IMETH)
```

```
    !
          END
```

Output

```
    Similarity range from   0.   to    5.000000
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       5*************
                    *
                    6*************
                    *            *
       3*************            *
                                 *
                                 7*************************
                                 *                         *
       1************************                           *
                                                           *
                                                           9**************
                                                           *
       4************************************              *
                                           *               *
                                           8*************
                                           *
       2************************************
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


    Similarity range from   0.   to    5.000000
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       5************6************7*************************9**************
                    *            *                         *
       3*************            *                         *
                                 *                         *
       1************************                           *
                                                           *
       4************************************8**************
                                           *
       2************************************
        +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Probability Distribution Functions

Routines

Usage Notes

Comments

Definitions and discussions of the terms basic to this chapter can be found in Johnson and Kotz (1969, 1970a, 1970b). These are also good references for the specific distributions.

In order to keep the calling sequences simple, whenever possible, the subprograms described in this chapter are written for standard forms of statistical distributions. Hence, the number of parameters for any given distribution may be fewer than the number often associated with the distribution. For example, while a gamma distribution is often characterized by two parameters (or even a third, "location"), there is only one parameter that is necessary, the "shape". The "scale" parameter can be used to scale the variable to the standard gamma distribution. Also, the functions relating to the normal distribution, `ANORDF` and `ANORIN`, are for a normal distribution with mean equal to zero and variance equal to one. For other means and variances, it is very easy for the user to standardize the variables by subtracting the mean and dividing by the square root of the variance.

The *cumulative distribution function* for the (real, single-valued) random variable *X* is the function *F* defined for all real *x* by

$$F(x) = \text{Prob}(X \leq x)$$

where Prob(•) denotes the probability of an event. The distribution function is often called the *cumulative distribution function* (CDF).

For distributions with finite ranges, such as the beta distribution, the CDF is 0 for values less than the left endpoint and 1 for values greater than the right endpoint. The subprograms described in this chapter return the correct values for the distribution functions when values outside of the range of the random variable are input, but warning error conditions are set in these cases.

Discrete Random Variables

For discrete distributions, the function giving the probability that the random variable takes on specific values is called the *probability function*, defined by

$$p(x) = \text{Prob}(X = x)$$

The "**PR**" routines described in this chapter evaluate probability functions.

The CDF for a discrete random variable is

$$F(x) = \sum_{A} p(k)$$

where *A* is the set such that $k \leq x$. The "**DF**" routines in this chapter evaluate cumulative distribution functions. Since the distribution function is a step function, its inverse does not exist uniquely.

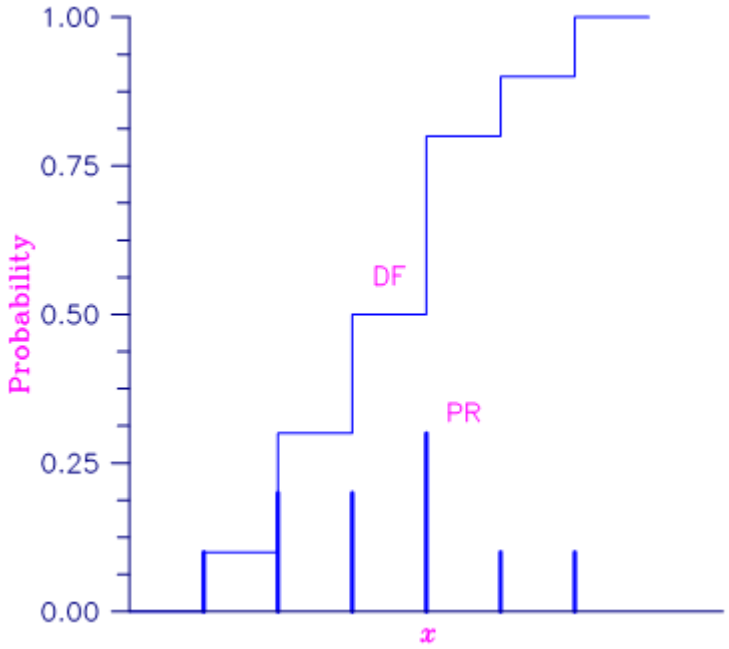


Figure 22, Discrete Random Variable

In the plot above, a routine like `BINPR` in this chapter evaluates the individual probability, given **X**. A routine like `BINDF`would evaluate the sum of the probabilities up to and including the probability at **X**.

Continuous Distributions

For continuous distributions, a probability function, as defined above, would not be useful because the probability of any given point is 0. For such distributions, the useful analog is the *probability density function* (PDF). The integral of the PDF is the probability over the interval, if the continuous random variable *X* has PDF *f*, then

$$\text{Prob}(a < X \leq b) = \int_{a}^{b} f(x)dx$$

The relationship between the CDF and the PDF is

$$F(x) = \int_{-\infty}^{x} f(t)dt$$

as shown in Figure 17-2.

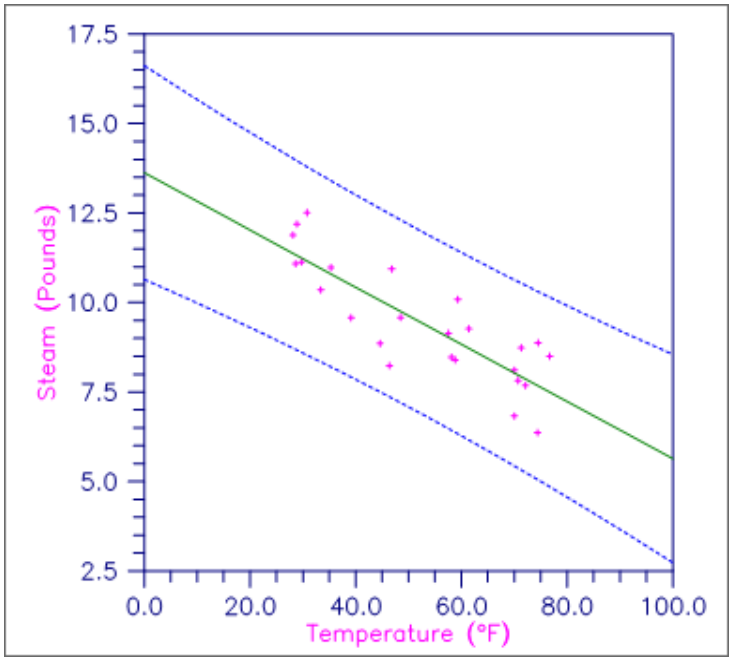


Figure 23, Probability Density Function

The "**DF**" routines described in this chapter evaluate cumulative distribution functions.

For (absolutely) continuous distributions, the value of *F*(*x*) uniquely determines *x* within the support of the distribution. The "**IN**" routines described in this chapter compute the inverses of the cumulative distribution functions, that is, given *P* =*F*(*x*) (called "**P**" for "probability"), a routine such as `BETIN` computes *x*. The inverses are defined only over the open interval (0,1).

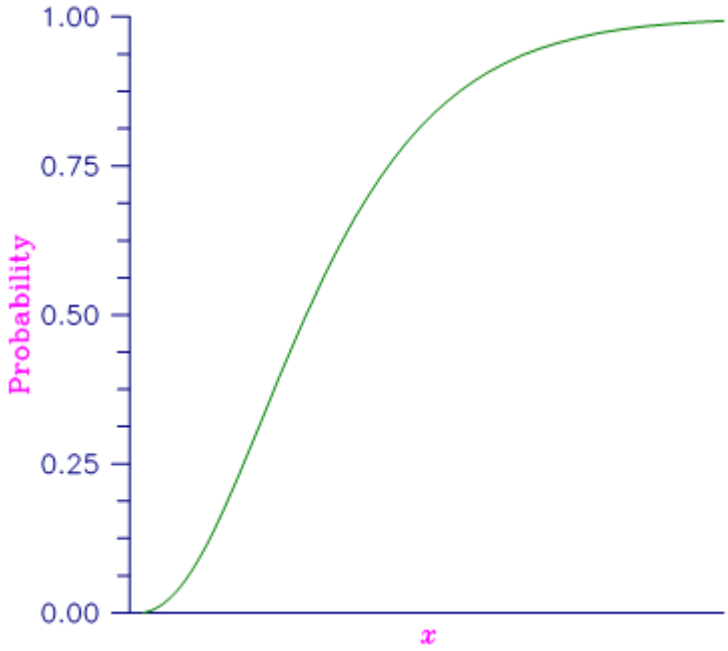


Figure 24, Cumulative Probability Distribution Function

There are three routines described in this chapter that deal with general continuous distribution functions. The routine **GCDF** computes a distribution function using values of the density function, and the routine **GCIN** computes the inverse. These two routines may be useful when the user has an estimate of a probability density, as perhaps computed by the routine **DESPL** or **DESKN** (see Chapter 15: "Density and Hazard Estimation"), or computed from a frequency polygon. The routine **GFNIN** computes the inverse of a distribution function that is specified as a FORTRAN function.

Parameter Estimation

A related task to evaluating a probability density or distribution function is to estimate the values of its parameters. For many of the distributions covered in this chapter, routine **MLE** provides maximum likelihood estimates of the unknown parameter values given a sample of observations.

Additional Comments

Whenever a probability close to 1.0 results from a call to a distribution function or is to be input to an inverse function, it is often impossible to achieve good accuracy because of the nature of the representation of numeric values. In this case, it may be better to work with the complementary distribution function (one minus the distribution function). If the distribution is symmetric about some point (as the normal distribution, for example) or is

reflective about some point (as the beta distribution, for example), the complementary distribution function has a simple relationship with the distribution function. For example, to evaluate the standard normal distribution at 4.0, using ANORIN directly, the result to six places is 0.999968. Only two of those digits are really useful, however. A more useful result may be 1.000000 minus this value, which can be obtained to six significant figures as 3.16713E-05 by evaluating ANORIN at -4.0. For the normal distribution, the two values are related by $\Phi(x) = 1 - \Phi(-x)$, where $\Phi(\cdot)$ is the normal distribution function. Another example is the beta distribution with parameters 2 and 10. This distribution is skewed to the right, so evaluating BETDF at 0.7, we obtain 0.999953. A more precise result is obtained by evaluating BETDF with parameters 10 and 2 at 0.3. This yields 4.72392E-5. (In both of these examples, it is wise not to trust the last digit.)

Many of the algorithms used by routines in this chapter are discussed by Abramowitz and Stegun (1964). The algorithms make use of various expansions and recursive relationships and often use different methods in different regions.

Cumulative distribution functions are defined for all real arguments, however, if the input to one of the distribution functions in this chapter is outside the range of the random variable, an error of Type 1 is issued, and the output is set to zero or one, as appropriate. A Type 1 error is of lowest severity, a "note", and, by default, no printing or stopping of the program occurs. The other common errors that occur in the routines of this chapter are Type 2, "alert", for a function value being set to zero due to underflow, Type 3, "warning", for considerable loss of accuracy in the result returned, and Type 5, "terminal", for incorrect and/or inconsistent input, complete loss of accuracy in the result returned, or inability to represent the result (because of overflow). When a Type 5 error occurs, the result is set to NaN (not a number, also used as a missing value code, obtained by routine AMACH(6). (See the section "User Errors" in the Reference Material.)

BINDF

This function evaluates the binomial cumulative distribution function.

Function Return Value

BINDF — Function value, the probability that a binomial random variable takes a value less than or equal to K. (Output)
BINDF is the probability that K or fewer successes occur in N independent Bernoulli trials, each of which has a PIN probability of success.

Required Arguments

K — Argument for which the binomial distribution function is to be evaluated. (Input)

N — Number of Bernoulli trials. (Input)

PIN — Probability of success on each independent trial. (Input)

FORTRAN 90 Interface

Generic: BINDF (K, N, PIN)
Specific: The specific interface names are S_BINDF and D_BINDF.

FORTRAN 77 Interface

Single: BINDF (K, N, PIN)
Double: The double precision name is DBINDF.

Description

Function BINDF evaluates the cumulative distribution function of a binomial random variable with parameters n and p where n = N and p = PIN. It does this by summing probabilities of the random variable taking on the specific values in its range. These probabilities are computed by the recursive relationship

$$\Pr(X = j) = \frac{(n + 1 - j)p}{j(1 - p)}\Pr(X = j - 1)$$

To avoid the possibility of underflow, the probabilities are computed forward from 0, if k is not greater than n times p, and are computed backward from n, otherwise. The smallest positive machine number, **ε**, is used as the starting value for summing the probabilities, which are rescaled by $(1 - p)^n$**ε** if forward computation is performed and by p^n**ε** if backward computation is done. For the special case of p = 0, **BINDF** is set to 1; and for the case p = 1, **BINDF** is set to 1 if $k = n$ and to 0 otherwise.

Comments

Informational Errors

Type	Code	Description
1	3	The input argument, K, is less than zero.
1	4	The input argument, K, is greater than the number of Bernoulli trials, N.

Example

Suppose X is a binomial random variable with n = 5 and p = 0.95. In this example, we find the probability that X is less than or equal to 3.

```
      USE UMACH_INT
      USE BINDF_INT

      IMPLICIT   NONE
      INTEGER    K, N, NOUT
      REAL       PIN, PR
!
      CALL UMACH (2, NOUT)
      K  = 3
      N  = 5
      PIN  = 0.95
      PR = BINDF(K,N, PIN)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is less than or equal to 3 is ' &
            , F6.4)
      END
```

Output

```
 The probability that X is less than or equal to 3 is 0.0226
```

BINPR

This function evaluates the binomial probability density function.

Function Return Value

BINPR Function value, the probability that a binomial random variable takes a value equal to **K**. (Output)

Required Arguments

K — Argument for which the binomial probability function is to be evaluated. (Input)

N — Number of Bernoulli trials. (Input)

PIN — Probability of success on each independent trial. (Input)

FORTRAN 90 Interface

Generic: **BINPR** (**K**, **N**, **PIN**)

Specific: The specific interface names are **S_BINPR** and **D_BINPR**.

FORTRAN 77 Interface

Single: **BINPR** (**K**, **N**, **PIN**)

Double: The double precision name is **DBINPR**.

Description

The function **BINPR** evaluates the probability that a binomial random variable with parameters *n* and *p* where *p* =**PIN** takes on the value *k*. It does this by computing probabilities of the random variable taking on the values in its range less than (or the values greater than) *k*. These probabilities are computed by the recursive relationship

$$\Pr(X = j) = \frac{(n + 1 - j)p}{j(1 - p)} \Pr(X = j - 1)$$

To avoid the possibility of underflow, the probabilities are computed forward from 0, if *k* is not greater than *n* times *p*, and are computed backward from *n*, otherwise. The smallest positive machine number, **ε**, is used as the starting value for computing the probabilities, which are rescaled by $(1 - p)^n\varepsilon$ if forward computation is performed and by $p^n\varepsilon$ if backward computation is done.

For the special case of *p* = 0, **BINPR** is set to 0 if *k* is greater than 0 and to 1 otherwise; and for the case *p* = 1, **BINPR** is set to 0 if *k* is less than *n* and to 1 otherwise.

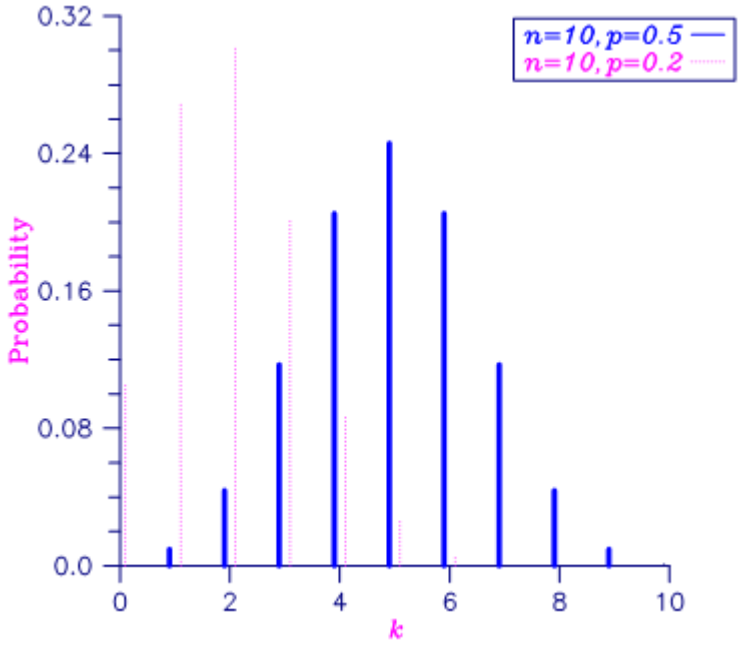


Figure 25, Binomial Probability Function

Comments

Informational Errors

Type	Code	Description
1	3	The input argument, K, is less than zero.
1	4	The input argument, K, is greater than the number of Bernoulli trials, N.

Example

Suppose *X* is a binomial random variable with *n* = 5 and *pin* = 0.95. In this example, we find the probability that *X* is equal to 3.

```
      USE UMACH_INT
      USE BINPR_INT
      IMPLICIT   NONE
      INTEGER    K, N, NOUT
      REAL       PIN, PR
!
      CALL UMACH (2, NOUT)
      K  = 3
      N  = 5
      PIN  = 0.95
      PR = BINPR(K,N,PIN)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 3 is ', F6.4)
      END
```

Output

```
 The probability that X is equal to 3 is 0.0214
```

GEODF

This function evaluates the discrete geometric cumulative probability distribution function.

Function Return Value

GEODF — Function value, the probability that a geometric random variable takes a value less than or equal to `IX`. (Output)

Required Arguments

IX — Argument for which the geometric cumulative distribution function is to be evaluated. (Input)

PIN — Probability parameter for each independent trial (the probability of success for each independent trial). `PIN` must be in the open interval (0, 1). (Input)

FORTRAN 90 Interface

Generic: `GEODF` (`IX`, `PIN`)

Specific: The specific interface names are `S_GEODF` and `D_GEODF`.

FORTRAN 77 Interface

Single: `GEODF` (`IX`, `PIN`)

Double: The double precision name is `DGEODF`.

Description

The function `GEODF` evaluates the discrete geometric cumulative probability distribution function with parameter p = `PIN`, defined

$$F(x \mid p) = \sum_{i=0}^{\lfloor x \rfloor} pq^i, \quad q = 1 - p, \quad 0 < p < 1$$

The return value is the probability that up to x trials would be observed before observing a success.

Example

In this example, we evaluate the probability function at **IX** = 3, **PIN** = 0.25.

```
          USE UMACH_INT
          USE GEODF_INT
          IMPLICIT NONE
          INTEGER NOUT, IX
          REAL PIN, PR
          CALL UMACH(2, NOUT)

          IX = 3
          PIN = 0.25e0
          PR = GEODF(IX, PIN)
          WRITE (NOUT, 99999) IX, PIN, PR
    99999 FORMAT (' GEODF(', I2, ', ', F4.2, ') = ', F10.6)
          END
```

Output

```
    GEODF( 3, 0.25) =   0.683594
```

GEOIN

This function evaluates the inverse of the geometric cumulative probability distribution function.

Function Return Value

GEOIN — Integer function value. The probability that a geometric random variable takes a value less than or equal to the returned value is the input probability, **P**. (Output)

Required Arguments

P — Probability for which the inverse of the discrete geometric cumulative distibution function is to be evaluated. **P** must be in the open interval (0, 1). (Input)

PIN — Probability parameter for each independent trial (the probability of success for each independent trial). **PIN** must be in the open interval (0, 1). (Input)

FORTRAN 90 Interface

Generic: **GEOIN** (**P**, **PIN**)
Specific: The specific interface names are **S_GEOIN** and **D_GEOIN**.

FORTRAN 77 Interface

Single: **GEOIN** (**P**, **PIN**)
Double: The double precision name is **DGEOIN**.

Description

The function **GEOIN** evaluates the inverse distribution function of a geometric random variable with parameter **PIN**. The inverse of the CDF is defined as the smallest integer *x* such that the geometric CDF is not less than a given value **P**, $0 < P < 1$.

Example

In this example, we evaluate the inverse probability function at **PIN** = 0.25, **P** = 0.6835.

```
      USE UMACH_INT
      USE GEOIN_INT
      IMPLICIT NONE
      INTEGER NOUT, IX
      REAL P, PIN
      CALL UMACH(2, NOUT)
      PIN = 0.25
      P =  0.6835
      IX = GEOIN(P, PIN)
      WRITE (NOUT, 99999) P, PIN, IX
99999 FORMAT (' GEOIN(', F4.2, ', ', F6.4 ') = ', I2)
      END
```

Output

```
 GEOIN(0.6835, 0.25) =  3
```

GEOPR

This function evaluates the discrete geometric probability density function.

Function Return Value

GEOPR — Function value, the probability that a random variable from a geometric distribution having parameter `PIN` will be equal to `IX`. (Output)

Required Arguments

IX — Argument for which the discrete geometric probability density function is to be evaluated. `IX` must be greater than or equal to 0. (Input)

PIN — Probability parameter of the geometric probability function (the probability of success for each independent trial). `PIN` must be in the open interval (0, 1). (Input)

FORTRAN 90 Interface

Generic: `GEOPR` (`IX`, `PIN`)
Specific: The specific interface names are `S_GEOPR` and `D_GEOPR`.

FORTRAN 77 Interface

Single: `GEOPR` (`IX`, `PIN`)
Double: The double precision name is `DGEOPR`.

Description

The function `GEOPR` evaluates the discrete geometric probability density function, defined

$$f\left(x \mid p\right) = pq^{x}, \quad q = 1 - p, \quad 0 < p < 1, \quad x = 0,\ 1,\ ...,\ HUGE(1)$$

where p = PIN.

Example

In this example, we evaluate the probability density function at **IX** = 3, **PIN** = 0.25.

```
          USE UMACH_INT
          USE GEOPR_INT
          IMPLICIT NONE
          INTEGER NOUT, IX
          REAL PIN, PR
          CALL UMACH(2, NOUT)
          IX = 3
          PIN = 0.25e0
          PR = GEOPR(IX, PIN)
          WRITE (NOUT, 99999) IX, PIN, PR
    99999 FORMAT (' GEOPR(', I2, ', ', F4.2, ') = ', F6.4)
          END
```

Output

```
     GEOPR( 3, 0.25) = 0.1055
```

HYPDF

This function evaluates the hypergeometric cumulative distribution function.

Function Return Value

HYPDF — Function value, the probability that a hypergeometric random variable takes a value less than or equal to **K**. (Output)
HYPDF is the probability that **K** or fewer defectives occur in a sample of size **N** drawn from a lot of size **L** that contains **M** defectives.
See Comment 1.

Required Arguments

K — Argument for which the hypergeometric cumulative distribution function is to be evaluated. (Input)

N — Sample size. (Input)
N must be greater than zero and greater than or equal to **K**.

M — Number of defectives in the lot. (Input)

L — Lot size. (Input)
L must be greater than or equal to **N** and **M**.

FORTRAN 90 Interface

Generic: **HYPDF** (**K**, **N**, **M**, **L**)
Specific: The specific interface names are **S_HYPDF** and **D_HYPDF**.

FORTRAN 77 Interface

Single: **HYPDF** (**K**, **N**, **M**, **L**)
Double: The double precision name is **DHYPDF**.

Description

The function **HYPDF** evaluates the cumulative distribution function of a hypergeometric random variable with parameters *n*, *l*, and *m*. The hypergeometric random variable *X* can be thought of as the number of items of a given type in a random sample of size *n* that is drawn without replacement from a population of size *l* containing *m* items of this type. The probability function is

$$\Pr(X = j) = \frac{\binom{m}{j}\binom{l-m}{n-j}}{\binom{l}{n}} \quad \text{for } j = i,\ i+1,\ i+2,\ \ldots\ \min(n,m)$$

where *i* = max(0, *n* - *l* + *m*).

If *k* is greater than or equal to *i* and less than or equal to min(*n*, *m*), **HYPDF** sums the terms in this expression for *j* going from *i* up to *k*. Otherwise, **HYPDF** returns 0 or 1, as appropriate. So, as to avoid rounding in the accumulation, **HYPDF** performs the summation differently depending on whether or not *k* is greater than the mode of the distribution, which is the greatest integer less than or equal to (*m* + 1)(*n* + 1)/(*l* + 2).

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = HYPDF(K, N, M, L)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(HYPDF(K, N, M, L))
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Informational errors

Type	Code	Description
1	5	The input argument, K, is less than zero.
1	6	The input argument, K, is greater than the sample size.

Example

Suppose *X* is a hypergeometric random variable with *n* = 100, *l* = 1000, and *m* = 70. In this example, we evaluate the distribution function at 7.

```
      USE UMACH_INT
      USE HYPDF_INT
      IMPLICIT   NONE
      INTEGER    K, L, M, N, NOUT
      REAL       DF
!
      CALL UMACH (2, NOUT)
      K  = 7
      N  = 100
      L  = 1000
      M  = 70
      DF = HYPDF(K,N,M,L)
      WRITE (NOUT,99999) DF
99999 FORMAT (' The probability that X is less than or equal to 7 is ' &
             , F6.4)
      END
```

Output

```
 The probability that X is less than or equal to 7 is 0.5995
```

HYPPR

This function evaluates the hypergeometric probability density function.

Function Return Value

HYPPR — Function value, the probability that a hypergeometric random variable takes a value equal to **K**. (Output)
HYPPR is the probability that exactly **K** defectives occur in a sample of size **N** drawn from a lot of size **L** that contains **M** defectives.
See Comment 1.

Required Arguments

K — Argument for which the hypergeometric probability function is to be evaluated. (Input)

N — Sample size. (Input)
N must be greater than zero and greater than or equal to **K**.

M — Number of defectives in the lot. (Input)

L — Lot size. (Input)
L must be greater than or equal to **N** and **M**.

FORTRAN 90 Interface

Generic: **HYPPR** (**K**, **N**, **M**, **L**)
Specific: The specific interface names are **S_HYPPR** and **D_HYPPR**.

FORTRAN 77 Interface

Single: **HYPPR** (**K**, **N**, **M**, **L**)
Double: The double precision name is **DHYPPR**.

Description

The function **HYPPR** evaluates the probability density function of a hypergeometric random variable with parameters *n*, *l*, and *m*. The hypergeometric random variable *X* can be thought of as the number of items of a given type in a random sample of size *n* that is drawn without replacement from a population of size *l* containing *m* items of this type. The probability density function is

$$\Pr(X = k) = \frac{\binom{m}{k}\binom{l-m}{n-k}}{\binom{l}{n}} \quad \text{for } k = i,\ i+1,\ i+2,\ \ldots\ \min(n,m)$$

where *i* = max(0, *n* - *l* + *m*). **HYPPR** evaluates the expression using log gamma functions.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = HYPPR(K, N, M, L)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(HYPPR(K, N, M, L))
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Informational errors

Type	Code	Description
1	5	The input argument, K, is less than zero.
1	6	The input argument, K, is greater than the sample size.

Example

Suppose *X* is a hypergeometric random variable with *n* = 100, *l* = 1000, and *m* = 70. In this example, we evaluate the probability function at 7.

```
      USE UMACH_INT
      USE HYPPR_INT

      IMPLICIT   NONE
      INTEGER    K, L, M, N, NOUT
      REAL       PR
```

```
!
      CALL UMACH (2, NOUT)
      K  = 7
      N  = 100
      L  = 1000
      M  = 70
      PR = HYPPR(K,N,M,L)
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 7 is ', F6.4)
      END
```

Output

```
 The probability that X is equal to 7 is 0.1628
```

POIDF

This function evaluates the Poisson cumulative distribution function.

Function Return Value

POIDF — Function value, the probability that a Poisson random variable takes a value less than or equal to K. (Output)

Required Arguments

K — Argument for which the Poisson cumulative distribution function is to be evaluated. (Input)

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

FORTRAN 90 Interface

Generic: POIDF (K, THETA)
Specific: The specific interface names are S_POIDF and D_POIDF.

FORTRAN 77 Interface

Single: POIDF (K, THETA)
Double: The double precision name is DPOIDF.

Description

The function POIDF evaluates the cumulative distribution function of a Poisson random variable with parameter THETA. THETA, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = THETA) is

$$f(x) = e^{-\theta}\theta^x/x!, \quad \text{for } x = 0, 1, 2, \ldots$$

The individual terms are calculated from the tails of the distribution to the mode of the distribution and summed. POIDF uses the recursive relationship

$$f(x+1) = f(x)\theta/(x+1), \quad \text{for } x = 0, 1, 2, \ldots k-1,$$

with $f(0) = e^{-\theta}$

Comments

Informational Error

Type	Code	Description
1	1	The input argument, K, is less than zero.

Example

Suppose X is a Poisson random variable with $\theta = 10$. In this example, we evaluate the distribution function at 7.

```
      USE UMACH_INT
      USE POIDF_INT
      IMPLICIT   NONE
      INTEGER    K, NOUT
      REAL       DF, THETA
!
      CALL UMACH (2, NOUT)
      K     = 7
      THETA = 10.0
      DF    = POIDF(K,THETA)
      WRITE (NOUT,99999) DF
99999 FORMAT (' The probability that X is less than or equal to ', &
             '7 is ', F6.4)
      END
```

Output

```
 The probability that X is less than or equal to 7 is 0.2202
```

POIPR

This function evaluates the Poisson probability density function.

Function Return Value

POIPR — Function value, the probability that a Poisson random variable takes a value equal to **K**. (Output)

Required Arguments

K — Argument for which the Poisson probability density function is to be evaluated. (Input)

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

FORTRAN 90 Interface

Generic: **POIPR** (**K**, **THETA**)
Specific: The specific interface names are **S_POIPR** and **D_POIPR**.

FORTRAN 77 Interface

Single: **POIPR** (**K**, **THETA**)
Double: The double precision name is **DPOIPR**.

Description

The function **POIPR** evaluates the probability density function of a Poisson random variable with parameter **THETA**. **THETA**, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = **THETA**) is

$$f(x) = e^{-\theta}\theta^{k}/k!, \qquad \text{for } k = 0, 1, 2, \ldots$$

POIPR evaluates this function directly, taking logarithms and using the log gamma function.

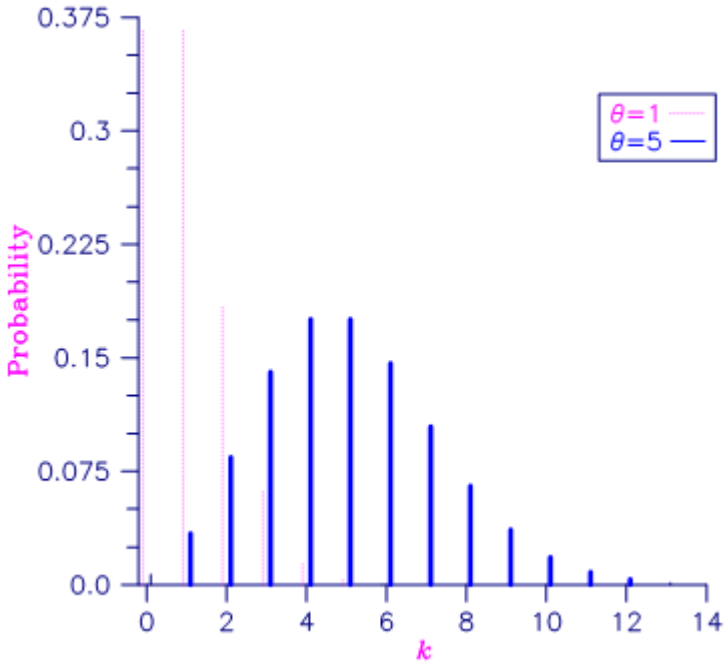


Figure 26, Poisson Probability Density Function

Comments

Informational Error

Type	Code	Description
1	1	The input argument, K, is less than zero.

Example

Suppose *X* is a Poisson random variable with θ = 10. In this example, we evaluate the probability function at 7.

```
      USE UMACH_INT
      USE POIPR_INT
      IMPLICIT   NONE

      INTEGER    K, NOUT
      REAL       PR, THETA
!
      CALL UMACH (2, NOUT)
      K     = 7
      THETA = 10.0
      PR    = POIPR(K,THETA)
```

```
      WRITE (NOUT,99999) PR
99999 FORMAT (' The probability that X is equal to 7 is ', F6.4)
      END
```

Output

```

 The probability that X is equal to 7 is 0.0901
```

UNDDF

This function evaluates the discrete uniform cumulative distribution function.

Function Return Value

UNDDF — Function value, the probability that a uniform random variable takes a value less than or equal to `IX`. (Output)

Required Arguments

IX — Argument for which the discrete uniform cumulative distribution function is to be evaluated. (Input)

N — Scale parameter. `N` must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: `UNDDF` (`IX`, `N`)
Specific: The specific interface names are `S_UNDDF` and `D_UNDDF`.

FORTRAN 77 Interface

Single: `UNDDF` (`IX`, `N`)
Double: The double precision name is `DUNDDF`.

Description

The notation below uses the floor and ceiling function notation, $\lfloor . \rfloor$ and $\lceil . \rceil$.

The function `UNDDF` evaluates the discrete uniform cumulative probability distribution function with scale parameter `N`, defined

$$F\left(x|N\right) = \frac{\lfloor x \rfloor}{N}, \quad 1 \le x \le N$$

Example

In this example, we evaluate the probability function at `IX` = 3, `N` = 5.

```
      USE UMACH_INT
      USE UNDDF_INT
      IMPLICIT NONE
      INTEGER NOUT, IX, N
      REAL PR
      CALL UMACH(2, NOUT)
      IX = 3
      N = 5
      PR = UNDDF(IX, N)
      WRITE (NOUT, 99999) IX, N, PR
99999 FORMAT (' UNDDF(', I2, ', ', I2, ') = ', F6.4)
      END
```

Output

```
 UNDDF( 3,  5) = 0.6000
```

UNDIN

This function evaluates the inverse of the discrete uniform cumulative distribution function.

Function Return Value

UNDIN — Integer function value. The probability that a uniform random variable takes a value less than or equal to the returned value is the input probability, **P**. (Output)

Required Arguments

P — Probability for which the inverse of the discrete uniform cumulative distribution function is to be evaluated. **P** must be nonnegative and less than or equal to 1.0. (Input)

N — Scale parameter. **N** must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: **UNDIN** (**P**, **N**)
Specific: The specific interface names are **S_UNDIN** and **D_UNDIN**.

FORTRAN 77 Interface

Single: **UNDIN** (**P**, **N**)
Double: The double precision name is **DUNDIN**.

Description

The notation below uses the floor and ceiling function notation, $\lfloor . \rfloor$ and $\lceil . \rceil$.

The function **UNDIN** evaluates the inverse distribution function of a discrete uniform random variable with scale parameter **N**, defined

$$x = \lceil pN \rceil, \quad 0 \le p \le 1$$

Example

In this example, we evaluate the inverse probability function at **P** = 0.6, **N** = 5.

```
      USE UMACH_INT
      USE UNDIN_INT
      IMPLICIT NONE
      INTEGER NOUT, N, IX
      REAL P
      CALL UMACH(2, NOUT)
      P = 0.60
      N = 5
      IX = UNDIN(P, N)
      WRITE (NOUT, 99999) P, N, IX
99999 FORMAT (' UNDIN(', F4.2, ', ', I2 ') = ', I2)
      END
```

Output

```
 UNDIN(0.60,  5) =  3
```

UNDPR

This function evaluates the discrete uniform probability density function.

Function Return Value

UNDPR — Function value, the probability that a random variable from a uniform distribution having scale parameter **N** will be equal to **IX**. (Output)

Required Arguments

IX — Argument for which the discrete uniform probability density function is to be evaluated. (Input)

N — Scale parameter. **N** must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: **UNDPR** (**IX**, **N**)
Specific: The specific interface names are **S_UNDPR** and **D_UNDPR**.

FORTRAN 77 Interface

Single: **UNDPR** (**IX**, **N**)
Double: The double precision name is **DUNDPR**.

Description

The discrete uniform PDF is defined for positive integers x in the range 1, ..., N, $N > 0$. It has the value $y = f\left(x|N\right) = \frac{1}{N}$, for $1 \le x \le N$, and $y = 0$, for $x > N$. Allowing the function to accept values of x resulting in $y = 0$, for $x > N$ is provided as a convenience to the user. Values of $x \le 0$ are errors.

Example

In this example, we evaluate the discrete uniform probability density function at **IX** = 3, **N** = 5.

```
      USE UMACH_INT
      USE UNDPR_INT
```

```
      IMPLICIT NONE
      INTEGER NOUT, IX, N
      REAL PR
      CALL UMACH(2, NOUT)
      IX = 3
      N = 5
      PR = UNDPR(IX, N)
      WRITE (NOUT, 99999) IX, N, PR
99999 FORMAT (' UNDPR(', I2, ', ', I2, ') = ', F6.4)
      END
```

Output

```
 UNDPR( 3,  5) = 0.2000
```

AKS1DF

This function evaluates the cumulative distribution function of the one-sided Kolmogorov-Smirnov goodness of fit D^+ or D^- test statistic based on continuous data for one sample.

Function Return Value

AKS1DF — The probability of a smaller **D**. (Output)

Required Arguments

NOBS — The total number of observations in the sample. (Input)

D — The D^+ or D^- test statistic. (Input)
D is the maximum positive difference of the empirical cumulative distribution function (CDF) minus the hypothetical CDF or the maximum positive difference of the hypothetical CDF minus the empirical CDF.

FORTRAN 90 Interface

Generic: **AKS1DF** (**NOBS**, **D**)
Specific: The specific interface names are **S_AKS1DF** and **D_AKS1DF**.

FORTRAN 77 Interface

Single: **AKS1DF** (**NOBS**, **D**)
Double: The double precision name is **DKS1DF**.

Description

Routine **AKS1DF** computes the cumulative distribution function (CDF) for the one-sided Kolmogorov-Smirnov one-sample

$$D^+ \text{ or } D^-$$

statistic when the theoretical CDF is strictly continuous. Let

$$F(x)$$

denote the theoretical distribution function, and let

$$S_n(x)$$

denote the empirical distribution function obtained from a sample of size **NOBS**. Then, the

$$D^+$$

statistic is computed as

$$D^+ = \sup_x \left[F(x) - S_n(x) \right]$$

while the one-sided

$$D^-$$

statistic is computed as

$$D^- = \sup_x \left[S_n(x) - F(x) \right]$$

Exact probabilities are computed according to a method given by Conover (1980, page 350) for sample sizes of 80 or less. For sample sizes greater than 80, Smirnov's asymptotic result is used, that is, the value of the CDF is taken as $1 - e^{-2nd^2}$, where *d* is D^+ or D^- (Kendall and Stuart, 1979, page 482). This asymptotic expression is conservative (the value returned by **AKS1DF** is smaller than the exact value, when the sample size exceeds 80).

Comments

1. Workspace may be explicitly provided, if desired, by use of **AK21DF/DK21DF**. The reference is:

   AK2DF (**NOBS**, **D**, **WK**)

   The additional argument is:

   WK — Work vector of length 3 * **NOBS** + 3 if **NOBS** ≤ 80. **WK** is not used if **NOBS** is greater than 80.

2. Informational errors

Type	Code	Description
1	2	Since the **D** test statistic is less than zero, the distribution function is zero at **D**.
1	3	Since the **D** test statistic is greater than one, the distribution function is one at **D**.

3. If **NOBS** ≤ 80, then exact one-sided probabilities are computed. In this case, on the order of **NOBS**^{2} operations are required. For **NOBS** > 80, approximate one-sided probabilities are computed. These approximate probabilities require very few computations.

4. An approximate two-sided probability for the **D** = max (D^+, D^-) statistic can be computed as twice the **AKS1DF** probability for **D** (minus one, if the probability from **AKS1DF** is greater than 0.5).

Programming Notes

Routine **AKS1DF** requires on the order of **NOBS**^{2} operations to compute the exact probabilities, where an operation consists of taking ten or so logarithms. Because so much computation is occurring within each "operation," **AKS1DF** is much slower than its two-sample counterpart, function AKS2DF.

Example

In this example, the exact one-sided probabilities for the tabled values of D^+ or D^-, given, for example, in Conover (1980, page 462), are computed. Tabled values at the 10% level of significance are used as input to **AKS1DF** for sample sizes of 5 to 50 in increments of 5 (the last two tabled values are obtained using the asymptotic critical values of

$$1.07 / \sqrt{NOBS}$$

The resulting probabilities should all be close to 0.90.

```
      USE UMACH_INT
      USE AKS1DF_INT
      IMPLICIT   NONE
      INTEGER    I, NOBS, NOUT
      REAL       D(10)
!
      DATA D/0.447, 0.323, 0.266, 0.232, 0.208, 0.190, 0.177, 0.165, &
          0.160, 0.151/
!
      CALL UMACH (2, NOUT)
!
      DO 10  I=1, 10
         NOBS = 5*I
!
         WRITE (NOUT,99999) D(I), NOBS, AKS1DF(NOBS,D(I))
!
99999    FORMAT (' One-sided Probability for D = ', F8.3, ' with NOBS ' &
               , '= ', I2, ' is ', F8.4)
   10 CONTINUE
      END
```

Output

```
 One-sided Probability for D =    0.447 with NOBS =  5 is   0.9000
 One-sided Probability for D =    0.323 with NOBS = 10 is   0.9006
 One-sided Probability for D =    0.266 with NOBS = 15 is   0.9002
 One-sided Probability for D =    0.232 with NOBS = 20 is   0.9009
 One-sided Probability for D =    0.208 with NOBS = 25 is   0.9002
```

```
    One-sided Probability for D =    0.190 with NOBS = 30 is   0.8992
    One-sided Probability for D =    0.177 with NOBS = 35 is   0.9011
    One-sided Probability for D =    0.165 with NOBS = 40 is   0.8987
    One-sided Probability for D =    0.160 with NOBS = 45 is   0.9105
    One-sided Probability for D =    0.151 with NOBS = 50 is   0.9077
```

AKS2DF

This function evaluates the cumulative distribution function of the Kolmogorov-Smirnov goodness of fit *D* test statistic based on continuous data for two samples.

Function Return Value

AKS2DF — The probability of a smaller **D**. (Output)

Required Arguments

NOBSX — The total number of observations in the first sample. (Input)

NOBSY — The total number of observations in the second sample. (Input)

D — The **D** test statistic. (Input)
D is the maximum absolute difference between empirical cumulative distribution functions (CDFs) of the two samples.

FORTRAN 90 Interface

Generic: AKS2DF (NOBSX, NOBSY, D)
Specific: The specific interface names are S_AKS2DF and D_AKS2DF.

FORTRAN 77 Interface

Single: AKS2DF (NOBSX, NOBSY, D)
Double: The double precision name is DKS2DF.

Description

Function **AKS2DF** computes the cumulative distribution function (CDF) for the two-sided Kolmogorov-Smirnov two-sample *D* statistic when the theoretical CDF is strictly continuous. Exact probabilities are computed according to a method given by Kim and Jennrich (1973). Approximate asymptotic probabilities are computed according to methods also given in this reference.

Let $F_n(x)$ and $G_m(x)$ denote the empirical distribution functions for the two samples, based on n = **NOBSX** and m = **NOBSY** observations. Then, the *D* statistic is computed as

$$D = \sup_x |F_n(x) - G_m(x)|$$

Comments

1. Workspace may be explicitly provided, if desired, by use of AK22DF/DK22DF. The reference is:

   AK22DF (NOBSX, NOBSY, D, WK)

   The additional argument is:

   WK — Work vector of length max(NOBSX, NOBSY) + 1.

2. Informational errors

Type	Code	Description
1	2	Since the D test statistic is less than zero, then the distribution function is zero at D.
1	3	Since the D test statistic is greater than one, then the distribution function is one at D.

Programming Notes

Function AKS2DF requires on the order of NOBSX * NOBSY operations to compute the exact probabilities, where an operation consists of an addition and a multiplication. For NOBSX * NOBSY less than 10000, the exact probability is computed. If this is not the case, then the Smirnov approximation discussed by Kim and Jennrich (1973) is used if the minimum of NOBSX and NOBSY is greater than ten percent of the maximum of NOBSX and NOBSY, or if the minimum is greater than 80. Otherwise, the Kolmogorov approximation discussed by Kim and Jennrich (1973) is used.

Example

Function AKS2DF is used to compute the probability of a smaller *D* statistic for a variety of sample sizes using values close to the 0.95 probability value.

```
      USE UMACH_INT
      USE AKS2DF_INT

      IMPLICIT   NONE
      INTEGER    I, NOBSX(10), NOBSY(10), NOUT
      REAL       D(10)
!
      DATA NOBSX/5, 20, 40, 70, 110, 200, 200, 200, 100, 100/
      DATA NOBSY/10, 10, 10, 10, 10, 20, 40, 60, 80, 100/
      DATA D/0.7, 0.55, 0.475, 0.4429, 0.4029, 0.2861, 0.2113, 0.1796, &
          0.18, 0.18/
!
```

```
      CALL UMACH (2, NOUT)
!
      DO 10  I=1, 10
!
         WRITE (NOUT,99999) D(I), NOBSX(I), NOBSY(I), &
                            AKS2DF(NOBSX(I),NOBSY(I),D(I))
!
99999    FORMAT (' Probability for D = ', F5.3, ' with NOBSX = ', I3, &
               ' and NOBSY = ', I3, ' is ', F9.6, '.')
   10 CONTINUE
      END
```

Output

```
 Probability for D = 0.700 with NOBSX =   5 and NOBSY =  10 is  0.980686.
 Probability for D = 0.550 with NOBSX =  20 and NOBSY =  10 is  0.987553.
 Probability for D = 0.475 with NOBSX =  40 and NOBSY =  10 is  0.972423.
 Probability for D = 0.443 with NOBSX =  70 and NOBSY =  10 is  0.961646.
 Probability for D = 0.403 with NOBSX = 110 and NOBSY =  10 is  0.928667.
 Probability for D = 0.286 with NOBSX = 200 and NOBSY =  20 is  0.921126.
 Probability for D = 0.211 with NOBSX = 200 and NOBSY =  40 is  0.917110.
 Probability for D = 0.180 with NOBSX = 200 and NOBSY =  60 is  0.914520.
 Probability for D = 0.180 with NOBSX = 100 and NOBSY =  80 is  0.908185.
 Probability for D = 0.180 with NOBSX = 100 and NOBSY = 100 is  0.946098.
```

ALNDF

This function evaluates the lognormal cumulative probability distribution function.

Function Return Value

ALNDF — Function value, the probability that a standard lognormal random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the lognormal cumulative distribution function is to be evaluated. (Input)

AMU — Location parameter of the lognormal cumulative distribution function. (Input)

SIGMA — Shape parameter of the lognormal cumulative distribution function. SIGMA must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: ALNDF (X, AMU, SIGMA)

Specific: The specific interface names are S_ALNDF and D_ALNDF.

FORTRAN 77 Interface

Single: ALNDF (X, AMU, SIGMA)

Double: The double precision name is DLNDF.

Description

The function ALNDF evaluates the lognormal cumulative probability distribution function, defined as

$$F(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\int_0^x \frac{1}{t} e^{-\frac{1}{2}\left(\frac{\log(t)-\mu}{\sigma}\right)^2} dt = \Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$$

where

$$\Phi(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-\frac{1}{2}u^2} du$$

is the standard normal CDF.

Example

In this example, we evaluate the probability distribution function at **X** = 0.7137, **AMU** = 0.0, **SIGMA** = 0.5.

```
      USE UMACH_INT
      USE ALNDF_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, SIGMA, PR
      CALL UMACH(2, NOUT)
      X = .7137
      AMU = 0.0
      SIGMA = 0.5
      PR = ALNDF(X, AMU, SIGMA)
      WRITE (NOUT, 99999) X, AMU, SIGMA, PR
99999 FORMAT (' ALNDF(', F6.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 ALNDF(  0.71, 0.00, 0.50) = 0.2500
```

ALNIN

This function evaluates the inverse of the lognormal cumulative probability distribution function.

Function Return Value

ALNIN — Function value, the probability that a lognormal random variable takes a value less than or equal to the returned value is the input probability **P**. (Output)

Required Arguments

P — Probability for which the inverse of the lognormal distribution function is to be evaluated. (Input)

AMU — Location parameter of the lognormal cumulative distribution function. (Input)

SIGMA — Shape parameter of the lognormal cumulative distribution function. **SIGMA** must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: **ALNIN** (**P**, **AMU**, **SIGMA**)

Specific: The specific interface names are **S_ALNIN** and **D_ALNIN**.

FORTRAN 77 Interface

Single: **ALNIN** (**P**, **AMU**, **SIGMA**)

Double: The double precision name is **DLNIN**.

Description

The function **ALNIN** evaluates the inverse distribution function of a lognormal random variable with location parameter **AMU** and scale parameter **SIGMA**. The probability that a standard lognormal random variable takes a value less than or equal to the returned value is **P**.

Example

In this example, we evaluate the inverse probability function at **P** = 0.25, **AMU** = 0.0, **SIGMA** = 0.5.

```
      USE UMACH_INT
      USE ALNIN_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, SIGMA, P
      CALL UMACH(2, NOUT)
      P = .25
      AMU = 0.0
      SIGMA = 0.5
      X = ALNIN(P, AMU, SIGMA)
      WRITE (NOUT, 99999) P, AMU, SIGMA, X
99999 FORMAT (' ALNIN(', F6.3, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 ALNIN( 0.250, 0.00, 0.50) = 0.7137
```

ALNPR

This function evaluates the lognormal probability density function.

Function Return Value

ALNPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the lognormal probability density function is to be evaluated. (Input)

AMU — Location parameter of the lognormal probability function. (Input)

SIGMA — Shape parameter of the lognormal probability function. SIGMA must be greater than 0. (Input)

FORTRAN 90 Interface

Generic: ALNPR (X, AMU, SIGMA)

Specific: The specific interface names are S_ALNPR and D_ALNPR.

FORTRAN 77 Interface

Single: ALNPR (X, AMU, SIGMA)

Double: The double precision name is DLNPR.

Description

The function ALNPR evaluates the lognormal probability density function, defined as

$$f\left(x \mid \mu,\sigma\right) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\left(\frac{\left(\log\left(x\right)-\mu\right)^2}{2\sigma^2}\right)}$$

Example

In this example, we evaluate the probability function at X = 1.0, AMU = 0.0, SIGMA = 0.5.

```
      USE UMACH_INT
      USE ALNPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, SIGMA, PR
      CALL UMACH(2, NOUT)
      X = 1.0
      AMU = 0.0
      SIGMA = 0.5
      PR = ALNPR(X, AMU, SIGMA)
      WRITE (NOUT, 99999) X, AMU, SIGMA, PR
99999 FORMAT (' ALNPR(', F6.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 ALNPR(  1.00, 0.00, 0.50) = 0.7979
```

ANORDF

This function evaluates the standard normal (Gaussian) cumulative distribution function.

Function Return Value

ANORDF — Function value, the probability that a normal random variable takes a value less than or equal to **X**. (Output)

Required Arguments

X — Argument for which the normal cumulative distribution function is to be evaluated. (Input)

FORTRAN 90 Interface

Generic: **ANORDF** (**X**)

Specific: The specific interface names are **S_ANORDF** and **D_ANORDF**.

FORTRAN 77 Interface

Single: **ANORDF** (**X**)

Double: The double precision name is **DNORDF**.

Description

Function **ANORDF** evaluates the cumulative distribution function, **Φ**, of a standard normal (Gaussian) random variable, that is,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt$$

The value of the distribution function at the point *x* is the probability that the random variable takes a value less than or equal to *x*.

The standard normal distribution (for which **ANORDF** is the distribution function) has mean of 0 and variance of 1. The probability that a normal random variable with mean **μ** and variance σ^2 is less than *y* is given by **ANORDF** evaluated at (*y* **-** **μ**)/**σ**.

Φ(*x*) is evaluated by use of the complementary error function, erfc. (See **ERFC**, IMSL MATH/LIBRARY Special Functions). The relationship is:

$$\Phi(x) = \text{erfc}(-x/\sqrt{2.0})/2$$

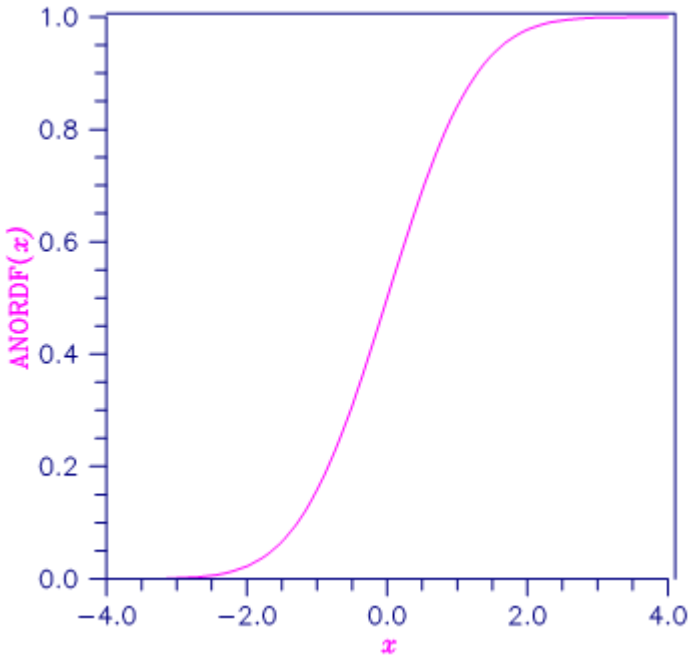


Figure 27, Standard Normal Distribution Function

Example

Suppose *X* is a normal random variable with mean 100 and variance 225. In this example, we find the probability that *X* is less than 90, and the probability that *X* is between 105 and 110.

```
      USE UMACH_INT
      USE ANORDF_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       P, X1, X2
!
      CALL UMACH (2, NOUT)
      X1 = (90.0-100.0)/15.0
      P  = ANORDF(X1)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 90 is ', F6.4)
      X1 = (105.0-100.0)/15.0
      X2 = (110.0-100.0)/15.0
      P  = ANORDF(X2) - ANORDF(X1)
```

```
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 105 and 110 is ', &
             F6.4)
      END
```

Output

```

 The probability that X is less than 90 is 0.2525
 The probability that X is between 105 and 110 is 0.1169
```

ANORIN

This function evaluates the inverse of the standard normal (Gaussian) cumulative distribution function.

Function Return Value

ANORIN — Function value. (Output)
The probability that a standard normal random variable takes a value less than or equal to **ANORIN** is **P**.

Required Arguments

P — Probability for which the inverse of the normal cumulative distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

FORTRAN 90 Interface

Generic: **ANORIN** (**P**)
Specific: The specific interface names are **S_ANORIN** and **D_ANORIN**.

FORTRAN 77 Interface

Single: **ANORIN** (**P**)
Double: The double precision name is **DNORIN**.

Description

Function **ANORIN** evaluates the inverse of the cumulative distribution function, **Φ**, of a standard normal (Gaussian) random variable, that is, x = **ANORIN**(**P**) = $\Phi^{-1}(p)$, where

$$p = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x. The standard normal distribution has a mean of 0 and a variance of 1.

Example

In this example, we compute the point such that the probability is 0.9 that a standard normal random variable is less than or equal to this point.

```
      USE UMACH_INT
      USE ANORIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       P, X
!
      CALL UMACH (2, NOUT)
      P = 0.9
      X = ANORIN(P)
      WRITE (NOUT,99999) X
99999 FORMAT (' The 90th percentile of a standard normal is ', F6.4)
      END
```

Output

```
 The 90th percentile of a standard normal is 1.2816
```

ANORPR

This function evaluates the normal probability density function.

Function Return Value

ANORPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the normal probability density function is to be evaluated. (Input)

FORTRAN 90 Interface

Generic: **ANORPR** (**X**)

Specific: The specific interface names are **S_NORPR** and **D_NORPR**.

FORTRAN 77 Interface

Single: **ANORPR** (**X**)

Double: The double precision name is **DNORPR**.

Description

The function **ANORPR** evaluates the normal probability density function, defined as

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, \quad -\infty < x$$

Example

In this example, we evaluate the probability function at **X** = 0.5.

```
      USE UMACH_INT
      USE ANORPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, PR
```

```
      CALL UMACH(2, NOUT)
      X = 0.5
      PR = ANORPR(X)
      WRITE (NOUT, 99999) X, PR
99999 FORMAT (' ANORPR(', F4.2, ') = ', F6.4)
      END
```

Output

```
 ANORPR(0.50) = 0.3521
```

BETDF

This function evaluates the beta cumulative distribution function.

Function Return Value

BETDF — Probability that a random variable from a beta distribution having parameters **PIN** and **QIN** will be less than or equal to **X**. (Output)

Required Arguments

X — Argument for which the beta distribution function is to be evaluated. (Input)

PIN — First beta distribution parameter. (Input)
PIN must be positive.

QIN — Second beta distribution parameter. (Input)
QIN must be positive.

FORTRAN 90 Interface

Generic: **BETDF** (**X**, **PIN**, **QIN**)
Specific: The specific interface names are **S_BETDF** and **D_BETDF**.

FORTRAN 77 Interface

Single: **BETDF** (**X**, **PIN**, **QIN**)
Double: The double precision name is **DBETDF**.

Description

Function **BETDF** evaluates the cumulative distribution function of a beta random variable with parameters **PIN** and **QIN**. This function is sometimes called the *incomplete beta ratio* and, with p = **PIN** and q = **QIN**, is denoted by $I_x(p, q)$. It is given by

$$I_x(p,q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}\int_0^x t^{p-1}(1-t)^{q-1}dt$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function $I_x(p, q)$ is the probability that the random variable takes a value less than or equal to x.

The integral in the expression above is called the *incomplete beta function* and is denoted by $\beta_x(p, q)$. The constant in the expression is the reciprocal of the *beta function* (the incomplete function evaluated at one) and is denoted by $\beta(p, q)$.

Function **BETDF** uses the method of Bosten and Battiste (1974).

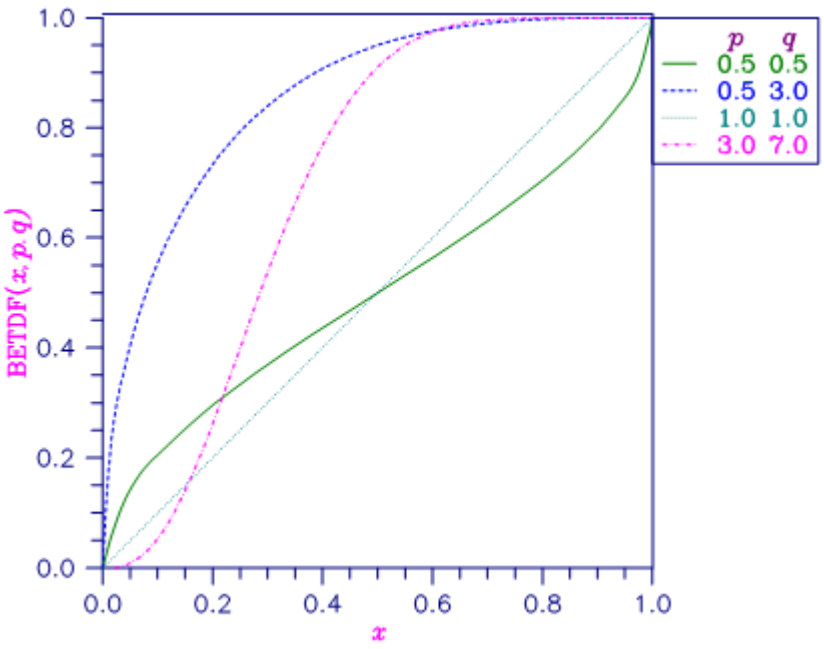


Figure 28, Beta Distribution Function

Comments

Informational Errors

Type	Code	Description
1	1	Since the input argument X is less than or equal to zero, the distribution function is equal to zero at X.
1	2	Since the input argument X is greater than or equal to one, the distribution function is equal to one at X.

Example

Suppose *X* is a beta random variable with parameters 12 and 12. (*X* has a symmetric distribution.) In this example, we find the probability that *X* is less than 0.6 and the probability that *X* is between 0.5 and 0.6. (Since *X* is a symmetric beta random variable, the probability that it is less than 0.5 is 0.5.)

```
      USE UMACH_INT
      USE BETDF_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       P, PIN, QIN, X
!
      CALL UMACH (2, NOUT)
      PIN = 12.0
      QIN = 12.0
      X   = 0.6
      P   = BETDF(X,PIN,QIN)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 0.6 is ', F6.4)
      X = 0.5
      P = P - BETDF(X,PIN,QIN)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 0.5 and 0.6 is ', &
            F6.4)
      END
```

Output

```
 The probability that X is less than 0.6 is 0.8364
 The probability that X is between 0.5 and 0.6 is 0.3364
```

BETIN

This function evaluates the inverse of the beta cumulative distribution function.

Function Return Value

BETIN — Function value. (Output)
The probability that a beta random variable takes a value less than or equal to **BETIN** is **P**.

Required Arguments

P — Probability for which the inverse of the beta distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

PIN — First beta distribution parameter. (Input)
PIN must be positive.

QIN — Second beta distribution parameter. (Input)
QIN must be positive.

FORTRAN 90 Interface

Generic: **BETIN** (**P**, **PIN**, **QIN**)
Specific: The specific interface names are **S_BETIN** and **D_BETIN**.

FORTRAN 77 Interface

Single: **BETIN** (**P**, **PIN**, **QIN**)
Double: The double precision name is **DBETIN**.

Description

The function **BETIN** evaluates the inverse distribution function of a beta random variable with parameters **PIN** and **QIN**, that is, with P = **P**, p = **PIN**, and q = **QIN**, it determines x (equal to **BETIN**(**P**, **PIN**, **QIN**)), such that

$$P = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}\int_0^x t^{p-1}(1-t)^{q-1}dt$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to x is P.

Comments

Informational Error

Type	Code	Description
3	1	The value for the inverse Beta distribution could not be found in 100 iterations. The best approximation is used.

Example

Suppose X is a beta random variable with parameters 12 and 12. (X has a symmetric distribution.) In this example, we find the value x_0 such that the probability that $X \leq x_0$ is 0.9.

```
      USE UMACH_INT
      USE BETIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       P, PIN, QIN, X
!
      CALL UMACH (2, NOUT)
      PIN = 12.0
      QIN = 12.0
      P   = 0.9
      X   = BETIN(P,PIN,QIN)
      WRITE (NOUT,99999) X
99999 FORMAT (' X is less than ', F6.4, ' with probability 0.9.')
      END
```

Output

```
 X is less than 0.6299 with probability 0.9.
```

BETPR

This function evaluates the beta probability density function.

Function Return Value

BETPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the beta probability density function is to be evaluated. (Input)

PIN — First beta distribution parameter. (Input)
`PIN` must be positive.

QIN — Second beta distribution parameter. (Input)
`QIN` must be positive.

FORTRAN 90 Interface

Generic: `BETPR` (`X`, `PIN`, `QIN`)
Specific: The specific interface names are `S_BETPR` and `D_BETPR`.

FORTRAN 77 Interface

Single: `BETPR` (`X`, `PIN`, `QIN`)
Double: The double precision name is `DBETPR`.

Description

The function `BETPR` evaluates the beta probability density function with parameters `PIN` and `QIN`. Using x = `X`, a = `PIN` and b = `QIN`, the beta distribution is defined as

$$f(x \mid a, b) = \frac{1}{B(a, b)}(1-x)^{b-1}x^{a-1}, \quad a, b > 0, \quad 0 \le x \le 1$$

where beta function $B(a, b)$ is computed using IMSL function `BETA` (see the Special Functions book, *Chapter 4, Gamma and Related Functions*).

Example

In this example, we evaluate the probability function at **X** = 0.75, **PIN** = 2.0, **QIN** = 0.5.

```
      USE UMACH_INT
      USE BETPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, PIN, QIN, PR
      CALL UMACH(2, NOUT)
      X = .75
      PIN = 2.0
      QIN = 0.5
      PR = BETPR(X, PIN, QIN)
      WRITE (NOUT, 99999) X, PIN, QIN, PR
99999 FORMAT (' BETPR(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 BETPR(0.75, 2.00, 0.50) = 1.1250
```

BETNDF

This function evaluates the noncentral beta cumulative distribution function (CDF).

Function Return Value

BETNDF — Probability that a random variable from a beta distribution having shape parameters SHAPE1 and SHAPE2 and noncentrality parameter LAMBDA will be less than or equal to X. (Output)

Required Arguments

X — Argument for which the noncentral beta cumulative distribution function is to be evaluated. (Input)
X must be non-negative and less than or equal to 1.

SHAPE1 — First shape parameter of the noncentral beta distribution. (Input)
SHAPE1 must be positive.

SHAPE2 — Second shape parameter of the noncentral beta distribution. (Input)
SHAPE2 must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: BETNDF (X, SHAPE1, SHAPE2, LAMBDA)
Specific: The specific interface names are S_BETNDF and D_BETNDF.

Description

The noncentral beta distribution is a generalization of the beta distribution. If Z is a noncentral chi-square random variable with noncentrality parameter λ and $2\alpha_1$ degrees of freedom, and Y is a chi-square random variable with $2\alpha_2$ degrees of freedom which is statistically independent of Z, then

$$X = \frac{Z}{Z+Y} = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

is a noncentral beta-distributed random variable and

$$F = \frac{\alpha_2 Z}{\alpha_1 Y} = \frac{\alpha_2 X}{\alpha_1 (1 - X)}$$

is a noncentral *F*-distributed random variable. The CDF for noncentral beta variable *X* can thus be simply defined in terms of the noncentral *F* CDF:

$$CDF_{ncβ}(x,\alpha_1,\alpha_2,\lambda) = CDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)$$

where $CDF_{ncβ}(x,\alpha_1,\alpha_2,\lambda)$ is a noncentral beta CDF with *x* = X, α_1 = SHAPE1, α_2 = SHAPE2, and noncentrality parameter λ = LAMBDA; $CDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)$ is a noncentral *F* CDF with argument *f*, numerator and denominator degrees of freedom $2\alpha_1$ and $2\alpha_2$ respectively, and noncentrality parameter λ and:

$$f = \frac{\alpha_2}{\alpha_1}\frac{x}{1-x}; \quad x = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

(See documentation for function FNDF for a discussion of how the noncentral *F* CDF is defined and calculated.)

With a noncentrality parameter of zero, the noncentral beta distribution is the same as the beta distribution.

Example

This example traces out a portion of a noncentral beta distribution with parameters SHAPE1 = 50, SHAPE2 = 5, and LAMBDA = 10.

```
      USE UMACH_INT
      USE BETNDF_INT
      USE FNDF_INT
      IMPLICIT NONE
      INTEGER NOUT, I
      REAL X, LAMBDA, SHAPE1, SHAPE2, &
         BCDFV, FCDFV, F(8)

      DATA F /0.0, 0.4, 0.8, 1.2, &
             1.6, 2.0, 2.8, 4.0 /

      CALL UMACH (2, NOUT)
      SHAPE1 = 50.0
      SHAPE2 = 5.0
      LAMBDA = 10.0

      WRITE (NOUT,'(/"  SHAPE1: ", F4.0, &
        & ";  SHAPE2: ", F4.0, &
        & ";  LAMBDA: ", F4.0 // &
        & 6x,"X",6x,"NCBETCDF(X)",3x,"NCBETCDF(X)"/ &
        & 14x,"expected")') SHAPE1, SHAPE2, LAMBDA

      DO I = 1, 8
         X = (SHAPE1*F(I)) / (SHAPE1*F(I) + SHAPE2)
         FCDFV = FNDF(F(I),2*SHAPE1,2*SHAPE2,LAMBDA)
```

```
            BCDFV = BETNDF(X, SHAPE1, SHAPE2, LAMBDA)
            WRITE (NOUT,'(2X, F8.6, 2(2X, E12.6))') &
               X, FCDFV, BCDFV
         END DO
         END
```

Output

```

      SHAPE1:  50.;  SHAPE2:   5.;  LAMBDA:  10.

          X      NCBETCDF(X)   NCBETCDF(X)
                   expected
      0.000000  0.000000E+00  0.000000E+00
      0.800000  0.488790E-02  0.488790E-02
      0.888889  0.202633E+00  0.202633E+00
      0.923077  0.521143E+00  0.521143E+00
      0.941176  0.733853E+00  0.733853E+00
      0.952381  0.850413E+00  0.850413E+00
      0.965517  0.947125E+00  0.947125E+00
      0.975610  0.985358E+00  0.985358E+00
```

BETNIN

This function evaluates the inverse of the noncentral beta cumulative distribution function (CDF).

Function Return Value

BETNIN — Function value, the value of the inverse of the cumulative distribution function evaluated at P. The probability that a noncentral beta random variable takes a value less than or equal to BETNIN is P. (Output)

Required Arguments

P — Probability for which the inverse of the noncentral beta cumulative distribution function is to be evaluated. (Input)
P must be non-negative and less than or equal to 1.

SHAPE1 — First shape parameter of the noncentral beta distribution. (Input)
SHAPE1 must be positive.

SHAPE2 — Second shape parameter of the noncentral beta distribution. (Input)
SHAPE2 must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: BETNIN (P, SHAPE1, SHAPE2, LAMBDA)
Specific: The specific interface names are S_BETNIN and D_BETNIN.

Description

The noncentral beta distribution is a generalization of the beta distribution. If Z is a noncentral chi-square random variable with noncentrality parameter λ and $2\alpha_1$ degrees of freedom, and Y is a chi-square random variable with $2\alpha_2$ degrees of freedom which is statistically independent of Z, then

$$X = \frac{Z}{Z+Y} = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

is a noncentral beta-distributed random variable and

$$F = \frac{\alpha_2 Z}{\alpha_1 Y} = \frac{\alpha_2 X}{\alpha_1 (1-X)}$$

is a noncentral *F*-distributed random variable. The CDF for noncentral beta variable *X* can thus be simply defined in terms of the noncentral *F* CDF:

$$p = CDF_{nc\beta}(x,\alpha_1,\alpha_2,\lambda) = CDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)$$

where $CDF_{nc\beta}(x,\alpha_1,\alpha_2,\lambda)$ is a noncentral beta CDF with *x* = **X**, α_1 = **SHAPE1**, α_2 = **SHAPE2**, and noncentrality parameter λ = **LAMBDA**; $CDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)$ is a noncentral *F* CDF with argument *f*, numerator and denominator degrees of freedom $2\alpha_1$ and $2\alpha_2$ respectively, and noncentrality parameter λ; *p* = the probability that $F \le f$ = the probability that $X \le x$ and:

$$f = \frac{\alpha_2}{\alpha_1}\frac{x}{1-x}; \quad x = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

(See the documentation for function **FNDF** for a discussion of how the noncentral *F* CDF is defined and calculated.) The correspondence between the arguments of function **BETNIN(P,SHAPE1,SHAPE2,LAMBDA)** and the variables in the above equations is as follows: α_1 = **SHAPE1**, α_2 = **SHAPE2**, λ = **LAMBDA**, and *p* = **P**.

Function **BETNIN** evaluates

$$x = CDF^{-1}{}_{nc\beta}(p,\alpha_1,\alpha_2,\lambda)$$

by first evaluating

$$f = CDF^{-1}{}_{ncF}(p,2\alpha_1,2\alpha_2,\lambda)$$

and then solving for *x* using

$$x = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

(See the documentation for function **FNIN** for a discussion of how the inverse noncentral *F* CDF is calculated.)

Example

This example traces out a portion of an inverse noncentral beta distribution with parameters SHAPE1 = 50, SHAPE2 = 5, and LAMBDA= 10.

```
      USE UMACH_INT
      USE BETNDF_INT
      USE BETNIN_INT
      USE UMACH_INT
      IMPLICIT NONE

      INTEGER  :: NOUT, I
      REAL     :: SHAPE1 = 50.0, SHAPE2=5.0, LAMBDA=10.0
      REAL     :: X, CDF, CDFINV
      REAL     :: F0(8)=(/ 0.0, .4, .8, 1.2, 1.6, 2.0, 2.8, 4.0 /)

      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(/"  SHAPE1: ", F4.0, "  SHAPE2: ", F4.0,'// &
         '" LAMBDA: ", F4.0 // ' //                             &
         '"       X         P = CDF(X)     CDFINV(P)")')       &
         SHAPE1, SHAPE2, LAMBDA
      DO I = 1, 8
         X = (SHAPE1*F0(I))/(SHAPE2 + SHAPE1*F0(I))
         CDF = BETNDF(X, SHAPE1, SHAPE2, LAMBDA)
         CDFINV = BETNIN(CDF, SHAPE1, SHAPE2, LAMBDA)
         WRITE (NOUT,'(3(2X, E12.6))') X, CDF, CDFINV
      END DO
      END
```

Output

```
  SHAPE1:  50.  SHAPE2:   5. LAMBDA:  10.

       X         P = CDF(X)     CDFINV(P)
0.000000E+00  0.000000E+00  0.000000E+00
0.800000E+00  0.488791E-02  0.800000E+00
0.888889E+00  0.202633E+00  0.888889E+00
0.923077E+00  0.521144E+00  0.923077E+00
0.941176E+00  0.733853E+00  0.941176E+00
0.952381E+00  0.850413E+00  0.952381E+00
0.965517E+00  0.947125E+00  0.965517E+00
0.975610E+00  0.985358E+00  0.975610E+00
```

BETNPR

This function evaluates the noncentral beta probability density function.

Function Return Value

BETNPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the noncentral beta probability density function is to be evaluated. (Input)
X must be non-negative and less than or equal to 1.

SHAPE1 — First shape parameter of the noncentral beta distribution. (Input)
SHAPE1 must be positive.

SHAPE2 — Second shape parameter of the noncentral beta distribution. (Input)
SHAPE2 must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: BETNPR (X, SHAPE1, SHAPE2, LAMBDA)
Specific: The specific interface names are S_BETNPR and D_BETNPR.

Description

The noncentral beta distribution is a generalization of the beta distribution. If Z is a noncentral chi-square random variable with noncentrality parameter λ and $2\alpha_1$ degrees of freedom, and Y is a chi-square random variable with $2\alpha_2$ degrees of freedom which is statistically independent of Z, then

$$X = \frac{Z}{Z+Y} = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2}$$

is a noncentral beta-distributed random variable and

$$F = \frac{\alpha_2 Z}{\alpha_1 Y} = \frac{\alpha_2 X}{\alpha_1 (1 - X)}$$

is a noncentral *F*-distributed random variable. The PDF for noncentral beta variable *X* can thus be simply defined in terms of the noncentral *F* PDF:

$$PDF_{nc\beta}(x,\alpha_1,\alpha_2,\lambda) = PDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)\frac{df}{dx}$$

where $PDF_{nc\beta}(x,\alpha_1,\alpha_2,\lambda)$ is a noncentral beta PDF with *x* = X, α_1 = SHAPE1, α_2 = SHAPE2, and noncentrality parameter λ = LAMBDA; $PDF_{ncF}(f,2\alpha_1,2\alpha_2,\lambda)$ is a noncentral *F* PDF with argument *f*, numerator and denominator degrees of freedom $2\alpha_1$ and $2\alpha_2$ respectively, and noncentrality parameter λ; and:

$$f = \frac{\alpha_2}{\alpha_1}\frac{x}{1-x}; \quad x = \frac{\alpha_1 f}{\alpha_1 f + \alpha_2};$$

$$\frac{df}{dx} = \frac{(\alpha_2 + \alpha_1 f)^2}{\alpha_1 \alpha_2} = \frac{\alpha_2}{\alpha_1}\frac{1}{(1-x)^2}$$

(See the documentation for function FNPR for a discussion of how the noncentral *F* PDF is defined and calculated.)

With a noncentrality parameter of zero, the noncentral beta distribution is the same as the beta distribution.

Example

This example traces out a portion of a noncentral beta distribution with parameters SHAPE1 = 50, SHAPE2 = 5, and LAMBDA = 10.

```
      USE UMACH_INT
      USE BETNPR_INT
      USE FNPR_INT
      IMPLICIT NONE

      INTEGER NOUT, I
      REAL X, LAMBDA, SHAPE1, SHAPE2, &
         BPDFV, FPDFV, DBETNPR, DFNPR, F(8), &
         BPDFVEXPECT, DFDX

      DATA F /0.0, 0.4, 0.8, 3.2, 5.6, 8.8, 14.0, 18.0/

      CALL UMACH (2, NOUT)
      SHAPE1 = 50.0
      SHAPE2 = 5.0
      LAMBDA = 10.0

      WRITE (NOUT,'(/"  SHAPE1: ", F4.0, ";  SHAPE2: ", F4.0, ";  '// &
         'LAMBDA: ", F4.0 // 6x,"X",6x,"NCBETPDF(X)",3x,"NCBETPDF'// &
```

```
            '(X)",/     14x,"expected")') SHAPE1, SHAPE2, LAMBDA

         DO I = 1, 8
            X = (SHAPE1*F(I)) / (SHAPE1*F(I) + SHAPE2)
            DFDX = (SHAPE2/SHAPE1) / (1.0 - X)**2
            FPDFV = FNPR(F(I),2*SHAPE1,2*SHAPE2,LAMBDA)
            BPDFVEXPECT = DFDX * FPDFV
            BPDFV = BETNPR(X, SHAPE1, SHAPE2, LAMBDA)
            WRITE (NOUT,'(2X, F8.6, 2(2X, E12.6))')  X, BPDFVEXPECT, BPDFV
         END DO
         END
```

Output

```
      SHAPE1:  50.;  SHAPE2:   5.;  LAMBDA:  10.

          X      NCBETPDF(X)   NCBETPDF(X)
                   expected
      0.000000  0.000000E+00  0.000000E+00
      0.800000  0.243720E+00  0.243720E+00
      0.888889  0.658624E+01  0.658624E+01
      0.969697  0.402367E+01  0.402365E+01
      0.982456  0.919544E+00  0.919542E+00
      0.988764  0.219100E+00  0.219100E+00
      0.992908  0.436654E-01  0.436647E-01
      0.994475  0.175215E-01  0.175217E-01
```

BNRDF

This function evaluates the bivariate normal cumulative distribution function.

Function Return Value

BNRDF — Function value, the probability that a bivariate normal random variable with correlation **RHO** takes a value less than or equal to **X** and less than or equal to **Y**. (Output)

Required Arguments

X — One argument for which the bivariate normal distribution function is to be evaluated. (Input)

Y — The other argument for which the bivariate normal distribution function is to be evaluated. (Input)

RHO — Correlation coefficient. (Input)

FORTRAN 90 Interface

Generic: **BNRDF** (**X**, **Y**, **RHO**)
Specific: The specific interface names are **S_BNRDF** and **D_BNRDF**.

FORTRAN 77 Interface

Single: **BNRDF** (**X**, **Y**, **RHO**)
Double: The double precision name is **DBNRDF**.

Description

Function **BNRDF** evaluates the cumulative distribution function F of a bivariate normal distribution with means of zero, variances of one, and correlation of **RHO**; that is, with ρ = **RHO**, and $|\rho| < 1$,

$$F(x,y) = \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{-\infty}^{x}\int_{-\infty}^{y}\exp\left(-\frac{u^2-2\rho uv+v^2}{2(1-\rho^2)}\right)du\,dv$$

To determine the probability that $U \le u_0$ and $V \le v_0$, where $(U, V)^T$ is a bivariate normal random variable with mean $\mu = (\mu_U, \mu_V)^T$ and variance-covariance matrix

$$\sum = \begin{pmatrix} \sigma_U^2 & \sigma_{UV} \\ \sigma_{UV} & \sigma_V^2 \end{pmatrix}$$

transform $(U, V)^T$ to a vector with zero means and unit variances. The input to **BNRDF** would be $X = (u_0 - \mu_U)/\sigma_U$, $Y = (v_0 - \mu_V)/\sigma_V$, and $\rho = \sigma_{UV}/(\sigma_U\sigma_V)$.

Function **BNRDF** uses the method of Owen (1962, 1965). Computation of Owen's T-function is based on code by M. Patefield and D. Tandy (2000). For $|\rho| = 1$, the distribution function is computed based on the univariate statistic, $Z = \min(x, y)$, and on the normal distribution function ANORDF.

Example

Suppose (X, Y) is a bivariate normal random variable with mean (0, 0) and variance-covariance matrix

$$\begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

In this example, we find the probability that X is less than –2.0 and Y is less than 0.0.

```
      USE BNRDF_INT
      USE UMACH_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       P, RHO, X, Y
!
      CALL UMACH (2, NOUT)
      X   = -2.0
      Y   = 0.0
      RHO = 0.9
      P   = BNRDF(X,Y,RHO)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is less than -2.0  and Y ', &
             'is less than 0.0 is ', F6.4)
      END
```

Output

```
 The probability that X is less than –2.0 and Y is less than 0.0 is 0.0228
```

CHIDF

This function evaluates the chi-squared cumulative distribution function.

Function Return Value

CHIDF — Function value, the probability that a chi-squared random variable takes a value less than or equal to **CHSQ**. (Output)

Required Arguments

CHSQ — Argument for which the chi-squared distribution function is to be evaluated. (Input)

DF — Number of degrees of freedom of the chi-squared distribution. (Input)
DF must be positive.

Optional Arguments

COMPLEMENT — Logical. If **.TRUE.**, the complement of the chi-squared cumulative distribution function is evaluated. If **.FALSE.**, the chi-squared cumulative distribution function is evaluated. (Input)
See the Description section for further details on the use of **COMPLEMENT**.
Default: **COMPLEMENT** = **.FALSE.**.

FORTRAN 90 Interface

Generic: **CHIDF** (**CHSQ**, **DF** [, ...])
Specific: The specific interface names are **S_CHIDF** and **D_CHIDF**.

FORTRAN 77 Interface

Single: **CHIDF** (**CHSQ**, **DF**)
Double: The double precision name is **DCHIDF**.

Description

Function **CHIDF** evaluates the cumulative distribution function, *F*, of a chi-squared random variable with **DF** degrees of freedom, that is, with ν = **DF**, and x = **CHSQ**,

$$F(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}\int_0^x e^{-t/2}t^{\nu/2-1}dt$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function at the point x is the probability that the random variable takes a value less than or equal to x.

For $\nu > \nu_{max}$ = {343 for double precision, 171 for single precision}, **CHIDF** uses the Wilson-Hilferty approximation (Abramowitz and Stegun [A&S] 1964, equation 26.4.17) for p in terms of the normal CDF, which is evaluated using function ANORDF.

For $\nu \leq \nu_{max}$, **CHIDF** uses series expansions to evaluate p: for $x < \nu$, **CHIDF** calculates p using A&S series 6.5.29, and for $x \geq \nu$, **CHIDF** calculates p using the continued fraction expansion of the incomplete gamma function given in A&S equation 6.5.31.

If **COMPLEMENT** = **.TRUE.**, the value of **CHIDF** at the point x is 1 **-** p, where 1 **-** p is the probability that the random variable takes a value greater than x. In those situations where the desired end result is 1 **-** p, the user can achieve greater accuracy in the right tail region by using the result returned by **CHIDF** with the optional argument **COMPLEMENT** set to **.TRUE.** rather than by using 1 **-** p where p is the result returned by **CHIDF** with **COMPLEMENT** set to **.FALSE.**.

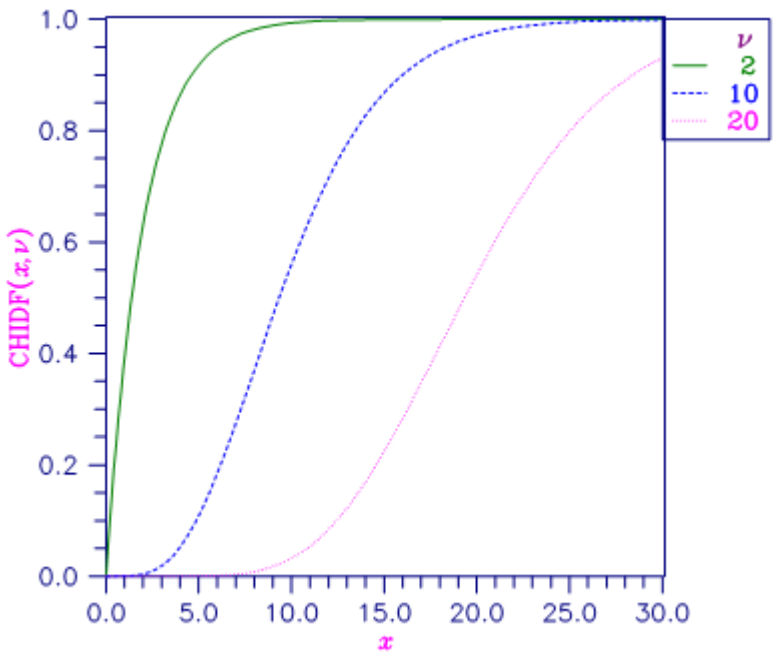


Figure 29, Chi-Squared Distribution Function

Comments

Informational Errors

Type	Code	Description
1	1	Since the input argument, CHSQ, is less than zero, the distribution function is zero at CHSQ.
2	3	The normal distribution is used for large degrees of freedom. However, it has produced underflow. Therefore, the probability, CHIDF, is set to zero.

Example

Suppose *X* is a chi-squared random variable with 2 degrees of freedom. In this example, we find the probability that *X* is less than 0.15 and the probability that *X* is greater than 3.0.

```
      USE CHIDF_INT
      USE UMACH_INT
      IMPLICIT   NONE

      INTEGER    NOUT
      REAL       CHSQ, DF, P

      CALL UMACH (2, NOUT)
      DF   = 2.0
      CHSQ = 0.15
      P    = CHIDF(CHSQ,DF)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that chi-squared with 2 df is less ', &
             'than 0.15 is ', F6.4)
      CHSQ = 3.0
      P    = CHIDF(CHSQ,DF, complement=.true.)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that chi-squared with 2 df is greater ' &
            , 'than 3.0 is ', F6.4)
      END
```

Output

```
 The probability that chi-squared with 2 df is less than 0.15 is 0.0723
 The probability that chi-squared with 2 df is greater than 3.0 is 0.2231
```

CHIIN

This function evaluates the inverse of the chi-squared cumulative distribution function.

Function Return Value

CHIIN — Function value. (Output)
The probability that a chi-squared random variable takes a value less than or equal to **CHIIN** is **P**.

Required Arguments

P — Probability for which the inverse of the chi-squared distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

DF — Number of degrees of freedom of the chi-squared distribution. (Input)
DF must be greater than or equal to 0.5.

FORTRAN 90 Interface

Generic: **CHIIN** (**P**, **DF**)
Specific: The specific interface names are **S_CHIIN** and **D_CHIIN**.

FORTRAN 77 Interface

Single: **CHIIN** (**P**, **DF**)
Double: The double precision name is **DCHIIN**.

Description

Function **CHIIN** evaluates the inverse distribution function of a chi-squared random variable with **DF** degrees of freedom, that is, with P = **P** and ν = **DF**, it determines x (equal to **CHIIN**(**P**, **DF**)), such that

$$P = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}\int_0^x e^{-t/2}t^{\nu/2-1}dt$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to x is P.

For $\nu < 40$, **CHIIN** uses bisection (if $\nu \leq 2$ or $P > 0.98$) or regula falsi to find the point at which the chi-squared distribution function is equal to P. The distribution function is evaluated using routine **CHIDF**.

For $40 \leq \nu < 100$, a modified Wilson-Hilferty approximation (Abramowitz and Stegun 1964, equation 26.4.18) to the normal distribution is used, and routine **ANORIN** is used to evaluate the inverse of the normal distribution function. For $\nu \geq 100$, the ordinary Wilson-Hilferty approximation (Abramowitz and Stegun 1964, equation 26.4.17) is used.

Comments

Informational Error

Type	Code	Description
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Example

In this example, we find the 99-th percentage points of a chi-squared random variable with 2 degrees of freedom and of one with 64 degrees of freedom.

```
      USE UMACH_INT
      USE CHIIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DF, P, X
!
      CALL UMACH (2, NOUT)
      P  = 0.99
      DF = 2.0
      X  = CHIIN(P,DF)
      WRITE (NOUT,99998) X
99998 FORMAT (' The 99-th percentage point of chi-squared with  2 df ' &
            , 'is ', F7.3)
      DF = 64.0
      X  = CHIIN(P,DF)
      WRITE (NOUT,99999) X
99999 FORMAT (' The 99-th percentage point of chi-squared with 64 df ' &
            , 'is ', F7.3)
      END
```

Output

```
 The 99-th percentage point of chi-squared with  2 df is   9.210
 The 99-th percentage point of chi-squared with 64 df is  93.217
```

CHIPR

This function evaluates the chi-squared probability density function.

Function Return Value

CHIPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the chi-squared probability density function is to be evaluated. (Input)

DF — Number of degrees of freedom of the chi-squared distribution. (Input)

FORTRAN 90 Interface

Generic: **CHIPR** (**X**, **DF**)
Specific: The specific interface names are **S_CHIPR** and **D_CHIPR**.

FORTRAN 77 Interface

Single: **CHIPR** (**X**, **DF**)
Double: The double precision name is **DCHIPR**.

Description

The function **CHIPR** evaluates the chi-squared probability density function. The chi-squared distribution is a special case of the gamma distribution and is defined as

$$f\left(x|v\right) = \Gamma\left(x|v/2,2\right) = \frac{1}{2^{v/2}\Gamma\left(v/2\right)}\left(x\right)^{v/2-1}e^{-\frac{x}{2}}, \quad x,v > 0$$

Example

In this example, we evaluate the probability function at **X** = 3.0, **DF** = 5.0.

```
      USE UMACH_INT
```

```
      USE CHIPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, DF, PR
      CALL UMACH(2, NOUT)
      X = 3.0
      DF = 5.0
      PR = CHIPR(X, DF)
      WRITE (NOUT, 99999) X, DF, PR
99999 FORMAT (' CHIPR(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 CHIPR(3.00, 5.00) = 0.1542
```

CSNDF

This function evaluates the noncentral chi-squared cumulative distribution function.

Function Return Value

CSNDF — Function value, the probability that a noncentral chi-squared random variable takes a value less than or equal to **CHSQ**. (Output)

Required Arguments

CHSQ — Argument for which the noncentral chi-squared cumulative distribution function is to be evaluated. (Input)

DF —Number of degrees of freedom of the noncentral chi-squared cumulative distribution. (Input)
DF must be positive and less than or equal to 200,000.

ALAM — The noncentrality parameter. (Input)
ALAM must be nonnegative, and **ALAM** + **DF** must be less than or equal to 200,000.

FORTRAN 90 Interface

Generic: **CSNDF** (**CHSQ**, **DF**, **ALAM**)
Specific: The specific interface names are **S_CSNDF** and **D_CSNDF**.

FORTRAN 77 Interface

Single: **CSNDF** (**CHSQ**, **DF**, **ALAM**)
Double: The double precision name is **DCSNDF**.

Description

Function **CSNDF** evaluates the cumulative distribution function of a noncentral chi-squared random variable with **DF** degrees of freedom and noncentrality parameter **ALAM**, that is, with ν = **DF**, λ = **ALAM**, and x = **CHSQ**,

$$F(x|\nu,\lambda) = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^i}{i!} \int_0^x \frac{(t/2)^{(\nu+2i)/2} e^{-t/2}}{t\ \Gamma(\frac{\nu+2i}{2})}\ dt$$

where $\Gamma(\cdot)$ is the gamma function. This is a series of central chi-squared distribution functions with Poisson weights. The value of the distribution function at the point *x* is the probability that the random variable takes a value less than or equal to *x*.

The noncentral chi-squared random variable can be defined by the distribution function above, or alternatively and equivalently, as the sum of squares of independent normal random variables. If Y_i have independent normal distributions with means μ_i and variances equal to one and

$$X = \sum_{i=1}^{n} Y_i^2$$

then *X* has a noncentral chi-squared distribution with *n* degrees of freedom and noncentrality parameter equal to

$$\sum_{i=1}^{n} \mu_i^2$$

With a noncentrality parameter of zero, the noncentral chi-squared distribution is the same as the chi-squared distribution.

Function **CSNDF** determines the point at which the Poisson weight is greatest, and then sums forward and backward from that point, terminating when the additional terms are sufficiently small or when a maximum of 1000 terms have been accumulated. The recurrence relation 26.4.8 of Abramowitz and Stegun (1964) is used to speed the evaluation of the central chi-squared distribution functions.

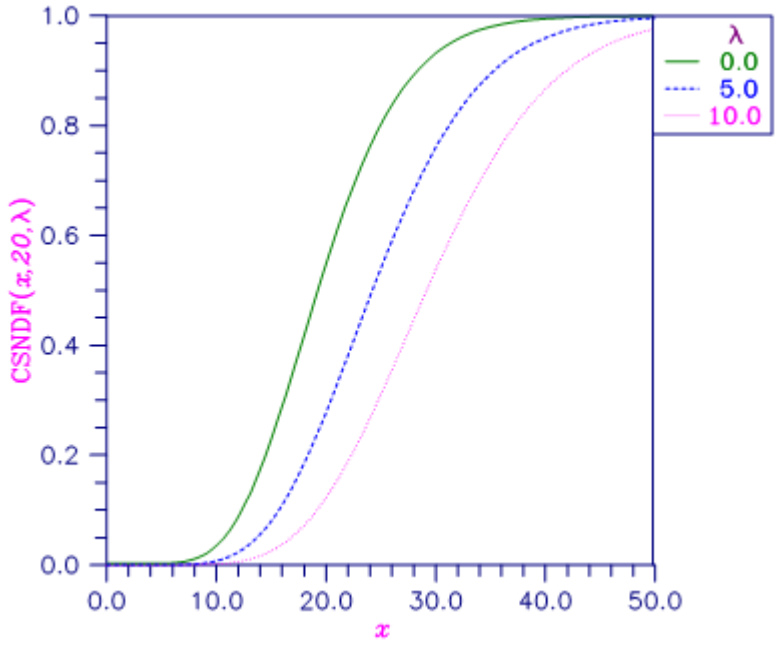


Figure 30, Noncentral Chi-squared Distribution Function

Example

In this example, **CSNDF** is used to compute the probability that a random variable that follows the noncentral chi-squared distribution with noncentrality parameter of 1 and with 2 degrees of freedom is less than or equal to 8.642.

```
      USE UMACH_INT
      USE CSNDF_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       ALAM, CHSQ, DF, P
!
      CALL UMACH (2, NOUT)
      DF   = 2.0
      ALAM = 1.0
      CHSQ = 8.642
      P    = CSNDF(CHSQ,DF,ALAM)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that a noncentral chi-squared random', &
             /, ' variable with 2 df and noncentrality 1.0 is less', &
             /, ' than 8.642 is ', F5.3)
      END
```

Output

```
 The probability that a noncentral chi-squared random
 variable with 2 df and noncentrality 1.0 is less
 than 8.642 is 0.950
```

CSNIN

This function evaluates the inverse of the noncentral chi-squared cumulative function.

Function Return Value

CSNIN — Function value. (Output)
The probability that a noncentral chi-squared random variable takes a value less than or equal to **CSNIN** is **P**.

Required Arguments

P — Probability for which the inverse of the noncentral chi-squared cumulative distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

DF — Number of degrees of freedom of the noncentral chi-squared distribution. (Input)
DF must be greater than or equal to 0.5 and less than or equal to 200,000.

ALAM — The noncentrality parameter. (Input)
ALAM must be nonnegative, and **ALAM** + **DF** must be less than or equal to 200,000.

FORTRAN 90 Interface

Generic: **CSNIN** (**P**, **DF**, **ALAM**)
Specific: The specific interface names are **S_CSNIN** and **D_CSNIN**.

FORTRAN 77 Interface

Single: **CSNIN** (**P**, **DF**, **ALAM**)
Double: The double precision name is **DCSNIN**.

Description

Function **CSNIN** evaluates the inverse distribution function of a noncentral chi-squared random variable with **DF** degrees of freedom and noncentrality parameter **ALAM**; that is, with P = **P**, v = **DF**, and =λ = **ALAM**, it determines c_0 (= **CSNIN(P,DF,ALAM)**), such that

$$P = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^i}{i!} \int_0^{c_0} \frac{x^{(v+2i)/2-1} e^{-x/2}}{2^{(v+2i)/2}\,\Gamma(\frac{v+2i}{2})} dx$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to c_0 is P .

Function **CSNIN** uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine CSNDF. See **CSNDF** for an alternative definition of the noncentral chi-squared random variable in terms of normal random variables.

Comments

Informational Error

Type	Code	Description
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Example

In this example, we find the 95-th percentage point for a noncentral chi-squared random variable with 2 degrees of freedom and noncentrality parameter 1.

```
      USE CSNIN_INT
      USE UMACH_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       ALAM, CHSQ, DF, P
!
      CALL UMACH (2, NOUT)
      DF   = 2.0
      ALAM = 1.0
      P    = 0.95
      CHSQ = CSNIN(P,DF,ALAM)
      WRITE (NOUT,99999) CHSQ
!
99999 FORMAT (' The 0.05 noncentral chi-squared critical value is ', &
            F6.3, '.')
!
      END
```

Output

```
 The 0.05 noncentral chi-squared critical value is  8.642.
```

CSNPR

This function evaluates the noncentral chi-squared probability density function.

Function Return Value

CSNPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the noncentral chi-squared probability density function is to be evaluated. (Input)
X must be non-negative.

DF — Number of degrees of freedom of the noncentral chi-squared distribution. (Input)
DF must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: CSNPR (X, DF, LAMBDA)
Specific: The specific interface names are S_CSNPR and D_CSNPR.

Description

The noncentral chi-squared distribution is a generalization of the chi-squared distribution. If $\{X_i\}$ are k independent, normally distributed random variables with means μ_i and variances σ^2_i, then the random variable:

$$X = \sum_{i=1}^{k} \left(\frac{X_i}{\sigma_i} \right)^2$$

is distributed according to the noncentral chi-squared distribution. The noncentral chi-squared distribution has two parameters: k which specifies the number of degrees of freedom (i.e. the number of X_i), and λ, which is related to the mean of the random variables X_i by:

$$\lambda = \sum_{i=1}^{k} \left(\frac{\mu_i}{\sigma_i} \right)^2$$

The noncentral chi-squared distribution is equivalent to a (central) chi-squared distribution with $k + 2i$ degrees of freedom, where i is the value of a Poisson distributed random variable with parameter $\boldsymbol{\lambda}/2$. Thus, the probability density function is given by:

$$F\left(x,k,\lambda\right) = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}\left(\lambda/2\right)^i}{i!} f\left(x,k+2i\right)$$

where the (central) chi-squared PDF $f(x, k)$ is given by:

$$f\left(x,k\right) = \frac{\left(x/2\right)^{k/2} e^{-x/2}}{x\Gamma\left(k/2\right)} \text{ for } x > 0, \text{ else } 0$$

where $\Gamma(\cdot)$ is the gamma function. The above representation of $\mathbf{F}(x, k, \boldsymbol{\lambda})$ can be shown to be equivalent to the representation:

$$F\left(x,k,\lambda\right) = \frac{e^{-\left(\lambda+x\right)/2}\left(x/2\right)^{k/2}}{x} \sum_{i=0}^{\infty} \phi_i$$

$$\phi_i = \frac{\left(\lambda x/4\right)^i}{i!\Gamma\left(k/2+i\right)}$$

Function **CSNPR** (**X**, **DF**, **LAMBDA**) evaluates the probability density function of a noncentral chi-squared random variable with **DF** degrees of freedom and noncentrality parameter **LAMBDA**, corresponding to k = **DF**, $\boldsymbol{\lambda}$ = **LAMBDA**, and x = **X**.

Function **CSNDF** (**X**, **DF**, **LAMBDA**) evaluates the cumulative distribution function incorporating the above probability density function.

With a noncentrality parameter of zero, the noncentral chi-squared distribution is the same as the central chi-squared distribution.

Example

This example calculates the noncentral chi-squared distribution for a distribution with 100 degrees of freedom and noncentrality parameter = 40.

```
      USE UMACH_INT
```

```
      USE CSNPR_INT
      IMPLICIT NONE

      INTEGER :: NOUT, I
      REAL    :: X(6)=(/ 0.0, 8.0, 40.0, 136.0, 280.0, 400.0 /)
      REAL    :: LAMBDA=40.0, DF=100.0, PDFV

      CALL UMACH (2, NOUT)
      WRITE (NOUT,'(//"DF: ", F4.0, "  LAMBDA: ", F4.0 //'// &
               ' "   X        PDF(X)")') DF, LAMBDA
      DO I = 1, 6
         PDFV = CSNPR(X(I), DF, LAMBDA)
         WRITE (NOUT,'(1X, F5.0, 2X, E12.5)') X(I), PDFV
      END DO
      END
```

Output

```
 DF: 100.  LAMBDA:  40.

    X        PDF(X)
    0.   0.00000E+00
    8.   0.00000E+00
   40.   0.34621E-13
  136.   0.21092E-01
  280.   0.40027E-09
  400.   0.11250E-21
```

EXPDF

This function evaluates the exponential cumulative distribution function.

Function Return Value

EXPDF — Function value, the probability that an exponential random variable takes a value less than or equal to **X**. (Output)

Required Arguments

X — Argument for which the exponential cumulative distribution function is to be evaluated. (Input)

B — Scale parameter of the exponential distribution function. (Input)

FORTRAN 90 Interface

Generic: **EXPDF** (**X**, **B**)
Specific: The specific interface names are **S_EXPDF** and **D_EXPDF**.

FORTRAN 77 Interface

Single: **EXPDF** (**X**, **B**)
Double: The double precision name is **DEXPDF**.

Description

The function **EXPDF** evaluates the exponential cumulative distribution function (CDF), defined:

$$F(x|b) = \int_0^x f(t|b)\ dt = 1 - e^{-\frac{x}{b}}$$

where

$$f(x|b) = \frac{1}{b} e^{-\frac{x}{b}}$$

is the exponential probability density function (PDF).

Example

In this example, we evaluate the probability function at **X** = 2.0, **B** = 1.0.

```
      USE UMACH_INT
      USE EXPDF_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, B, PR
      CALL UMACH(2, NOUT)
      X = 2.0
      B = 1.0
      PR = EXPDF(X, B)
      WRITE (NOUT, 99999) X, B, PR
99999 FORMAT (' EXPDF(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXPDF(2.00, 1.00) = 0.8647
```

EXPIN

This function evaluates the inverse of the exponential cumulative distribution function.

Function Return Value

EXPIN — Function value, the value of the inverse of the cumulative distribution function. (Output)

Required Arguments

P — Probability for which the inverse of the exponential distribution function is to be evaluated. (Input)

B — Scale parameter of the exponential distribution function. (Input)

FORTRAN 90 Interface

Generic: `EXPIN` (P, B)
Specific: The specific interface names are `S_EXPIN` and `D_EXPIN`.

FORTRAN 77 Interface

Single: `EXPIN` (P, B)
Double: The double precision name is `DEXPIN`.

Description

The function `EXPIN` evaluates the inverse distribution function of an exponential random variable with scale parameter b = B.

Example

In this example, we evaluate the inverse probability function at P = 0.8647, B = 1.0.

```
      USE UMACH_INT
      USE EXPIN_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, B, P
```

```
      CALL UMACH(2, NOUT)

      P = 0.8647
      B = 1.0
      X = EXPIN(P, B)
      WRITE (NOUT, 99999) P, B, X
99999 FORMAT (' EXPIN(', F6.4, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXPIN(0.8647, 1.00) = 2.0003
```

EXPPR

This function evaluates the exponential probability density function.

Function Return Value

EXPPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the exponential probability density function is to be evaluated. (Input)

B — Scale parameter of the exponential probability density function. (Input)

FORTRAN 90 Interface

Generic: **EXPPR** (**X**, **B**)
Specific: The specific interface names are **S_EXPPR** and **D_EXPPR**.

FORTRAN 77 Interface

Single: **EXPPR** (**X**, **B**)
Double: The double precision name is **DEXPPR**.

Description

The function **EXPPR** evaluates the exponential probability density function. The exponential distribution is a special case of the gamma distribution and is defined as

$$f\left(x \mid b\right) = \Gamma\left(x \mid 1,b\right) = \frac{1}{b}e^{-\frac{x}{b}}, \quad x,b > 0$$

This relationship is used in the computation of $f(x|b)$.

Example

In this example, we evaluate the probability function at **X** = 2.0, **B** = 1.0.

```
      USE UMACH_INT
      USE EXPPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, B, PR
      CALL UMACH(2, NOUT)
      X = 2.0
      B = 1.0
      PR = EXPPR(X, B)
      WRITE (NOUT, 99999) X, B, PR
99999 FORMAT (' EXPPR(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXPPR(2.00, 1.00) = 0.1353
```

EXVDF

This function evaluates the extreme value cumulative distribution function.

Function Return Value

EXVDF — Function value, the probability that an extreme value random variable takes a value less than or equal to **X**. (Output)

Required Arguments

X — Argument for which the extreme value cumulative distribution function is to be evaluated. (Input)

AMU — Location parameter of the extreme value probability distribution function. (Input)

BETA — Scale parameter of the extreme value probability distribution function. (Input)

FORTRAN 90 Interface

Generic: **EXVDF** (**X**, **AMU**, **BETA**)
Specific: The specific interface names are **S_EXVDF** and **D_EXVDF**.

FORTRAN 77 Interface

Single: **EXVDF** (**X**, **AMU**, **BETA**)
Double: The double precision name is **DEXVDF**.

Description

The function **EXVDF** evaluates the extreme value cumulative distribution function, defined as

$$F\left(x \mid \mu,\beta\right) = 1 - e^{-e^{\frac{x-\mu}{\beta}}}$$

The extreme value distribution is also known as the Gumbel minimum distribution.

Example

In this example, we evaluate the probability function at X = 1.0, AMU = 0.0, BETA = 1.0.

```
      USE UMACH_INT
      USE EXVDF_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, B, PR
      CALL UMACH(2, NOUT)
      X = 1.0
      AMU = 0.0
      B = 1.0
      PR = EXVDF(X, AMU, B)
      WRITE (NOUT, 99999) X, AMU, B, PR
99999 FORMAT (' EXVDF(', F6.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXVDF(  1.00, 0.00, 1.00) = 0.9340
```

EXVIN

This function evaluates the inverse of the extreme value cumulative distribution function.

Function Return Value

EXVIN — Function value, the value of the inverse of the extreme value cumulative distribution function. (Output)

Required Arguments

P — Probability for which the inverse of the extreme value distribution function is to be evaluated. (Input)

AMU — Location parameter of the extreme value probability function. (Input)

BETA — Scale parameter of the extreme value probability function. (Input)

FORTRAN 90 Interface

Generic: `EXVIN` (`P`, `AMU`, `BETA`)
Specific: The specific interface names are `S_EXVIN` and `D_EXVIN`.

FORTRAN 77 Interface

Single: `EXVIN` (`P`, `AMU`, `BETA`)
Double: The double precision name is `DEXVIN`.

Description

The function `EXVIN` evaluates the inverse distribution function of an extreme value random variable with location parameter `AMU` and scale parameter `BETA`.

Example

In this example, we evaluate the inverse probability function at `P` = 0.934, `AMU` = 1.0, `BETA` = 1.0

```
      USE UMACH_INT
      USE EXVIN_INT
```

```
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, B, PR
      CALL UMACH(2, NOUT)
      PR = .934
      AMU = 0.0
      B = 1.0
      X = EXVIN(PR, AMU, B)
      WRITE (NOUT, 99999) PR, AMU, B, X
99999 FORMAT (' EXVIN(', F6.3, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXVIN( 0.934, 0.00, 1.00) = 0.9999
```

EXVPR

This function evaluates the extreme value probability density function.

Function Return Value

EXVPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the extreme value probability density function is to be evaluated. (Input)

AMU — Location parameter of the extreme value probability density function. (Input)

BETA — Scale parameter of the extreme value probability density function. (Input)

FORTRAN 90 Interface

Generic: EXVPR (X, AMU, BETA)

Specific: The specific interface names are S_EXVPR and D_EXVPR.

FORTRAN 77 Interface

Single: EXVPR (X, AMU, BETA)

Double: The double precision name is DEXVPR.

Description

The function EXVPR evaluates the extreme value probability density function, defined as

$$f\left(x \mid \mu,\beta\right) = \beta^{-1} e^{\frac{x-\mu}{\beta}} e^{-e^{\frac{x-\mu}{\beta}}}, \quad -\infty < x, \quad \mu < +\infty, \quad \beta > 0$$

The extreme value distribution is also known as the Gumbel minimum distribution.

Example

In this example, we evaluate the extreme value probability density function at **X** = 2.0, **AMU** = 0.0, **BETA** = 1.0.

```
      USE UMACH_INT
      USE EXVPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, AMU, B, PR
      CALL UMACH(2, NOUT)
      X = -2.0
      AMU = 0.0
      B = 1.0
      PR = EXVPR(X, AMU, B)
      WRITE (NOUT, 99999) X, AMU, B, PR
99999 FORMAT (' EXVPR(', F6.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 EXVPR( -2.00, 0.00, 1.00) = 0.1182
```

FDF

This function evaluates the *F* cumulative distribution function.

Function Return Value

FDF — Function value, the probability that an *F* random variable takes a value less than or equal to the input *F*. (Output)

Required Arguments

F — Argument for which the *F* cumulative distribution function is to be evaluated. (Input)

DFN — Numerator degrees of freedom. (Input)
`DFN` must be positive.

DFD — Denominator degrees of freedom. (Input)
`DFD` must be positive.

Optional Arguments

COMPLEMENT — Logical. If `.TRUE.`, the complement of the *F* cumulative distribution function is evaluated. If `.FALSE.`, the *F* cumulative distribution function is evaluated. (Input)
See the Description section for further details on the use of `COMPLEMENT`.
Default: `COMPLEMENT` = .`FALSE`..

FORTRAN 90 Interface

Generic: `FDF` (`F`, `DFN`, `DFD` `[`, ...`]`)
Specific: The specific interface names are `S_FDF` and `D_FDF`.

FORTRAN 77 Interface

Single: `FDF` (`F`, `DFN`, `DFD`)
Double: The double precision name is `DFDF`.

Description

Function **FDF** evaluates the distribution function of a Snedecor's *F* random variable with **DFN** numerator degrees of freedom and **DFD** denominator degrees of freedom. The function is evaluated by making a transformation to a beta random variable and then using the routine BETDF. If *X* is an *F* variate with ν_1 and ν_2 degrees of freedom and $Y = \nu_1 X/(\nu_2 + \nu_1 X)$, then *Y* is a beta variate with parameters $p = \nu_1/2$ and $q = \nu_2/2$. The function **FDF** also uses a relationship between *F* random variables that can be expressed as follows.

```
FDF(X, DFN, DFD) = 1.0 - FDF(1.0/X, DFD, DFN)
```

If **COMPLEMENT** = **.TRUE.**, the value of **FDF** at the point *x* is 1 **-** *p*, where 1 **-** *p* is the probability that the random variable takes a value greater than *x*. In those situations where the desired end result is 1 **-** *p*, the user can achieve greater accuracy in the right tail region by using the result returned by **FDF** with the optional argument **COMPLEMENT** set to **.TRUE.** rather than by using 1 **-** *p* where *p* is the result returned by **FDF** with **COMPLEMENT** set to **.FALSE.**.

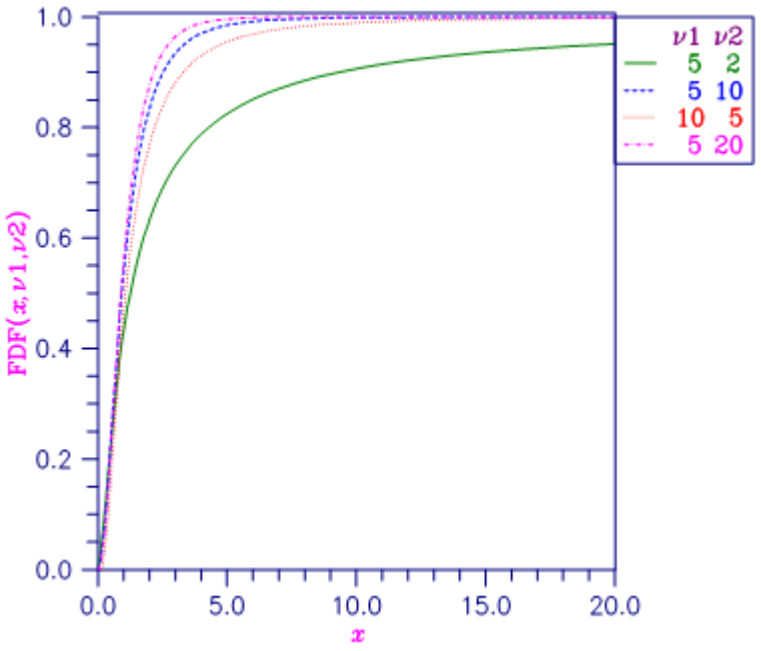


Figure 31, F Distribution Function

Comments

Informational Error

Type	Code	Description
1	3	Since the input argument F is not positive, the distribution function is zero at F.

Example

In this example, we find the probability that an *F* random variable with one numerator and one denominator degree of freedom is greater than 648.

```
      USE UMACH_INT
      USE FDF_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DFD, DFN, F, P
!
      CALL UMACH (2, NOUT)
      F   = 648.0
      DFN = 1.0
      DFD = 1.0
      P   = FDF(F,DFN,DFD, COMPLEMENT=.TRUE.)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that an F(1,1) variate is greater ', &
             'than 648 is ', F6.4)
      END
```

Output

```
 The probability that an F(1, 1) variate is greater than 648 is 0.0250
```

FIN

This function evaluates the inverse of the *F* cumulative distribution function.

Function Return Value

FIN — Function value. (Output)
The probability that an *F* random variable takes a value less than or equal to **FIN** is **P**.

Required Arguments

P — Probability for which the inverse of the *F* distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

DFN — Numerator degrees of freedom. (Input)
DFN must be positive.

DFD — Denominator degrees of freedom. (Input)
DFD must be positive.

FORTRAN 90 Interface

Generic: **FIN** (**P**, **DFN**, **DFD**)
Specific: The specific interface names are **S_FIN** and **D_FIN**.

FORTRAN 77 Interface

Single: **FIN** (**P**, **DFN**, **DFD**)
Double: The double precision name is **DFIN**.

Description

Function **FIN** evaluates the inverse distribution function of a Snedecor's *F* random variable with **DFN** numerator degrees of freedom and **DFD** denominator degrees of freedom. The function is evaluated by making a transformation to a beta random variable and then using the routine **BETIN**. If *X* is an *F* variate with ν_1 and ν_2 degrees of

freedom and $Y = \nu_1 X/(\nu_2 + \nu_1 X)$, then Y is a beta variate with parameters $p = \nu_1/2$ and $q = \nu_2/2$. If $P \leq 0.5$, **FIN** uses this relationship directly, otherwise, it also uses a relationship between F random variables that can be expressed as follows, using routine FDF, which is the F cumulative distribution function:

```
FDF (F, DFN, DFD) = 1.0 - FDF(1.0/F, DFD, DFN).
```

Comments

Informational Error

Type	Code	Description
4	4	FIN is set to machine infinity since overflow would occur upon modifying the inverse value for the F distribution with the result obtained from the inverse beta distribution.

Example

In this example, we find the 99-th percentage point for an F random variable with 1 and 7 degrees of freedom.

```
      USE UMACH_INT
      USE FIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DFD, DFN, F, P
!
      CALL UMACH (2, NOUT)
      P   = 0.99
      DFN = 1.0
      DFD = 7.0
      F   = FIN(P,DFN,DFD)
      WRITE (NOUT,99999) F
99999 FORMAT (' The F(1,7) 0.01 critical value is ', F6.3)
      END
```

Output

```
 The F(1, 7) 0.01 critical value is 12.246
```

FPR

This function evaluates the *F* probability density function.

Function Return Value

FPR — Function value, the value of the probability density function. (Output)

Required Arguments

F — Argument for which the *F* probability density function is to be evaluated. (Input)

DFN — Numerator degrees of freedom. (Input)
DFN must be positive.

DFD — Denominator degrees of freedom. (Input)
DFD must be positive.

FORTRAN 90 Interface

Generic: **FPR** (**F**, **DFN**, **DFD**)
Specific: The specific interface names are **S_FPR** and **D_FPR**

FORTRAN 77 Interface

Single: **FPR** (**F**, **DFN**, **DFD**)
Double: The double precision name is **DFPR**.

Description

The function **FPR** evaluates the F probability density function, defined as

$$f\left(x|v_1,v_2\right) = n\left(v_1,v_2\right)x^{\frac{v_1-2}{2}}\left(1+\frac{v_1 x}{v_2}\right)^{-\frac{\left(v_1+v_2\right)}{2}},$$

$$n\left(v_1,v_2\right) = \frac{\Gamma\left(\frac{v_1+v_2}{2}\right)}{\Gamma\left(\frac{v_1}{2}\right)\Gamma\left(\frac{v_2}{2}\right)}\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}}, \quad x>0, \quad v_i>0, \quad i=1,2$$

The parameters v_1 and v_2, correspond to the arguments **DFN** and **DFD**.

Example

In this example, we evaluate the probability function at **F** = 2.0, **DFN** = 10.0, **DFD** = 1.0.

```
      USE UMACH_INT
      USE FPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL F, DFN, DFD, PR
      CALL UMACH(2, NOUT)
      F = 2.0
      DFN = 10.0
      DFD = 1.0
      PR = FPR(F, DFN, DFD)
      WRITE (NOUT, 99999) F, DFN, DFD, PR
99999 FORMAT (' FPR(', F6.2, ', ', F6.2, ', ', F6.2, ') = ', F6.4)
      END
```

Output

```
 FPR(  2.00,  10.00,   1.00) = 0.1052
```

FNDF

This function evaluates the noncentral *F* cumulative distribution function (CDF).

Function Return Value

FNDF — Probability that a random variable from an *F* distribution having noncentrality parameter LAMBDA takes a value less than or equal to the input F. (Output)

Required Arguments

F — Argument for which the noncentral *F* cumulative distribution function is to be evaluated. (Input)
F must be non-negative.

DF1 — Number of numerator degrees of freedom of the noncentral *F* distribution. (Input)
DF1 must be positive.

DF2 — Number of denominator degrees of freedom of the noncentral *F* distribution. (Input)
DF2 must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: FNDF (F, DF1, DF2, LAMBDA)
Specific: The specific interface names are S_FNDF and D_FNDF.

Description

If *X* is a noncentral chi-square random variable with noncentrality parameter λ and ν_1 degrees of freedom, and *Y* is a chi-square random variable with ν_2 degrees of freedom which is statistically independent of *X*, then

$$F = \left(X / v_1\right) / \left(Y / v_2\right)$$

is a noncentral *F*-distributed random variable whose CDF is given by

$$CDF\left(f,v_1,v_2,\lambda\right) = \sum_{j=0}^{\infty} c_j$$

where

$$c_j = \omega_j I_x\left(\frac{v_1}{2} + j, \frac{v_2}{2}\right)$$

$$\omega_j = e^{-\lambda/2}\left(\lambda/2\right)^j / j! = \frac{\lambda}{2j}\omega_{j-1}$$

$$I_x\left(a,b\right) = B_x\left(a,b\right) / B\left(a,b\right)$$

$$B_x\left(a,b\right) = \int_0^x t^{a-1}\left(1-t\right)^{b-1}dt = x^a\sum_{j=0}^{\infty}\frac{\Gamma\left(j+1-b\right)}{\left(a+j\right)\Gamma\left(1-b\right)j!}x^j$$

$$x = v_1 f / \left(v_2 + v_1 f\right)$$

$$B\left(a,b\right) = B_1\left(a,b\right) = \frac{\Gamma\left(a\right)\Gamma\left(b\right)}{\Gamma\left(a+b\right)}$$

$$I_x\left(a+1,b\right) = I_x\left(a,b\right) - T_x\left(a,b\right)$$

$$T_x\left(a,b\right) = \frac{\Gamma\left(a+b\right)}{\Gamma\left(a+1\right)\Gamma\left(b\right)}x^a\left(1-x\right)^b = T_x\left(a-1,b\right)\frac{a-1+b}{a}x$$

and $\Gamma(\cdot)$ is the gamma function. The above series expansion for the noncentral *F* CDF was taken from Butler and Paolella (1999) (see Paolella.pdf), with the correction for the recursion relation given below:

$$I_x\left(a+1,b\right) = I_x\left(a,b\right) - T_x\left(a,b\right)$$

extracted from the AS 63 algorithm for calculating the incomplete beta function as described by Majumder and Bhattacharjee (1973).

The correspondence between the arguments of function **FNDF** (**F**, **DF1**, **DF2**, **LAMBDA**) and the variables in the above equations is as follows: ν_1 = **DF1**, ν_2 = **DF2**, λ = **LAMBDA**, and f = **F**.

For λ = 0, the noncentral *F* distribution is the same as the *F* distribution.

Example

This example traces out a portion of a noncentral *F* distribution with parameters **DF1** = 100, **DF2** = 10, and **LAMBDA** = 10.

```
          USE UMACH_INT
          USE FNDF_INT
          IMPLICIT NONE
          INTEGER NOUT, I
          REAL X, LAMBDA, DF1, DF2, CDFV, X0(8)
          DATA X0 / 0.0, .4, .8, 1.2, 1.6, 2.0, 2.8, 4.0 /

          CALL UMACH (2, NOUT)
          DF1 = 100.0
          DF2 = 10.0
          LAMBDA = 10.0
          WRITE (NOUT,'("DF1: ", F4.0, ";  DF2: ", F4.0, &
             ";  LAMBDA: ", F4.0 // "   X        CDF(X)")')&
             DF1, DF2, LAMBDA
          DO I = 1, 8
             X = X0(I)
             CDFV = FNDF(X, DF1, DF2, LAMBDA)
             WRITE (NOUT,'(1X, F5.1, 2X, E12.6)') X, CDFV
          END DO
          END
```

Output

```
    DF1: 100.;  DF2:  10.;  LAMBDA:  10.

       X        CDF(X)
       0.0  0.000000E+00
       0.4  0.488790E-02
       0.8  0.202633E+00
       1.2  0.521143E+00
       1.6  0.733853E+00
       2.0  0.850413E+00
       2.8  0.947125E+00
       4.0  0.985358E+00
```

FNIN

This function evaluates the inverse of the noncentral *F* cumulative distribution function (CDF).

Function Return Value

FNIN — Function value, the value of the inverse of the cumulative distribution function evaluated at **P**. The probability that a noncentral *F* random variable takes a value less than or equal to **FNIN** is **P**. (Output)

Required Arguments

P — Probability for which the inverse of the noncentral *F* cumulative distribution function is to be evaluated. (Input)
P must be non-negative and less than 1.

DF1 — Number of numerator degrees of freedom of the noncentral *F* distribution. (Input)
DF1 must be positive.

DF2 — Number of denominator degrees of freedom of the noncentral *F* distribution. (Input)
DF2 must be positive.

LAMBDA — Noncentrality parameter. (Input)
LAMBDA must be non-negative.

FORTRAN 90 Interface

Generic: **FNIN** (**P**, **DF1**, **DF2**, **LAMBDA**)
Specific: The specific interface names are **S_FNIN** and **D_FNIN**.

Description

If *X* is a noncentral chi-square random variable with noncentrality parameter λ and ν_1 degrees of freedom, and *Y* is a chi-square random variable with ν_2 degrees of freedom which is statistically independent of *X*, then

$$F = \left(X / v_1 \right) / \left(Y / v_2 \right)$$

is a noncentral *F*-distributed random variable whose CDF is given by

$$p = CDF\left(f, v_1, v_2, \lambda\right) = \sum_{j=0}^{\infty} c_j$$

where:

$$c_j = \omega_j I_x\left(\frac{v_1}{2} + j, \frac{v_2}{2}\right)$$

$$\omega_j = e^{-\lambda/2}\left(\lambda/2\right)^j / j! = \frac{\lambda}{2j}\omega_{j-1}$$

$$I_x\left(a,b\right) = B_x\left(a,b\right) / B\left(a,b\right)$$

$$B_x\left(a,b\right) = \int_0^x t^{a-1}\left(1-t\right)^{b-1}dt = x^a\sum_{j=0}^{\infty}\frac{\Gamma\left(j+1-b\right)}{\left(a+j\right)\Gamma\left(1-b\right)j!}x^j$$

$$x = v_1 f / \left(v_2 + v_1 f\right)$$

$$B\left(a,b\right) = B_1\left(a,b\right) = \frac{\Gamma\left(a\right)\Gamma\left(b\right)}{\Gamma\left(a+b\right)}$$

$$I_x\left(a+1,b\right) = I_x\left(a,b\right) - T_x\left(a,b\right)$$

$$T_x\left(a,b\right) = \frac{\Gamma\left(a+b\right)}{\Gamma\left(a+1\right)\Gamma\left(b\right)}x^a\left(1-x\right)^b = T_x\left(a-1,b\right)\frac{a-1+b}{a}x$$

and $\Gamma(\cdot)$ is the gamma function, and $p = CDF(f)$ is the probability that $F \le f$. The correspondence between the arguments of function **FNIN**(**P**,**DF1**,**DF2**,**LAMBDA**) and the variables in the above equations is as follows: ν_1 = **DF1**, ν_2 = **DF2**, λ = **LAMBDA**, and p = **P**.

Function **FNIN** evaluates

$$f = CDF^{-1}\left(p, v_1, v_1, \lambda\right)$$

Function **FNIN** uses bisection and modified regula falsi search algorithms to invert the distribution function *CDF*(*f*), which is evaluated using function **FNDF**. For sufficiently small *p*, an accurate approximation of $CDF^{-1}(p)$ can be used which requires no such inverse search algorithms.

Example

This example traces out a portion of an inverse noncentral *F* distribution with parameters **DF1** = 100, **DF2** = 10, and **LAMBDA** = 10.

```
          USE UMACH_INT
          USE FNDF_INT
          USE FNIN_INT
          IMPLICIT NONE
          INTEGER NOUT, I
          REAL F, LAMBDA, DF1, DF2, CDF, CDFINV,F0(8)
          DATA F0 / 0.0, .4, .8, 1.2, 1.6, 2.0, 2.8, 4.0 /

          CALL UMACH (2, NOUT)
          DF1 = 100.0
          DF2 = 10.0
          LAMBDA = 10.0
          WRITE (NOUT,'("DF1: ", F4.0, ";  DF2: ", F4.0, &
             ";  LAMBDA: ", F4.0 // "   F     P = CDF(F)     CDFINV(P)")')&
             DF1, DF2, LAMBDA
          DO I = 1, 8
             F = F0(I)
             CDF = FNDF(F, DF1, DF2, LAMBDA)
             CDFINV = FNIN(CDF, DF1, DF2, LAMBDA)
             WRITE (NOUT,'(1X, F5.1, 2(2X, E12.6))') F, CDF, CDFINV
          END DO
          END
```

Output

```
    DF1: 100.;  DF2:  10.;  LAMBDA:  10.

       F     P = CDF(F)     CDFINV(P)
       0.0  0.000000E+00  0.000000E+00
       0.4  0.488790E-02  0.400000E+00
       0.8  0.202633E+00  0.800000E+00
       1.2  0.521143E+00  0.120000E+01
       1.6  0.733853E+00  0.160000E+01
       2.0  0.850413E+00  0.200000E+01
       2.8  0.947125E+00  0.280000E+01
       4.0  0.985358E+00  0.400000E+01
```

FNPR

This function evaluates the noncentral *F* probability density function.

Function Return Value

FNPR — Function value, the value of the probability density function. (Output)

Required Arguments

F — Argument for which the noncentral *F* probability density function is to be evaluated. (Input)
`F` must be non-negative.

DF1 — Number of numerator degrees of freedom of the noncentral *F* distribution. (Input)
`DF1` must be positive.

DF2 — Number of denominator degrees of freedom of the noncentral *F* distribution. (Input)
`DF2` must be positive.

LAMBDA — Noncentrality parameter. (Input)
`LAMBDA` must be non-negative.

FORTRAN 90 Interface

Generic: `FNPR` (`F`, `DF1`, `DF2`, `LAMBDA`)
Specific: The specific interface names are `S_FNPR` and `D_FNPR`.

Description

If *X* is a noncentral chi-square random variable with noncentrality parameter λ and ν_1 degrees of freedom, and *Y* is a chi-square random variable with ν_2 degrees of freedom which is statistically independent of *X*, then

$$F = \left(X / \nu_1\right) / \left(Y / \nu_2\right)$$

is a noncentral *F*-distributed random variable whose PDF is given by

$$PDF\left(f,\nu_1,\nu_2,\lambda\right) = \Psi\sum_{k=0}^{\infty}\Phi_k$$

where

$$\Psi = \frac{e^{-\lambda/2}\left(\nu_1 f\right)^{\nu_1/2}\left(\nu_2\right)^{\nu_2/2}}{f\left(\nu_1 f+\nu_2\right)^{(\nu_1+\nu_2)/2}\Gamma\left(\nu_2/2\right)}$$

$$\Phi_k = \frac{R^k\Gamma\left(\frac{\nu_1+\nu_2}{2}+k\right)}{k!\Gamma\left(\frac{\nu_1}{2}+k\right)}$$

$$R = \frac{\lambda\nu_1 f}{2\left(\nu_1 f+\nu_2\right)}$$

and $\Gamma(\cdot)$ is the gamma function, ν_1 = **DF1**, ν_2 = **DF2**, λ = **LAMBDA**, and f = **F**.

With a noncentrality parameter of zero, the noncentral *F* distribution is the same as the *F* distribution.

The efficiency of the calculation of the above series is enhanced by:

- calculating each term Φ_k in the series recursively in terms of either the term Φ_{k-1} preceding it or the term Φ_{k+1} following it, and
- initializing the sum with the largest series term and adding the subsequent terms in order of decreasing magnitude.

Special cases:

For $R = \lambda f = 0$

$$PDF\left(f,\nu_1,\nu_2,\lambda\right) = \Psi\Phi_0 = \Psi\frac{\Gamma(\left[\nu_1+\nu_2\right]/2)}{\Gamma\left(\nu_1/2\right)}$$

For $\lambda = 0$

$$PDF\left(f,\nu_1,\nu_2,\lambda\right) = \frac{\left(\nu_1 f\right)^{\nu_1/2}\left(\nu_2\right)^{\nu_2/2}\Gamma(\left[\nu_1+\nu_2\right]/2)}{f\left(\nu_1 f+\nu_2\right)^{(\nu_1+\nu_2)/2}\Gamma\left(\nu_1/2\right)\Gamma\left(\nu_2/2\right)}$$

For $f = 0$

$$PDF(f,\nu_1,\nu_2,\lambda) = \frac{e^{-\lambda/2} f^{\nu_1/2 - 1} (\nu_1/\nu_2)^{\nu_1/2} \Gamma([\nu_1 + \nu_2]/2)}{\Gamma(\nu_1/2)\,\Gamma(\nu_2/2)} = \begin{cases} 0 & \text{if } \nu_1 > 2; \\ e^{-\lambda/2} & \text{if } \nu_1 = 2; \\ \infty & \text{if } \nu_1 < 2; \end{cases}$$

Example

This example traces out a portion of a noncentral *F* distribution with parameters **DF1** = 100, **DF2** = 10, and **LAMBDA** = 10.

```
          USE UMACH_INT
          USE FNPR_INT
          IMPLICIT NONE

          INTEGER NOUT, I
          REAL F, LAMBDA, DF1, DF2, PDFV, X0(8)
          DATA X0 /0.0, 0.4, 0.8, 3.2, 5.6,8.8, 14.0, 18.0/

          CALL UMACH (2, NOUT)
          DF1 = 100.0
          DF2 = 10.0
          LAMBDA = 10.0

          WRITE (NOUT,'("DF1: ", F4.0, ";  DF2: ", F4.0, ";  LAMBDA'// &
             ': ", F4.0 //"   F        PDF(F)")') DF1, DF2, LAMBDA

          DO I = 1, 8
             F = X0(I)
             PDFV = FNPR(F, DF1, DF2, LAMBDA)
             WRITE (NOUT,'(1X, F5.1, 2X, E12.6)') F, PDFV
          END DO
          END
```

Output

```
    DF1: 100.;  DF2:  10.;  LAMBDA:  10.

       F        PDF(F)
       0.0  0.000000E+00
       0.4  0.974879E-01
       0.8  0.813115E+00
       3.2  0.369482E-01
       5.6  0.283023E-02
       8.8  0.276607E-03
      14.0  0.219632E-04
      18.0  0.534831E-05
```

GAMDF

This function evaluates the gamma cumulative distribution function.

Function Return Value

GAMDF — Function value, the probability that a gamma random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the gamma distribution function is to be evaluated. (Input)

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

FORTRAN 90 Interface

Generic: GAMDF (X, A)
Specific: The specific interface names are S_GAMDF and D_GAMDF.

FORTRAN 77 Interface

Single: GAMDF (X, A)
Double: The double precision name is DGAMDF.

Description

Function GAMDF evaluates the distribution function, *F*, of a gamma random variable with shape parameter *a*; that is,

$$F(x) = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt$$

where Γ(·) is the gamma function. (The gamma function is the integral from 0 to ∞ of the same integrand as above). The value of the distribution function at the point *x* is the probability that the random variable takes a value less than or equal to *x*.

The gamma distribution is often defined as a two-parameter distribution with a scale parameter *b* (which must be positive), or even as a three-parameter distribution in which the third parameter *c* is a location parameter. In the most general case, the probability density function over (*c*, ∞) is

$$f(t) = \frac{1}{b^a \Gamma(a)} e^{-(t-c)/b} (t-c)^{a-1}$$

If *T* is such a random variable with parameters *a*, *b*, and *c*, the probability that $T \le t_0$ can be obtained from **GAMDF** by setting **X** = (t_0 - *c*)/*b*.

If **X** is less than *a* or if **X** is less than or equal to 1.0, **GAMDF** uses a series expansion. Otherwise, a continued fraction expansion is used. (See Abramowitz and Stegun, 1964.)

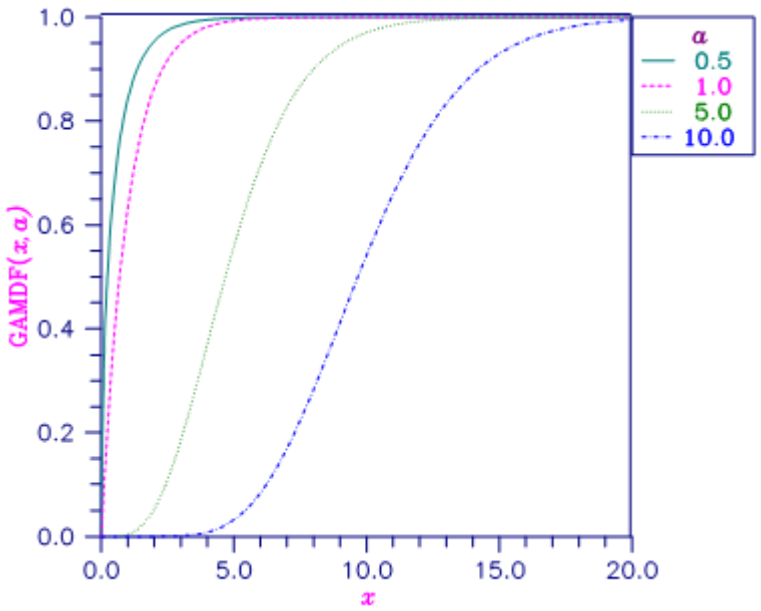


Figure 32, Gamma Distribution Function

Comments

Informational Error

Type	Code	Description
1	2	Since the input argument X is less than zero, the distribution function is set to zero.

Example

Suppose X is a gamma random variable with a shape parameter of 4. (In this case, it has an *Erlang distribution* since the shape parameter is an integer.) In this example, we find the probability that X is less than 0.5 and the probability that X is between 0.5 and 1.0.

```
      USE UMACH_INT
      USE GAMDF_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       A, P, X
!
      CALL UMACH (2, NOUT)
      A = 4.0
      X = 0.5
      P = GAMDF(X,A)
      WRITE (NOUT,99998) P
99998 FORMAT (' The probability that X is less than 0.5 is ', F6.4)
      X = 1.0
      P = GAMDF(X,A) - P
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is between 0.5 and 1.0 is ', &
            F6.4)
      END
```

Output

```
 The probability that X is less than 0.5 is 0.0018
 The probability that X is between 0.5 and 1.0 is 0.0172
```

GAMIN

This function evaluates the inverse of the gamma cumulative distribution function.

Function Return Value

GAMIN — Function value. (Output)
The probability that a gamma random variable takes a value less than or equal to **GAMIN** is **P**.

Required Arguments

P — Probability for which the inverse of the gamma cumulative distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

FORTRAN 90 Interface

Generic: **GAMIN** (**P**, **A**)
Specific: The specific interface names are **S_GAMIN** and **D_GAMIN**.

FORTRAN 77 Interface

Single: **GAMIN** (**P**, **A**)
Double: The double precision name is **DGAMIN**.

Description

Function **GAMIN** evaluates the inverse distribution function of a gamma random variable with shape parameter *a*, that is, it determines *x* (= **GAMIN(P, A)**), such that

$$P = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt$$

where **Γ**(·) is the gamma function. The probability that the random variable takes a value less than or equal to *x* is *P*. See the documentation for routine GAMDF for further discussion of the gamma distribution.

Function **GAMIN** uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine **GAMDF**.

Comments

Informational Error

Type	Code	Description
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Example

In this example, we find the 95-th percentage point for a gamma random variable with shape parameter of 4.

```
      USE UMACH_INT
      USE GAMIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       A, P, X
!
      CALL UMACH (2, NOUT)
      A = 4.0
      P = 0.95
      X = GAMIN(P,A)
      WRITE (NOUT,99999) X
!
99999 FORMAT (' The 0.05 gamma(4) critical value is ', F6.3, &
             '.')
!
      END
```

Output

```
 The 0.05 gamma(4) critical value is 7.754.
```

GAMPR

This function evaluates the gamma probability density function.

Function Return Value

GAMPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the gamma probability density function is to be evaluated. (Input)

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

FORTRAN 90 Interface

Generic: **GAMPR** (**X**, **A**)
Specific: The specific interface names are **S_GAMPR** and **D_GAMPR**.

FORTRAN 77 Interface

Single: **GAMPR** (**X**, **A**)
Double: The double precision name is **DGAMPR**.

Description

The function **GAMPR** evaluates the gamma probability density function, defined as

$$\Gamma(x|a) = \frac{1}{\Gamma(a)}(x)^{a-1}e^{-x}, \quad x, a > 0$$

Example

In this example, we evaluate the probability function at **X** = 4.0, **A** = 5.0.

```
      USE UMACH_INT
      USE GAMPR_INT
```

```
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, PR
      CALL UMACH(2, NOUT)
      X = 4.0
      A = 5.0
      PR = GAMPR(X, A)
      WRITE (NOUT, 99999) X, A, PR
99999 FORMAT (' GAMPR(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 GAMPR(4.00, 5.00) = 0.1954
```

RALDF

This function evaluates the Rayleigh cumulative distribution function.

Function Return Value

RALDF — Function value, the probability that a Rayleigh random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the Rayleigh cumulative distribution function is to be evaluated. (Input)

ALPHA — Scale parameter of the Rayleigh cumulative distribution function. (Input)

FORTRAN 90 Interface

Generic: RALDF (X, ALPHA)
Specific: The specific interface names are S_RALDF and D_RALDF.

FORTRAN 77 Interface

Single: RALDF (X, ALPHA)
Double: The double precision name is DRALDF.

Description

The function RALDF evaluates the Rayleigh cumulative probability distribution function, which is a special case of the Weibull cumulative probability distribution function, where the shape parameter GAMMA is 2.0

$$F(x) = 1 - e^{\frac{x^2}{2\alpha^2}}$$

RALDF evaluates the Rayleigh cumulative probability distribution function using the relationsip

RALDF (X, ALPHA) = `WBLDF(X, SQRT(2.0)*ALPHA, 2.0)`.

Example

In this example, we evaluate the Rayleigh cumulative distribution function at **X** = 0.25, **ALPHA** = 0.5.

```
      USE UMACH_INT
      USE RALDF_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, ALPHA, PR
      CALL UMACH(2, NOUT)
      X = 0.25
      ALPHA = 0.5
      PR = RALDF(X, ALPHA)
      WRITE (NOUT, 99999) X, ALPHA, PR
99999 FORMAT (' RALDF(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 RALDF(0.25, 0.50) = 0.1175
```

RALIN

This function evaluates the inverse of the Rayleigh cumulative distribution function.

Function Return Value

RALIN — Function value, the value of the inverse of the cumulative distribution function. (Output)

Required Arguments

P — Probability for which the inverse of the Rayleigh distribution function is to be evaluated. (Input)

ALPHA — Scale parameter of the Rayleigh cumulative distribution function. (Input)

FORTRAN 90 Interface

Generic: RALIN (P, ALPHA)
Specific: The specific interface names are S_RALIN and D_RALIN.

FORTRAN 77 Interface

Single: RALIN (P, ALPHA)
Double: The double precision name is DRALIN.

Description

The function RALIN evaluates the inverse distribution function of a Rayleigh random variable with scale parameter ALPHA.

Example

In this example, we evaluate the inverse probability function at P = 0.1175, ALPHA= 0.5.

```
      USE UMACH_INT
      USE RALIN_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, ALPHA, P
```

```
      CALL UMACH(2, NOUT)
      P = 0.1175
      ALPHA = 0.5
      X = RALIN(P, ALPHA)
      WRITE (NOUT, 99999) P, ALPHA, X
99999 FORMAT (' RALIN(', F6.4, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 RALIN(0.1175, 0.50) = 0.2500
```

RALPR

This function evaluates the Rayleigh probability density function.

Function Return Value

RALPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the Rayleigh probability density function is to be evaluated. (Input)

ALPHA — Scale parameter of the Rayleigh probability function. (Input)

FORTRAN 90 Interface

Generic: **RALPR** (**X**, **ALPHA**)
Specific: The specific interface names are **S_RALPR** and **D_RALPR**.

FORTRAN 77 Interface

Single: **RALPR** (**X**, **ALPHA**)
Double: The double precision name is **DRALPR**.

Description

The function **RALPR** evaluates the Rayleigh probability density function, which is a special case of the Weibull probability density function where **GAMMA** is equal to 2.0, and is defined as

$$f\left(x|\alpha\right) = \frac{x}{\alpha^2} e^{-\frac{x^2}{2\alpha^2}}, \; x > 0$$

Example

In this example, we evaluate the Rayleigh probability density function at **X** = 0.25, **ALPHA** = 0.5.

```
      USE UMACH_INT
```

```
      USE RALPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, ALPHA, PR
      CALL UMACH(2, NOUT)
      X = 0.25
      ALPHA = 0.5
      PR = RALPR(X, ALPHA)
      WRITE (NOUT, 99999) X, ALPHA, PR
99999 FORMAT (' RALPR(', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 RALPR(0.25, 0.50) = 0.8825
```

TDF

This function evaluates the Student's *t* cumulative distribution function.

Function Return Value

TDF — Function value, the probability that a Student's *t* random variable takes a value less than or equal to the input **T**. (Output)

Required Arguments

T — Argument for which the Student's *t* distribution function is to be evaluated. (Input)

DF — Degrees of freedom. (Input)
DF must be greater than or equal to 1.0.

Optional Arguments

COMPLEMENT — Logical. If **.TRUE.**, the complement of the Student's *t* cumulative distribution function is evaluated. If **.FALSE.**, the Student's *t* cumulative distribution function is evaluated. (Input)
See the Description section for further details on the use of **COMPLEMENT**.
Default: **COMPLEMENT** = .**FALSE.**.

FORTRAN 90 Interface

Generic: **TDF** (**T**, **DF** [, ...])
Specific: The specific interface names are **S_TDF** and **D_TDF**.

FORTRAN 77 Interface

Single: **TDF** (**T**, **DF**)
Double: The double precision name is **DTDF**.

Description

Function **TDF** evaluates the cumulative distribution function of a Student's *t* random variable with **DF** degrees of freedom. If the square of **T** is greater than or equal to **DF**, the relationship of a *t* to an *F* random variable (and subsequently, to a beta random variable) is exploited, and routine **BETDF** is used. Otherwise, the method described by Hill (1970) is used. Let ν = **DF**. If ν is not an integer, if ν is greater than 19, or if ν is greater than 200, a Cornish-Fisher expansion is used to evaluate the distribution function. If ν is less than 20 and **ABS**(**T**) is less than 2.0, a trigonometric series (see Abramowitz and Stegun 1964, equations 26.7.3 and 26.7.4, with some rearrangement) is used. For the remaining cases, a series given by Hill (1970) that converges well for large values of **T** is used.

If **COMPLEMENT** = **.TRUE.**, the value of **TDF** at the point *x* is 1 **-** *p*, where 1 **-** *p* is the probability that the random variable takes a value greater than *x*. In those situations where the desired end result is 1 **-** *p*, the user can achieve greater accuracy in the right tail region by using the result returned by **TDF** with the optional argument **COMPLEMENT** set to **.TRUE.** rather than by using 1 **-** *p* where *p* is the result returned by **TDF** with **COMPLEMENT** set to **.FALSE.**.

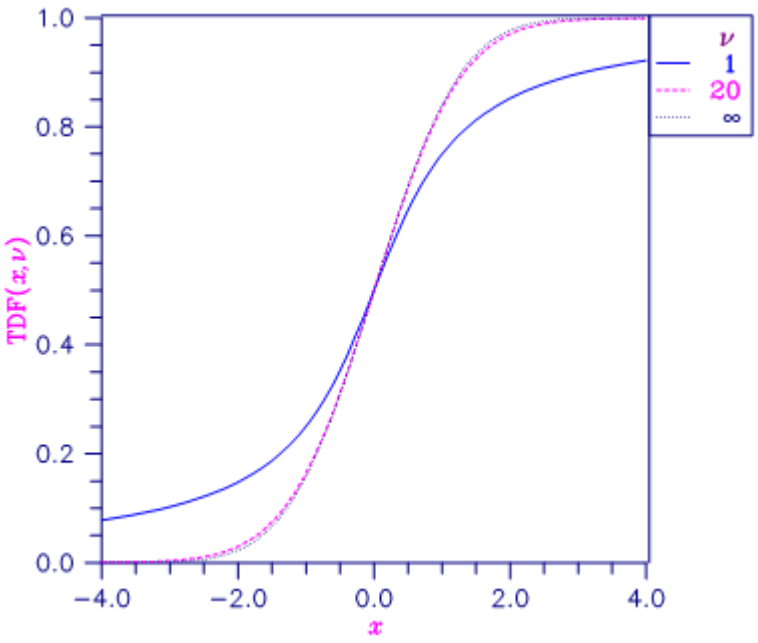


Figure 33, Student's t Distribution Function

Example

In this example, we find the probability that a *t* random variable with 6 degrees of freedom is greater in absolute value than 2.447. We use the fact that *t* is symmetric about 0.

```
      USE TDF_INT
      USE UMACH_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DF, P, T
!
      CALL UMACH (2, NOUT)
      T  = 2.447
      DF = 6.0
      P  = 2.0*TDF(-T,DF)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that a t(6) variate is greater ', &
             'than 2.447 in', /, ' absolute value is ', F6.4)
      END
```

Output

```
 The probability that a t(6) variate is greater than 2.447 in absolute value is 0.0500
```

TIN

This function evaluates the inverse of the Student's *t* cumulative distribution function.

Function Return Value

TIN — Function value. (Output)
The probability that a Student's *t* random variable takes a value less than or equal to **TIN** is **P**.

Required Arguments

P — Probability for which the inverse of the Student's *t* cumulative distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

DF — Degrees of freedom. (Input)
DF must be greater than or equal to 1.0.

FORTRAN 90 Interface

Generic: **TIN** (**P**, **DF**)
Specific: The specific interface names are **S_TIN** and **D_TIN**.

FORTRAN 77 Interface

Single: **TIN** (**P**, **DF**)
Double: The double precision name is **DTIN**.

Description

Function **TIN** evaluates the inverse distribution function of a Student's *t* random variable with **DF** degrees of freedom. Let ν = **DF**. If ν equals 1 or 2, the inverse can be obtained in closed form, if ν is between 1 and 2, the relationship of a *t* to a beta random variable is exploited and routine **BETIN** is used to evaluate the inverse; otherwise the algorithm of Hill (1970) is used. For small values of ν greater than 2, Hill's algorithm inverts an integrated expansion in $1/(1 + t^2/\nu)$ of the *t* density. For larger values, an asymptotic inverse Cornish-Fisher type expansion about normal deviates is used.

Comments

Informational Error

Type	Code	Description
4	3	`TIN` is set to machine infinity since overflow would occur upon modifying the inverse value for the *F* distribution with the result obtained from the inverse β distribution.

Example

In this example, we find the 0.05 critical value for a two-sided *t* test with 6 degrees of freedom.

```
      USE TIN_INT
      USE UMACH_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DF, P, T
!
      CALL UMACH (2, NOUT)
      P  = 0.975
      DF = 6.0
      T  = TIN(P,DF)
      WRITE (NOUT,99999) T
99999 FORMAT (' The two-sided t(6) 0.05 critical value is ', F6.3)
      END
```

Output

```
 The two-sided t(6) 0.05 critical value is  2.447
```

TPR

This function evaluates the Student's *t* probability density function

Function Return Value

TPR — Function value, the value of the probability density function. (Output)

Required Arguments

T — Argument for which the Student's *t* probability density function is to be evaluated. (Input)

DF — Degrees of freedom. (Input)
DF must be greater than or equal to 1.0.

FORTRAN 90 Interface

Generic: **TPR** (**T**, **DF**)
Specific: The specific interface names are **S_TPR** and **D_TPR**

FORTRAN 77 Interface

Single: **TPR** (**T**, **DF**)
Double: The double precision name is **DTPR**.

Description

The function **TPR** evaluates the Student's *t* probability density function, defined as

$$f(t \mid \nu) = \left(\beta(0.5, 0.5\nu)\sqrt{\nu}\right)^{-1}\left(1 + \frac{t^2}{\nu}\right)^{-(\nu+1)/2}, \quad -\infty < t < +\infty, \quad \nu \geq 1$$

Where ν = **DF**.

The normalizing factor uses the Beta function, **BETA** (see the Special Functions book, *Chapter 4, Gamma and Related Functions*).

Example

In this example, we evaluate the probability function at **T** = 1.5, **DF** = 10.0.

```
      USE UMACH_INT
      USE TPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL T, DF, PR
      CALL UMACH(2, NOUT)
      T = 1.5
      DF = 10.0
      PR = TPR(T, DF)
      WRITE (NOUT, 99999) T, DF, PR
99999 FORMAT (' TPR(', F4.2, ', ', F6.2, ') = ', F6.4)
      END
```

Output

```
 TPR(1.50,  10.00) = 0.1274
```

TNDF

This function evaluates the noncentral Student's *t* cumulative distribution function.

Function Return Value

TNDF — Function value, the probability that a noncentral Student's *t* random variable takes a value less than or equal to **T**. (Output)

Required Arguments

T — Argument for which the noncentral Student's *t* cumulative distribution function is to be evaluated. (Input)

IDF — Number of degrees of freedom of the noncentral Student's *t* cumulative distribution. (Input) **IDF** must be positive.

DELTA — The noncentrality parameter. (Input)

FORTRAN 90 Interface

Generic: **TNDF** (**T**, **IDF**, **DELTA**)

Specific: The specific interface names are **S_TNDF** and **D_TNDF**.

FORTRAN 77 Interface

Single: **TNDF** (**T**, **IDF**, **DELTA**)

Double: The double precision name is **DTNDF**.

Description

Function **TNDF** evaluates the cumulative distribution function *F* of a noncentral *t* random variable with **IDF** degrees of freedom and noncentrality parameter **DELTA**; that is, with ν = **IDF**, δ = **DELTA**, and t_0 = **T**,

$$F(t_0) = \int_{-\infty}^{t_0} \frac{v^{v/2} e^{-\delta^2/2}}{\sqrt{\pi}\Gamma(v/2)(v+x^2)^{(v+1)/2}} \sum_{i=0}^{\infty} \Gamma((v+i+1)/2)\left(\frac{\delta^i}{i!}\right)\left(\frac{2x^2}{v+x^2}\right)^{i/2} dx$$

where $\Gamma(\cdot)$ is the gamma function. The value of the distribution function at the point t_0 is the probability that the random variable takes a value less than or equal to t_0.

The noncentral *t* random variable can be defined by the distribution function above, or alternatively and equivalently, as the ratio of a normal random variable and an independent chi-squared random variable. If *w* has a normal distribution with mean δ and variance equal to one, *u* has an independent chi-squared distribution with ν degrees of freedom, and

$$x = w / \sqrt{u/v}$$

then *x* has a noncentral *t* distribution with ν degrees of freedom and noncentrality parameter δ.

The distribution function of the noncentral *t* can also be expressed as a double integral involving a normal density function (see, for example, Owen 1962, page 108). The function **TNDF** uses the method of Owen (1962, 1965), which uses repeated integration by parts on that alternate expression for the distribution function.

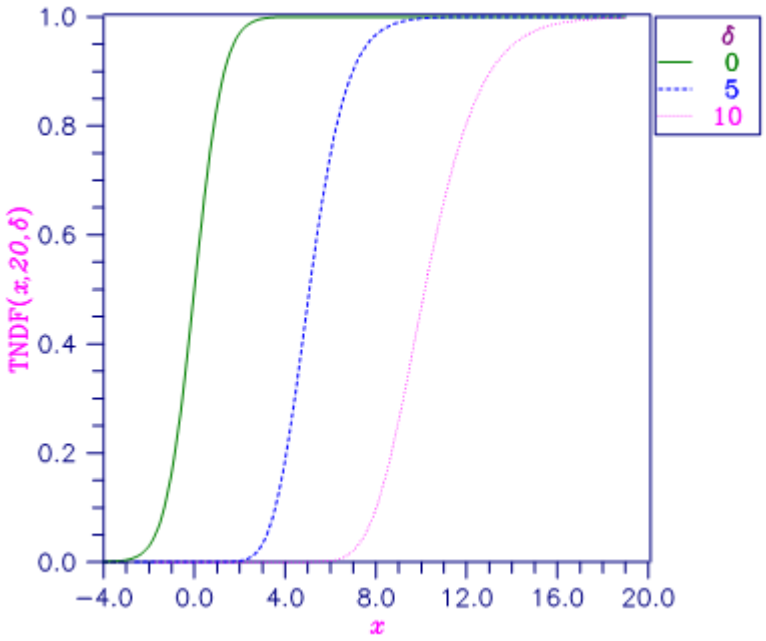


Figure 34, Noncentral Student's t Distribution Function

Comments

Informational error

Type	Code	Description
4	2	An accurate result cannot be computed due to possible underflow for the machine precision available. `DELTA*SQRT(IDF/(IDF+T**2))` must be less than `SQRT(-1.9*ALOG(S))`, where `S=AMACH(1)`.

Example

Suppose *T* is a noncentral *t* random variable with 6 degrees of freedom and noncentrality parameter 6. In this example, we find the probability that *T* is less than 12.0. (This can be checked using the table on page 111 of Owen 1962, with $\eta = 0.866$, which yields $\lambda = 1.664$.)

```
      USE UMACH_INT
      USE TNDF_INT
      IMPLICIT   NONE
      INTEGER    IDF, NOUT
      REAL       DELTA, P, T
!
      CALL UMACH (2, NOUT)
      IDF   = 6
      DELTA = 6.0
      T     = 12.0
      P     = TNDF(T,IDF,DELTA)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that T is less than 12.0 is ', F6.4)
      END
```

Output

```
 The probability that T is less than 12.0 is 0.9501
```

TNIN

This function evaluates the inverse of the noncentral Student's *t* cumulative distribution function.

Function Return Value

TNIN — Function value. (Output)
The probability that a noncentral Student's *t* random variable takes a value less than or equal to `TNIN` is `P`.

Required Arguments

P — Probability for which the inverse of the noncentral Student's *t* cumulative distribution function is to be evaluated. (Input)
`P` must be in the open interval (0.0, 1.0).

IDF — Number of degrees of freedom of the noncentral Student's *t* cumulative distribution. (Input) `IDF` must be positive.

DELTA — The noncentrality parameter. (Input)

FORTRAN 90 Interface

Generic: `TNIN` (`P`, `IDF`, `DELTA`)
Specific: The specific interface names are `S_TNIN` and `D_TNIN`.

FORTRAN 77 Interface

Single: `TNIN` (`P`, `IDF`, `DELTA`)
Double: The double precision name is `DTNIN`.

Description

Function `TNIN` evaluates the inverse distribution function of a noncentral *t* random variable with `IDF` degrees of freedom and noncentrality parameter `DELTA`; that is, with P = `P`, ν = `IDF`, and δ = `DELTA`, it determines t_0 (= `TNIN(P,IDF,DELTA)`), such that

$$P = \int_{-\infty}^{t_0} \frac{v^{v/2} e^{-\delta^2/2}}{\sqrt{\pi}\Gamma(v/2)(v+x^2)^{(v+1)/2}} \sum_{i=0}^{\infty} \Gamma((v+i+1)/2)\left(\frac{\delta^i}{i!}\right)\left(\frac{2x^2}{v+x^2}\right)^{i/2} dx$$

where $\Gamma(\cdot)$ is the gamma function. The probability that the random variable takes a value less than or equal to t_0 is P. See TNDF for an alternative definition in terms of normal and chi-squared random variables. The function TNIN uses bisection and modified regula falsi to invert the distribution function, which is evaluated using routine TNDF.

Comments

Informational Error

Type	Code	Description
4	1	Over 100 iterations have occurred without convergence. Convergence is assumed.

Example

In this example, we find the 95-th percentage point for a noncentral t random variable with 6 degrees of freedom and noncentrality parameter 6.

```
      USE TNIN_INT
      USE UMACH_INT
      IMPLICIT   NONE
      INTEGER    IDF, NOUT
      REAL       DELTA, P, T
!
      CALL UMACH (2, NOUT)
      IDF   = 6
      DELTA = 6.0
      P     = 0.95
      T     = TNIN(P,IDF,DELTA)
      WRITE (NOUT,99999) T
!
99999 FORMAT (' The 0.05 noncentral t critical value is ', F6.3, &
             '.')
!
      END
```

Output

```
 The 0.05 noncentral t critical value is 11.995.
```

TNPR

This function evaluates the noncentral Student's *t* probability density function.

Function Return Value

TNPR — Function value, the value of the probability density function. (Output)

Required Arguments

T — Argument for which the noncentral Student's *t* probability density function is to be evaluated. (Input)

DF — Number of degrees of freedom of the noncentral Student's *t* distribution. (Input)
DF must be positive.

DELTA — Noncentrality parameter. (Input)

FORTRAN 90 Interface

Generic: TNPR (T, DF, DELTA)
Specific: The specific interface names are S_TNPR and D_TNPR.

Description

The noncentral Student's *t* distribution is a generalization of the Student's *t* distribution.

If *w* is a normally distributed random variable with unit variance and mean δ and *u* is a chi-square random variable with ν degrees of freedom that is statistically independent of *w*, then

$$T = w / \sqrt{u / \nu}$$

is a noncentral *t*-distributed random variable with ν degrees of freedom and noncentrality parameter δ, that is, with ν = DF, and δ = DELTA. The probability density function for the noncentral *t*-distribution is:

$$f\left(t,\nu,\delta\right) = \frac{\nu^{\nu/2} e^{-\delta^2/2}}{\sqrt{\pi}\,\Gamma\left(\nu/2\right)\left(\nu + t^2\right)^{(\nu+1)/2}} \sum_{i=0}^{\infty} \Phi_i$$

where

$$\Phi_i = \frac{\Gamma\left(\left(v+i+1\right)/2\right)\left[\delta t\right]^i\left(2/\left(v+t^2\right)\right)^{i/2}}{i!}$$

and t = **T**.

For **δ** = 0, the PDF reduces to the (central) Student's t PDF:

$$f\left(t,v,0\right) = \frac{\Gamma\left(\left(v+1\right)/2\right)\left(1+\left(t^2/v\right)\right)^{-(v+1)/2}}{\sqrt{v\pi}\,\Gamma\left(v/2\right)}$$

and, for t = 0, the PDF becomes:

$$f\left(0,v,\delta\right) = \frac{\Gamma\left(\left(v+1\right)/2\right)e^{-\delta^2/2}}{\sqrt{v\pi}\,\Gamma\left(v/2\right)}$$

Example

This example calculates the noncentral Student's t PDF for a distribution with 2 degrees of freedom and noncentrality parameter **δ** = 10.

```
      USE TNPR_INT
      USE UMACH_INT
      IMPLICIT NONE

      INTEGER  :: NOUT, I
      REAL     :: X(6)=(/ -.5, 1.5, 3.5, 7.5, 51.5, 99.5 /)
      REAL     :: DF, DELTA, PDFV

      CALL UMACH (2, NOUT)
      DF = 2.0
      DELTA = 10.0

      WRITE (NOUT,'("DF: ", F4.0, "  DELTA: ", F4.0 // &
             "   X        PDF(X)")') DF, DELTA

      DO I = 1, 6
         PDFV = TNPR(X(I), DF, DELTA)
         WRITE (NOUT,'(1X, F4.1, 2X, E12.5)') X(I), PDFV
      END DO
      END
```

Output

```
 DF:   2.  DELTA:  10.

    X        PDF(X)
 -0.5   0.16399E-23
  1.5   0.74417E-09
  3.5   0.28972E-02
```

```
      7.5   0.78853E-01
     51.5   0.14215E-02
     99.5   0.20290E-03
```

UNDF

This function evaluates the uniform cumulative distribution function.

Function Return Value

UNDF — Function value, the probability that a uniform random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the uniform cumulative distribution function is to be evaluated. (Input)

A — Location parameter of the uniform cumulative distribution function. (Input)

B — Value used to compute the scale parameter (B - A) of the uniform cumulative distribution function. (Input)

FORTRAN 90 Interface

Generic: UNDF (X, A, B)

Specific: The specific interface names are S_UNDF and D_UNDF.

FORTRAN 77 Interface

Single: UNDF (X, A, B)

Double: The double precision name is DUNDF.

Description

The function UNDF evaluates the uniform cumulative distribution function with location parameter A and scale parameter (B - A). The function definition is

$$F\left(x|A,B\right) = \begin{cases} 0, & \text{if } x < A \\ \frac{x-A}{B-A}, & \text{if } A \le x \le B \\ 1, & \text{if } x > B \end{cases}$$

Example

In this example, we evaluate the probability function at **X** = 0.65, **A** = 0.25, **B** = 0.75.

```
          USE UMACH_INT
          USE UNDF_INT
          IMPLICIT NONE
          INTEGER NOUT
          REAL X, A, B, PR
          CALL UMACH(2, NOUT)
          X = 0.65
          A = 0.25
          B = 0.75
          PR = UNDF(X, A, B)
          WRITE (NOUT, 99999) X, A, B, PR
    99999 FORMAT (' UNDF(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
          END
```

Output

```
     UNDF(0.65, 0.25, 0.75) = 0.8000
```

UNIN

This function evaluates the inverse of the uniform cumulative distribution function.

Function Return Value

UNIN — Function value, the value of the inverse of the cumulative distribution function. (Output)

Required Arguments

P — Probability for which the inverse of the uniform cumulative distribution function is to be evaluated. (Input)

A — Location parameter of the uniform cumulative distribution function. (Input)

B — Value used to compute the scale parameter (B - A) of the uniform cumulative distribution function. (Input)

FORTRAN 90 Interface

Generic: `UNIN` (P, A, B)

Specific: The specific interface names are `S_UNIN` and `D_UNIN`.

FORTRAN 77 Interface

Single: `UNIN` (P, A, B)

Double: The double precision name is `DUNIN`.

Description

The function `UNIN` evaluates the inverse distribution function of a uniform random variable with location parameter A and scale parameter (B - A).

Example

In this example, we evaluate the inverse probability function at P = 0.80, A = 0.25, B = 0.75.

```
      USE UMACH_INT
      USE UNIN_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, B, P
      CALL UMACH(2, NOUT)
      P = 0.80
      A = 0.25
      B = 0.75
      X = UNIN(P, A, B)
      WRITE (NOUT, 99999) P, A, B, X
99999 FORMAT (' UNIN(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 UNIN(0.80, 0.25, 0.75) = 0.6500
```

UNPR

This function evaluates the uniform probability density function.

Function Return Value

UNPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the uniform probability density function is to be evaluated. (Input)

A — Location parameter of the uniform probability function. (Input)

B — Value used to compute the scale parameter (B - A) of the uniform probability density function. (Input)

FORTRAN 90 Interface

Generic: UNPR (X, A, B)

Specific: The specific interface names are S_UNPR and D_UNPR.

FORTRAN 77 Interface

Single: UNPR (X, A, B)

Double: The double precision name is DUNPR.

Description

The function UNPR evaluates the uniform probability density function with location parameter A and scale parameter (B - A), defined

$$f\left(x|A,B\right) = \begin{cases} \frac{1}{B-A} & \text{for } A \le x \le B \\ 0 & \text{otherwise} \end{cases}$$

Example

In this example, we evaluate the uniform probability density function at **X** = 0.65, **A** = 0.25, **B** = 0.75.

```
      USE UMACH_INT
      USE UNPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, B, PR
      CALL UMACH(2, NOUT)
      X = 0.65
      A = 0.25
      B = 0.75
      PR = UNPR(X, A, B)
      WRITE (NOUT, 99999) X, A, B, PR
99999 FORMAT (' UNPR(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 UNPR(0.65, 0.25, 0.75) = 2.0000
```

WBLDF

This function evaluates the Weibull cumulative distribution function.

Function Return Value

WBLDF — Function value, the probability that a Weibull random variable takes a value less than or equal to X. (Output)

Required Arguments

X — Argument for which the Weibull cumulative distribution function is to be evaluated. (Input)

A — Scale parameter. (Input)

B — Shape parameter. (Input)

FORTRAN 90 Interface

Generic: `WBLDF` (X, A, B)
Specific: The specific interface names are `S_WBLDF` and `D_WBLDF`.

FORTRAN 77 Interface

Single: `WBLDF` (X, A, B)
Double: The double precision name is `DWBLDF`.

Description

The function `WBLDF` evaluates the Weibull cumulative distribution function with scale parameter A and shape parameter B, defined

$$F\left(x|a,b\right) = 1 - e^{-\left(\frac{x}{a}\right)^b}$$

To deal with potential loss of precision for small values of $\left(\frac{x}{a}\right)^b$, the difference expression for p is re-written as

$$u = \left(\frac{x}{a}\right)^b, \quad p = u\left[\frac{\left(e^{-u} - 1\right)}{-u}\right]$$

and the right factor is accurately evaluated using **EXPRL**.

Example

In this example, we evaluate the Weibull cumulative distribution function at **X** = 1.5, **A** = 1.0, **B** = 2.0.

```
      USE UMACH_INT
      USE WBLDF_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, B, PR
      CALL UMACH(2, NOUT)
      X = 1.5
      A = 1.0
      B = 2.0
      PR = WBLDF(X, A, B)
      WRITE (NOUT, 99999) X, A, B, PR
99999 FORMAT (' WBLDF(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 WBLDF(1.50, 1.00, 2.00) = 0.8946
```

WBLIN

This function evaluates the inverse of the Weibull cumulative distribution function.

Function Return Value

WBLIN — Function value, the value of the inverse of the Weibull cumulative distribution distribution function. (Output)

Required Arguments

P — Probability for which the inverse of the Weibull cumulative distribution function is to be evaluated. (Input)

A — Scale parameter. (Input)

B — Shape parameter. (Input)

FORTRAN 90 Interface

Generic: `WBLIN` (`P`, `A`, `B`)

Specific: The specific interface names are `S_WBLIN` and `D_WBLIN`.

FORTRAN 77 Interface

Single: `WBLIN` (`P`, `A`, `B`)

Double: The double precision name is `DWBLIN`.

Description

The function `WBLIN` evaluates the inverse distribution function of a Weibull random variable with scale parameter `A` and shape parameter `B`.

Example

In this example, we evaluate the inverse probability function at `P` = 0.8946, `A` = 1.0, `B` = 2.0.

```
      USE UMACH_INT
      USE WBLIN_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, B, P
      CALL UMACH(2, NOUT)
      P = 0.8946
      A = 1.0
      B = 2.0
      X = WBLIN(P, A, B)
      WRITE (NOUT, 99999) P, A, B, X
99999 FORMAT (' WBLIN(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 WBLIN(0.8946, 1.00, 2.00) = 1.5000
```

WBLPR

This function evaluates the Weibull probability density function.

Function Return Value

WBLPR — Function value, the value of the probability density function. (Output)

Required Arguments

X — Argument for which the Weibull probability density function is to be evaluated. (Input)

A — Scale parameter. (Input)

B — Shape parameter. (Input)

FORTRAN 90 Interface

Generic: **WBLPR** (**X**, **A**, **B**)
Specific: The specific interface names are **S_WBLPR** and **D_WBLPR**.

FORTRAN 77 Interface

Single: **WBLPR** (**X**, **A**, **B**)
Double: The double precision name is **DWBLPR**.

Description

The function **WBLPR** evaluates the Weibull probability density function with scale parameter **A** and shape parameter **B**, defined

$$f\left(x|a,b\right) = \frac{b}{a}\left(\frac{x}{a}\right)^{b-1} e^{-\left(\frac{x}{a}\right)^{b}}, \quad a,b > 0$$

Example

In this example, we evaluate the Weibull probability density function at **X** = 1.5, **A** = 1.0, **B** = 2.0.

```
      USE UMACH_INT
      USE WBLPR_INT
      IMPLICIT NONE
      INTEGER NOUT
      REAL X, A, B, PR`
      CALL UMACH(2, NOUT)
      X = 1.5
      A = 1.0
      B = 2.0
      PR = WBLPR(X, A, B)
      WRITE (NOUT, 99999) X, A, B, PR
99999 FORMAT (' WBLPR(', F4.2, ', ', F4.2, ', ', F4.2, ') = ', F6.4)
      END
```

Output

```
 WBLPR(1.50, 1.00, 2.00) = 0.3162
```

GCDF

This function evaluates a general continuous cumulative distribution function given ordinates of the density.

Function Return Value

GCDF — Function value, the probability that a random variable whose density is given in **F** takes a value less than or equal to **X0**. (Output)

Required Arguments

X0 —Point at which the cumulative distribution function is to be evaluated. (Input)

X — Array containing the abscissas or the endpoints. (Input)
If **IOPT** = 1 or 3, **X** is of length 2. If **IOPT** = 2 or 4, **X** is of length **M**. For **IOPT** = 1 or 3, **X**(1) contains the lower endpoint of the support of the distribution and **X**(2) is the upper endpoint. For **IOPT** = 2 or 4, **X** contains, in strictly increasing order, the abscissas such that **X**(**I**) corresponds to **F**(**I**).

F — Vector of length **M** containing the probability density ordinates corresponding to increasing abscissas. (Input)
If **IOPT** = 1 or 3, for **I** = 1, 2, ..., **M**, **F**(**I**) corresponds to **X**(1) + (**I** - 1) * (**X**(2) - **X**(1))/(**M** - 1); otherwise, **F** and **X** correspond one for one.

Optional Arguments

IOPT — Indicator of the method of interpolation. (Input)
Default: **IOPT** = 1.

IOPT	Interpolation Method
1	Linear interpolation with equally spaced abscissas.
2	Linear interpolation with possibly unequally spaced abscissas.
3	A cubic spline is fitted to equally spaced abscissas.
4	A cubic spline is fitted to possibly unequally spaced abscissas.

M —Number of ordinates of the density supplied. (Input)
M must be greater than 1 for linear interpolation (**IOPT** = 1 or 2) and greater than 3 if a curve is fitted through the ordinates (**IOPT** = 3 or 4).
Default: **M** = size (**F**,1).

FORTRAN 90 Interface

Generic: **GCDF** (**X0**, **X**, **F** [, ...])
Specific: The specific interface names are **S_GCDF** and **D_GCDF**.

FORTRAN 77 Interface

Single: **GCDF** (**X0**, **IOPT**, **M**, **X**, **F**)
Double: The double precision name is **DGCDF**.

Description

Function **GCDF** evaluates a continuous distribution function, given ordinates of the probability density function. It requires that the range of the distribution be specified in **X**. For distributions with infinite ranges, endpoints must be chosen so that most of the probability content is included. The function **GCDF** first fits a curve to the points given in **X** and **F** with either a piecewise linear interpolant or a C^1 cubic spline interpolant based on a method by Akima (1970). Function **GCDF** then determines the area, A, under the curve. (If the distribution were of finite range and if the fit were exact, this area would be 1.0.) Using the same fitted curve, **GCDF** next determines the area up to the point x_0 (= **X0**). The value returned is the area up to x_0 divided by **A**. Because of the scaling by A, it is not assumed that the integral of the density defined by **X** and **F** is 1.0. For most distributions, it is likely that better approximations to the distribution function are obtained when **IOPT** equals 3 or 4, that is, when a cubic spline is used to approximate the function. It is also likely that better approximations can be obtained when the abscissas are chosen more densely over regions where the density and its derivatives (when they exist) are varying greatly.

Comments

1. If **IOPT** = 3, automatic workspace usage is:

GCDF	6 * **M** units, or
DGCDF	11 * **M** units.

2. If **IOPT** = 4, automatic workspace usage is

GCDF	5 * M units, or
DGCDF	9 * M units.

3. Workspace may be explicitly provided, if desired, by the use of G4DF/DG4DF. The reference is:

   G4DF (P, IOPT, M, X, F, WK, IWK)

   The arguments in addition to those of GCDF are:

   WK — Work vector of length 5 * M if IOPT = 3, and of length 4 * M if IOPT = 4.

   IWK — Work vector of length M.

Example

In this example, we evaluate the beta distribution function at the point 0.6. The probability density function of a beta random variable with parameters *p* and *q* is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \le x \le 1$$

where Γ(·) is the gamma function. The density is equal to 0 outside the interval [0, 1]. We compute a constant multiple (we can ignore the constant gamma functions) of the density at 300 equally spaced points and input this information in X and F. Knowing that the probability density of this distribution is very peaked in the vicinity of 0.5, we could perhaps get a better fit by using unequally spaced abscissas, but we will keep it simple. Note that this is the same example as one used in the description of routine BETDF. The result from BETDF would be expected to be more accurate than that from GCDF since BETDF is designed specifically for this distribution.

```
      USE UMACH_INT
      USE GCDF_INT
      IMPLICIT   NONE
      INTEGER    M
      PARAMETER  (M=300)
!
      INTEGER    I, IOPT, NOUT
      REAL       F(M), H, P, PIN1, QIN1, X(2), X0, XI
!
      CALL UMACH (2, NOUT)
      X0   = 0.6
      IOPT = 3
!                                Initializations for a beta(12,12)
!                                distribution.
      PIN1 = 11.0
      QIN1 = 11.0
      XI   = 0.0
      H    = 1.0/(M-1.0)
      X(1) = XI
      F(1) = 0.0
      XI   = XI + H
!                                Compute ordinates of the probability
!                                density function.
      DO 10  I=2, M - 1
         F(I) = XI**PIN1*(1.0-XI)**QIN1
         XI   = XI + H
   10 CONTINUE
      X(2) = 1.0
      F(M) = 0.0
      P    = GCDF(X0, X, F, IOPT=IOPT)
      WRITE (NOUT,99999) P
99999 FORMAT (' The probability that X is less than 0.6 is ', F6.4)
      END
```

Output

```
 The probability that X is less than 0.6 is 0.8364
```

GCIN

This function evaluates the inverse of a general continuous cumulative distribution function given ordinates of the density.

Function Return Value

GCIN — Function value. (Output)
The probability that a random variable whose density is given in **F** takes a value less than or equal to **GCIN** is **P**.

Required Arguments

P —Probability for which the inverse of the cumulative distribution function is to be evaluated. (Input)
P must be in the open interval (0.0, 1.0).

X —Array containing the abscissas or the endpoints. (Input)
If **IOPT** = 1 or 3, **X** is of length 2. If **IOPT** = 2 or 4, **X** is of length **M**. For **IOPT** = 1 or 3, **X**(1) contains the lower endpoint of the support of the distribution and **X**(2) is the upper endpoint. For **IOPT** = 2 or 4, **X** contains, in strictly increasing order, the abscissas such that **X**(**I**) corresponds to **F**(**I**).

F —Vector of length **M** containing the probability density ordinates corresponding to increasing abscissas. (Input)
If **IOPT** = 1 or 3, for **I** = 1, 2, …, **M**, **F**(**I**) corresponds to **X**(1) + (**I** - 1) * (**X**(2) - **X**(1))/(**M** - 1); otherwise, **F** and **X** correspond one for one.

Optional Arguments

IOPT — Indicator of the method of interpolation. (Input)
Default: **IOPT** = 1.

IOPT	Interpolation Method
1	Linear interpolation with equally spaced abscissas.
2	Linear interpolation with possibly unequally spaced abscissas.
3	A cubic spline is fitted to equally spaced abscissas.
4	A cubic spline is fitted to possibly unequally spaced abscissas.

M —Number of ordinates of the density supplied. (Input)
M must be greater than 1 for linear interpolation (**IOPT** = 1 or 2) and greater than 3 if a curve is fitted through the ordinates (**IOPT** = 3 or 4).
Default: **M** = size (**F**,1).

FORTRAN 90 Interface

Generic: **GCIN** (**P**, **X**, **F** [, ...])
Specific: The specific interface names are **S_GCIN** and **D_GCIN**.

FORTRAN 77 Interface

Single: **GCIN** (**P**, **IOPT**, **M**, **X**, **F**)
Double: The double precision name is **DGCIN**.

Description

Function **GCIN** evaluates the inverse of a continuous distribution function, given ordinates of the probability density function. The range of the distribution must be specified in **X**. For distributions with infinite ranges, endpoints must be chosen so that most of the probability content is included.

The function **GCIN** first fits a curve to the points given in **X** and **F** with either a piecewise linear interpolant or a *C* cubic spline interpolant based on a method by Akima (1970). Function **GCIN** then determines the area, *A*, under the curve. (If the distribution were of finite range and if the fit were exact, this area would be 1.0.) It next finds the maximum abscissa up to which the area is less than *AP* and the minimum abscissa up to which the area is greater than *AP* . The routine then interpolates for the point corresponding to *AP*. Because of the scaling by *A*, it is not assumed that the integral of the density defined by **X** and **F** is 1.0.

For most distributions, it is likely that better approximations to the distribution function are obtained when **IOPT** equals 3 or 4, that is, when a cubic spline is used to approximate the function. It is also likely that better approximations can be obtained when the abscissas are chosen more densely over regions where the density and its derivatives (when they exist) are varying greatly.

Comments

1. If **IOPT** = 3, automatic workspace usage is

GCIN	6 * **M** units, or
DGCIN	11 * **M** units.

2. If **IOPT** = 4, automatic workspace usage is

   GCIN 5 * **M** units, or

   DGCIN 9 * **M** units.

3. Workspace may be explicitly provided, if desired, by the use of **G3IN**/**DG3IN**. The reference is:

   G3IN (**P**, **IOPT**, **M**, **X**, **F**, **WK**, **IWK**)

   The arguments in addition to those of **GCIN** are:

   WK — Work vector of length 5 * **M** if **IOPT** = 3, and of length 4 * **M** if **IOPT** = 4.

   IWK — Work vector of length **M**.

Example

In this example, we find the 90-th percentage point for a beta random variable with parameters 12 and 12. The probability density function of a beta random variable with parameters *p* and *q* is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \le x \le 1$$

where $\Gamma(\cdot)$ is the gamma function. The density is equal to 0 outside the interval [0, 1]. With $p = q$, this is a symmetric distribution. Knowing that the probability density of this distribution is very peaked in the vicinity of 0.5, we could perhaps get a better fit by using unequally spaced abscissas, but we will keep it simple and use 300 equally spaced points. Note that this is the same example that is used in the description of routine **BETIN**. The result from **BETIN** would be expected to be more accurate than that from **GCIN** since **BETIN** is designed specifically for this distribution.

```
          USE GCIN_INT
          USE UMACH_INT
          IMPLICIT   NONE
          INTEGER    M
          PARAMETER  (M=300)
    !
          INTEGER    I, IOPT, NOUT
          REAL       C, F(M), H, P, PIN, PIN1, QIN, QIN1, X(2), X0, XI, BETA
    !
          CALL UMACH (2, NOUT)
          P    = 0.9
          IOPT = 3
    !                                Initializations for a beta(12,12)
    !                                distribution.
          PIN  = 12.0
          QIN  = 12.0
          PIN1 = PIN - 1.0
          QIN1 = QIN - 1.0
          C    = 1.0/BETA(PIN,QIN)
```

```
      XI   = 0.0
      H    = 1.0/(M-1.0)
      X(1) = XI
      F(1) = 0.0
      XI   = XI + H
!                                Compute ordinates of the probability
!                                density function.
      DO 10  I=2, M - 1
         F(I) = C*XI**PIN1*(1.0-XI)**QIN1
         XI   = XI + H
   10 CONTINUE
      X(2) = 1.0
      F(M) = 0.0
      X0   = GCIN(P,X,F,IOPT=IOPT)
      WRITE (NOUT,99999) X0
99999 FORMAT (' X is less than ', F6.4, ' with probability 0.9.')
      END
```

Output

```
 X is less than 0.6304 with probability 0.9.
```

GFNIN

This function evaluates the inverse of a general continuous cumulative distribution function given in a subprogram.

Function Return Value

GFNIN — The inverse of the function F at the point P. (Output)
F(GFNIN) is "close" to P.

Required Arguments

F — User-supplied FUNCTION to be inverted. F must be continuous and strictly monotone. The form is F(X), where
X — The argument to the function. (Input)
F — The value of the function at X. (Output)
F must be declared EXTERNAL in the calling program.

P — The point at which the inverse of F is desired. (Input)

GUESS — An initial estimate of the inverse of F at P. (Input)

Optional Arguments

EPS — Convergence criterion. (Input)
When the relative change in GFNIN from one iteration to the next is less than EPS, convergence is assumed. A common value for EPS is 0.0001. Another common value is 100 times the machine epsilon.
Default: EPS = 100 times the machine epsilon.

FORTRAN 90 Interface

Generic: GFNIN (F, P, GUESS [, ...])
Specific: The specific interface names are S_GFNIN and D_GFNIN.

FORTRAN 77 Interface

Single: **GFNIN** (**F**, **P**, **EPS**, **GUESS**)

Double: The double precision name is **DGFNIN**.

Description

Function **GFNIN** evaluates the inverse of a continuous, strictly monotone function. Its most obvious use is in evaluating inverses of continuous distribution functions that can be defined by a FORTRAN function. If the distribution function cannot be specified in a FORTRAN function, but the density function can be evaluated at a number of points, then routine **GCIN** can be used.

Function **GFNIN** uses regula falsi and/or bisection, possibly with the Illinois modification (see Dahlquist and Bjorck 1974). A maximum of 100 iterations are performed.

Comments

1. Informational errors

Type	Code	Description
4	1	After 100 attempts, a bound for the inverse cannot be determined. Try again with a different initial estimate.
4	2	No unique inverse exists.
4	3	Over 100 iterations have occurred without convergence. Convergence is assumed.

2. The function to be inverted need not be a distribution function, it can be any continuous, monotonic function.

Example

In this example, we find the 99–th percentage point for an *F* random variable with 1 and 7 degrees of freedom. (This problem could be solved easily using routine **FIN**. Compare the example for **FIN**). The function to be inverted is the *F* distribution function, for which we use routine **FDF**. Since **FDF** requires the degrees of freedom in addition to the point at which the function is evaluated, we write another function **F** that receives the degrees of freedom via a common block and then calls **FDF**. The starting point (initial guess) is taken as two standard deviations above the mean (since this would be a good guess for a normal distribution). It is not necessary to supply such a good guess. In this particular case, an initial estimate of 1.0, for example, yields the same answer in essentially the same number of iterations. (In fact, since the *F* distribution is skewed, the initial guess, 7.0, is really not that close to the final answer.)

```
      USE UMACH_INT
      USE GFNIN_INT
      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       DFD, DFN, F, F0, GUESS, P, SQRT
      COMMON     /FCOM/ DFN, DFD
      INTRINSIC  SQRT
      EXTERNAL   F
!
      CALL UMACH (2, NOUT)
      P   = 0.99
      DFN = 1.0
      DFD = 7.0
!                                Compute GUESS as two standard
!                                deviations above the mean.
      GUESS = DFD/(DFD-2.0) + 2.0*SQRT(2.0*DFD*DFD*(DFN+DFD-2.0)/(DFN* &
             (DFD-2.0)**2*(DFD-4.0)))
      F0    = GFNIN(F,P,GUESS)
      WRITE (NOUT,99999) F0
99999 FORMAT (' The F(1,7) 0.01 critical value is ', F6.3)
      END
!
      REAL FUNCTION F (X)
      REAL       X
!
      REAL       DFD, DFN, FDF
      COMMON     /FCOM/ DFN, DFD
      EXTERNAL   FDF
!
      F = FDF(X,DFN,DFD)
      RETURN
      END
```

Output

```
 The F(1,7) 0.01 critical value is 12.246
```

MLE

more...

Calculates maximum likelihood estimates for the parameters of one of several univariate probability distributions.

Required Arguments

X— Array containing the data. (Input)

IPDF — Specifies the probability density function. (Input)

Distribution		`size(PARAM)`	`i`	`i`
Discrete uniform	0	1	1	scale - upper limit
Bernoulli	1	1	1	probability of success (mean)
Binomial	2	1	1	probability of success
Negative binomial	3	1	1	probability of success
Poisson	4	1	1	location (mean) - θ
Geometric	5	1	1	probability of success
Continuous uniform	6	2	1 2	scale - lower boundary scale - upper boundary
Beta	7	2	1 2	shape - p shape - q
Exponential	8	1	1	scale - b
Gamma	9	2	1 2	shape - k scale - θ
Weibull	10	2	1 2	scale - λ shape - k
Rayleigh	11	1	1	scale - α

Distribution		size(PARAM)	i	i
Extreme value	12	2	1 2	location - μ scale - σ
Generalized extreme value	13	3	1 2 3	location - μ scale - σ shape - β
Pareto	14	2	1 2	scale (lower boundary) x_m shape - k
Generalized Pareto	15	2	1 2	scale - σ shape - α
Normal	16	2	1 2	location (mean) - μ scale (variance) - σ^2
Log-normal	17	2	1 2	location (mean of log(x)) - μ scale (variance of log(x)) - σ^2
Logistic	18	2	1 2	location (mean) - μ scale - s
Log-logistic	19	2	1 2	scale (exp(mean)) - e^{μ} shape - β
Inverse Gaussian	20	2	1 2	location (mean) - μ shape - λ

PARAM — Array of length *p* containing the parameter values, where *p* = size(PARAM) denotes the number of parameters (see the IPDF table above). On input, the values of PARAM are used as starting values, when USEMM = .FALSE.. On output, final parameter estimates replace the starting values. (Input/Output)

Optional Arguments

NOBS — Number of observations to use in the analysis. (Input)
Default: NOBS = size(X).

PARAMLB — Array of length *p* containing the lower bounds of the parameters. (Input)
Default: The default lower bound depends on the range of the parameter. That is, if PARAM(i) is positive, PARAMLB(i) = 0.01. If PARAM(i) is non-negative (≥0), then PARAMLB(i) = 0.0. If PARAM(i) can be any real value, then PARAMLB(i) = -10000.00. Exceptions are PARAMLB(i) = 0.25 for the scale parameter of the extreme value distribution, PARAMLB(i) = -5.0 for the shape parameter of the generalized Pareto distribution, and PARAMLB(i) = -10.0 for the shape parameter of the generalized extreme value distribution.

PARAMUB — Array of length *p* containing the upper bounds of the parameters. (Input)
Default: PARAMUB(i) = 10000.00. Exceptions are PARAMUB(i) = 5.0 for the shape parameter of the generalized Pareto distribution and PARAMUB(i) = 10.0 for the shape parameter of the generalized extreme value distribution

USEMM — Logical. If .true., starting values are set to the method of moments estimates. (Input)
If USEMM = .FALSE., PARAM values are used.
Default: USEMM = .TRUE..

XSCALE — Array of length *p* containing the scaling factors for the parameters. XSCALE is used in the routine BCONF mainly in scaling the gradient and the distance between two points. See BCONF in the Math Libray, *Chapter 8, "Optimization"* for details.
Default: XSCALE =1.0.

MAXIT — Maximum number of iterations. (Input)
Default: MAXIT = 100.

MAXFUN — Maximum number of function evaluations. (Input)
Default: MAXFUN = 400.

MAXGRAD — Maximum number of gradient evaluations. (Input)
Default: MAXGRAD = 400.

IPRINT — Printing option (Input)

IPRINT	Action
0	No printing
1	Print final results only
2	Print intermediate and final results

Default: IPRINT = 0.

MLOGLIKE — Minus log-likelihood evaluated at the parameter estimates. (Output)

SE — Array of length *p* containing the standard errors of the parameter estimates. (Output)

HESS — Array of size *p* by *p* containing the Hessian matrix. (Output)

FORTRAN 90 Interface

Generic: CALL MLE (X,IPDF,PARAM [, ...])
Specific: The specific interface names are S_MLE and D_MLE.

Description

Routine **MLE** calculates maximum likelihood estimates for the parameters of a univariate probability distribution, where the distribution is one specified by **IPDF** and where the input data *X* is (assumed to be) a random sample from that distribution.

Let $\{x_i, i = 1, \ldots, N\}$ represent a random sample from a probability distribution with density function $f(x|\boldsymbol{\theta})$, which depends on a vector $\boldsymbol{\theta} \in \Re^p$ containing the values of the parameters of the distribution. The values in $\boldsymbol{\theta}$ are fixed but unknown and the problem is to find an estimate for $\boldsymbol{\theta}$ given the sample data.

The likelihood function is defined to be the product

$$L(\theta;\{x_i;\ i = 1,\ ...,\ N\}) = \prod_{i=1,\ldots,N} f(x_i;\ \theta)$$

The estimator

$$\begin{aligned}\hat{\theta}_{MLE} &= \text{argmax}_\theta\ L(\theta;\ \{x_1,x_2,...x_N\}) \\ &= \text{argmax}_\theta \prod_{i=1,\ldots,N} f(x_i;\ \theta) \\ &= \text{argmax}_\theta \sum_{i=1,\ldots,N} \log(f(x_i;\ \theta)).\end{aligned}$$

That is, the estimator that maximizes *L* also maximizes log *L* and is the maximum likelihood estimate, or *MLE* for $\boldsymbol{\theta}$.

The likelihood problem is in general a constrained non-linear optimization problem, where the constraints are determined by the permissible range of $\boldsymbol{\theta}$. In a few situations, the problem has a closed form solution. Otherwise, **MLE** uses the numerical method as documented in routine **BCONF** (see *Chapter 8, "Optimization"* in the Math Library for details) to solve the likelihood problem. If **USEMM** is **.TRUE.** (the default), method of moments estimates serve as starting values of the parameters. In some cases, method of moments estimators may not exist, such as when certain moments of the true distribution do not exist; thus it is possible that the starting values are not truly method of moments estimates. If **USEMM** is set to **.FALSE.**, input values of **PARAM** are used as starting values.

Upper and lower bounds, when needed for the optimization, have default values for each selection of **IPDF** (defaults will vary depending on the allowable range of the parameters). It is possible that the optimization will fail. In such cases, the user may try adjusting upper and lower bounds using the optional parameters **PARAMLB**, **PARAMUB**, or adjusting up or down the scaling factors in **XSCALE**, which can sometimes help the optimization converge.

Standard errors and covariances are supplied, in most cases, using the asymptotic properties of *ML* estimators. Under some general regularity conditions, *ML* estimates are consistent and asymptotically normally distributed with variance-covariance equal to the inverse *Fisher's Information* matrix evaluated at the true value of the parameter, θ_0:

$$Var\left(\hat{\theta}\right) = I(\theta_0)^{-1} = -E\left[\frac{\partial^2 \log L}{\partial \theta^2}\right]_{\theta_0}^{-1}$$

MLE approximates the asymptotic variance using the negative inverse Hessian evaluated at the *ML* estimate:

$$Var(\hat{\theta}) \approx -\left[\frac{\partial^2 \log L}{\partial \theta^2}\right]_{\theta=\hat{\theta}_{MLE}}^{-1}$$

The Hessian is approximated numerically for all but a few cases where it can be determined in closed form.

In cases when the asymptotic result does not hold, standard errors may be available from the known sampling distribution. For example, the *ML* estimate of the Pareto distribution location parameter is the minimum of the sample. The variance is estimated using the known sampling distribution of the minium, or first order-statistic for the Pareto distribution.

For further details regarding the properties of the estimators and the theory of the maximum likelihood method, see Kendall and Stuart (1979). The different probability distributions have wide coverage in the statistical literature. See Johnson & Kotz (1970a, 1970b, or later editions).

Parameter estimation (including maximum likelihoood) for the generalized Pareto distribution is studied in Hosking and Wallis (1987) and Giles and Feng (2009), and estimation for the generalized extreme value distribution is treated in Hosking, Wallis, and Wood (1985).

Comments

1. The location parameter is not estimated for the generalized Pareto distribution (**IPDF=15**). Instead, the minimum of the sample is subtracted from each observation before the estimation procedure.
2. Only the probability of success parameter is estimated for the binomial and negative binomial distributions, (**IPDF=2**,**3**). The number of trials and the number of failures, respectively, must be provided in **PARAM**(**1**) on input.
3. **MLE** issues an error if missing or **NaN** values are encountered in the input data. Missing or **NaN** values should be removed before calling **MLE**.
4. Informational errors

Type	Code	Description
3	1	The Hessian is not calculated for the negative binomial distribution.
3	2	Hessian is not used to calculate the standard errors of the estimates for the continuous uniform distribution.
3	3	The Hessian is not used to calculate the standard errors of the estimates for the Pareto distribution.
3	4	For the Pareto distribution, the Hessian cannot be calculated because the parameter estimate is 0.

Examples

Example 1

The data are $N = 100$ observations generated from the logistic distribution with location parameter $\mu = 0.85$ and scale parameter $\sigma = 0.5$.

```
          use mle_int
          implicit none

          integer, parameter :: ipdf=18, npar=2
          real(kind(1e0)) :: param(npar), stderr(npar), hess(npar,npar)
          real(kind(1e0)) :: fval

    !                       Logistic distribution mu = 0.85, sigma=0.5
          real(kind(1e0)) :: log1(100)
          data log1 /&
             2.020394,  2.562315, -0.5453395, 1.258546, 0.7704533, &
             0.3662717, 0.6885536, 2.619634,  -0.49581, 2.972249, &
             0.5356222, 0.4262079, 1.023666, 0.8286033, 1.319018, &
             2.123659, 0.3904647, -0.1196832, 1.629261, 1.069602, &
             0.9438083, 1.314796, 1.404453, -0.5496156, 0.8326595, &
             1.570288, 1.326737, 0.9619384, -0.1795268, 1.330161, &
             -0.2916453, 0.7430826, 1.640854, 1.582755, 1.559261, &
             0.6177695, 1.739638, 1.308973, 0.568709, 0.2587071, &
             0.745583, 1.003815, 1.475413, 1.444586, 0.4515438, &
             1.264374, 1.788313, 1.062330, 2.126034, 0.3626510, &
             1.365612, 0.5044735, 2.51385, 0.7910572, 0.5932584, &
             1.140248, 2.104453, 1.345562, -0.9120445, 0.0006519341, &
             1.049729, -0.8246097, 0.8053433, 1.493787, -0.5199705, &
             2.285175, 0.9005916, 2.108943, 1.40268, 1.813626, &
             1.007817, 1.925250, 1.037391, 0.6767235, -0.3574937, &
             0.696697, 1.104745, -0.7691124, 1.554932, 2.090315, &
             0.60919, 0.4949385, -2.449544, 0.668952, 0.9480486, &
             0.9908558, -1.495384, 2.179275, 0.1858808, -0.3715074, &
             0.1447150, 0.857202, 1.805844, 0.405371, 1.425935, &
             0.3187476, 1.536181, -0.6352768, 0.5692068, 1.706736/

             param = 1.0
             stderr = 0.0
             hess = 0.0
```

```
            call mle(log1,ipdf,param,iprint=2,usemm=.true., &
                     mloglike=fval,se=stderr,hess=hess)
         end
```

Output

```
 Univariate Statistics from UVSTA

 Variable       Mean      Variance     Std. Dev.      Skewness      Kurtosis
     1        0.9068        0.8600        0.9274       -0.6251        0.9725

 Variable    Minimum       Maximum         Range    Coef. Var.         Count
     1       -2.4495        2.9722        5.4218        1.0227      100.0000

 Variable     Lower CLM     Upper CLM     Lower CLV     Upper CLV
     1           0.7228        1.0908        0.6629        1.1606

 Maximum likelihood estimation for the logistic distribution

 Starting estimates:   0.90677   0.51128

 Initial -log-likelihood:  132.75304

 -Log-likelihood  132.61487

 MLE for parameter      1      0.95341

 MLE for parameter      2      0.50944

 Std error for parameter   1      0.08845

 Std error for parameter   2      0.04364

      Hessian
         1       2
1  -127.9    -5.7
2    -5.7  -525.4
```

Example 2

The data are *N* = 100 observations generated from the generalized extreme value distribution with location parameter $\mu = 0$, scale parameter $\sigma = 1.0$, and shape parameter $\xi = -0.25$.

```
      use mle_int
      implicit none

      integer, parameter :: ipdf=13, npar=3
      real(kind(1e0)) :: param(npar), stderr(npar), &
         hess(npar,npar)
      real(kind(1e0)) :: fval

!                                 Generalized Extreme Value
!                                 oc = 0, scale =1, shape = =-0.25
      real(kind(1e0)) :: gev(100)
      data gev/ &
         0.7688048, 0.1944504, -0.2992029, -0.3853738, -1.185593, &
         0.3056149, -0.4407711, 0.5001115, 0.3635027, -1.058632, &
```

```
            -0.2927695, -0.3205969, 0.03367599, 0.8850839, 1.860485, &
            0.4841038, 0.5421101, 1.883694, 1.707392, 0.2166106, &
            1.537204, 1.340291, 0.4589722, 1.616080, -0.8389288, &
            0.7057426, 1.532988, 1.161350, 0.9475416, 0.4995294, &
            -0.2392898, 0.8167126, 0.992479, -0.8357962, -0.3194499, &
            1.233603, 2.321555, -0.3715629, -0.1735171, 0.4624801, &
            -0.6249577, 0.7040129, -0.3598889, 0.7121399, -0.5178735, &
            -1.069429, 0.7169358, 0.4148059, 1.606248, -0.4640152, &
            1.463425, 0.9544342, -1.383239, 0.1393160, 0.622689, &
            0.365793, 0.7592438, 0.810005, 0.3483791, 2.375727, &
            -0.08124195, -0.4726068, 0.1496043, 0.4961212, 1.532723, &
            -0.1106993, 1.028553, 0.856018, -0.6634978, 0.3573150, &
            0.06391576, 0.3760349, -0.5998756, 0.4158309, -0.2832369, &
            -1.023551, 1.116887, 1.237714, 1.900794, 0.6010037, &
            1.599663, -0.3341879, 0.5278575, 0.5497694, 0.6392933, &
            0.592865, 1.646261, -1.042950, -1.113611, 1.229645, &
            1.655998, 0.6913992, 0.4548073, 0.4982649, -1.073640, &
            -0.4765107, -0.8692533, -0.8316462, -0.03609102, 0.655814/

!                                   initialize
      param=1.0
      stderr=0.0
      hess = 0.0

      call mle(gev, ipdf, param, iprint=2, usemm=.true., &
         mloglike=fval, se=stderr, hess=hess)

      end
```

Output

```
Univariate Statistics from UVSTA

Variable       Mean      Variance     Std. Dev.      Skewness      Kurtosis
    1        0.3805        0.7484        0.8651       0.05492       -0.6240

Variable    Minimum       Maximum         Range    Coef. Var.         Count
    1       -1.3832        2.3757        3.7590        2.2738      100.0000

Variable     Lower CLM     Upper CLM     Lower CLV     Upper CLV
    1           0.2088        0.5521        0.5769        1.0100

Maximum likelihood estimation for the generalized extreme value distribution

Starting estimates:  -0.00888   0.67451   0.00000

Initial -log-likelihood:  135.43820

-Log-likelihood  126.09403

MLE for parameter      1      0.07500

MLE for parameter      2      0.85115

MLE for parameter      3     -0.27960

Std error for parameter   1      0.09467

Std error for parameter   2      0.07007

Std error for parameter   3      0.06695

         Hessian
        1       2       3
1  -141.1   -51.3  -112.8
2   -51.3  -337.0  -241.0
3  -112.8  -241.0  -438.8
```

Random Number Generation

Routines

18.4 Univariate Continuous Distributions

18.5 Multivariate Distributions

18.6 Order Statistics

18.7 Stochastic Processes

18.8 Samples and Permutations

18.9 Low Discrepancy Sequences

Usage Notes

In the following discussions, the phrases "random numbers," "random deviates," "deviates," and "variates" are used interchangeably. The phrase "pseudorandom" is sometimes used to emphasize that the numbers generated are not really "random" since they result from a deterministic process. The usefulness of pseudorandom numbers derives from the similarity, in a statistical sense, of samples of the pseudorandom numbers to samples of observations from the specified distributions. In short, while the pseudorandom numbers are completely deterministic and repeatable, they *simulate* the realizations of independent and identically distributed random variables.

The Basic Uniform Generators

The random number generators in this chapter use either a multiplicative congruential method, or a generalized feedback shift register (GFSR) method, or a Mersenne Twister method. The selection of the type of generator is made by calling the routine **RNOPT**. If no selection is made explicitly, a multiplicative generator (with multiplier 16807) is used. Whatever distribution is being simulated, uniform (0, 1) numbers are first generated and then transformed if necessary. The generation of the uniform (0, 1) numbers is done by the routine **RNUN**, or by its function analog **RNUNF**. These routines are *portable* in the sense that, given the same seed and for a given type of generator, they produce the same sequence in all computer/compiler environments. There are many other issues that must be considered in developing programs for the methods described below (see Gentle 1981 and 1990).

The Multiplicative Congruential Generators

The form of the multiplicative congruential generators is

$$x_i \equiv cx_{i-1} \bmod \left(2^{31} - 1\right)$$

Each x_i is then scaled into the unit interval (0, 1). If the multiplier, c, is a primitive root modulo $2^{31} - 1$ (which is a prime), then the generator will have maximal period of $2^{31} - 2$. There are several other considerations, however. The lattice structure induced by congruential generators (see Marsaglia 1968) can be assessed by the lattice test of Marsaglia (1972) or the spectral test of Coveyou and MacPherson (1967) (see also Knuth 1981, pages 89-113). Also, empirical studies, such as by Fishman and Moore (1982 and 1986), indicate that different values of multipliers, all of which perform well under the lattice test and the spectral test, may yield quite different performances where the criterion is similarity of samples generated to samples from a true uniform distribution.

There are three possible choices for c in the IMSL generators: 16807 (which is 7^5), 397204094 (which is $2 \cdot 7^2 \cdot 4053103$), and 950706376 (which is $2^3 \cdot 118838297$). The selection is made by the routine **RNOPT**. The choice of 16807 will result in the fastest execution time (see Gentle 1981), but Fishman and Moore's studies would seem to indicate that the performance of 950706376 is best among these three choices. If no selection is made explicitly, the routines use the multiplier 16807, which has been in use for some time (Lewis, Goodman, and Miller 1969). It is the "minimal standard generator" discussed by Park and Miller (1988).

The user can also select a shuffled version of the multiplicative congruential generators using **RNOPT**. The shuffled generators use a scheme due to Learmonth and Lewis (1973a). In this scheme, a table is filled with the first 128 uniform (0, 1) numbers resulting from the simple multiplicative congruential generator. Then, for each x_i from the simple generator, the low-order bits of x_i are used to select a random integer, j, from 1 to 128. The j-th entry in the table is then delivered as the random number; and x_i, after being scaled into the unit interval, is inserted into the j-th position in the table.

The Generalized Feedback Shift Register Generator

The GFSR generator uses the recursion $X_t = X_{t-1563} \oplus X_{t-96}$. This generator, which is different from earlier GFSR generators, was proposed by Fushimi (1990), who discusses the theory behind the generator and reports on several empirical tests of it. Background discussions on this type of generator can be found in Kennedy and Gentle (1980), pages 150-162.

The Mersenne Twister Generator

Both of the Mersenne Twister generators have a period of $2^{19937} - 1$ and a 624-dimensional equidistribution property. See Matsumoto et al. 1998 for details.

Setting the Seed

The seed of the generator can be set in **RNSET** and can be retrieved by **RNGET**. Prior to invoking any generator in this chapter , the user can call **RNSET** to initialize the seed, which is an integer variable taking a value between 1 and 2147483646. If it is not initialized by **RNSET**, a random seed is obtained from the system clock. Once it is initialized, the seed need not be set again. The seed is updated and passed from one routine to another by means of a named **COMMON** block, **R2NCOM**.

If the user wishes to restart a simulation, **RNGET** can be used to obtain the final seed value of one run to be used as the starting value in a subsequent run. Also, if two random number streams are desired in one run, **RNSET** and **RNGET** can be used before and after the invocations of the generators in each stream. If a shuffled generator or the GFSR generator is used, in addition to resetting the seed, the user must also reset some values in a

table. For the shuffled generators, this is done using the routines RNGES and RNSES and for the GFSR generator, the table is retrieved and set by the routines RNGEF and RNSEF. The tables for the shuffled generators are separate for single and double precision; so, if precisions are mixed in a program, it is necessary to manage each precision separately for the shuffled generators.

Timing Considerations

The generation of the uniform (0,1) numbers is done by the routine RNUN or by its function analog RNUNF. The particular generator selected in RNOPT, that is, the value of the multiplier and whether shuffling is done or whether the GFSR generator is used, affects the speed of RNUN and RNUNF. The smaller multiplier (16807, selected by IOPT = 1) is faster than the other multipliers. The multiplicative congruential generators that do not shuffle are faster than the ones that do. The GFSR generator is roughly as fast as the fastest multiplicative congruential generator, but the initialization for it (required only on the first invocation) takes longer than the generation of thousands of uniform random numbers. Precise statements of relative speeds depend on the computing system.

Whether RNUN or RNUNF is used also has an effect on the speed due to the overhead in invoking an external routine, or due to the program's inability to optimize computations by holding some operands in registers. This effect, of course, may be different in different environments. On an array processor or other computers with pipelined instructions, RNUN is likely to be considerably faster than RNUNF when several random numbers are to be generated at one time. In the case of array processors, the multiplicative congruential generators in RNUN are coded to generate subsequences in larger blocks (see Gentle 1990).

Use of Customized Uniform Generators

The basic uniform (0, 1) generators RNUN or RNUNF are used by all other routines in this chapter. If, for some reason, the user would prefer a different basic uniform generator, routines named "RNUN" and "RNUNF" can be written so that they include the named COMMON, through which the seed is passed, and that calls the user's custom generator. The named COMMON is

```
      COMMON /R2NCOM/ D2P31A, DSEED, D2P31R, DWK, DINTTB, INDCTR, &
            INTTB, WK, ICEED, IDSTFS, INTFS, ISRCFS, S2P31R, IWFS
      DOUBLE PRECISION D2P31A, D2P31R, DSEED, DWK(128)
      REAL    S2P31R, WK(128)
      INTEGER ICEED, IDSTFS, INDCTR, ISRCFS, IWFS(1563)
      LOGICAL DINTTB, INTTB, INTFS
      SAVE    /R2NCOM/
```

The user's "RNUN" and "RNUNF" can pass the seed through any of the variables, but the routines RNSET and RNGET expect the seed to be in ICEED. (The user should not expect to use any utility routines other than RNSET and RNGET if customized versions of RNUN or RNUNF are used.) The double precision versions of the nonuni-

form generators, such as **DRNBET** and **DRNGAM** (RNGAM), use the double precision versions of the uniform generators, **DRNUN** (RNUN) and **DRNUNF** (RNUNF), so to use the double precision nonuniform generators with customized uniform generators, the user would supply routines to replace **DRNUN** and **DRNUNF**.

Distributions Other than the Uniform

The nonuniform generators use a variety of transformation procedures. All of the transformations used are exact (mathematically). The most straightforward transformation is the *inverse CDF technique*, but it is often less efficient than others involving *acceptance/rejection* and *mixtures*. See Kennedy and Gentle (1980) for discussion of these and other techniques.

Many of the nonuniform generators in this chapter use different algorithms depending on the values of the parameters of the distributions. This is particularly true of the generators for discrete distributions. Schmeiser (1983) gives an overview of techniques for generating deviates from discrete distributions.

Although, as noted above, the uniform generators yield the same sequences on different computers, because of rounding, the nonuniform generators that use acceptance/rejection may occasionally produce different sequences on different computer/compiler environments.

Although the generators for nonuniform distributions use fast algorithms, if a very large number of deviates from a fixed distribution are to be generated, it might be worthwhile to consider a table sampling method, as implemented in the routines RNGDA, RNGDS, RNGDT, RNGCS, and RNGCT. After an initialization stage, which may take some time, the actual generation may proceed very fast.

Order Statistics and Antithetic Variates

For those generators, such as RNCHY and RNNOR, that use the inverse CDF technique, it is possible to generate any set of order statistics directly by use of a customized uniform generator, as discussed above, by generating order statistics in a custom "**RNUN**" or "**RNUNF**". In some routines that employ an inverse CDF technique, such as RNEXP and RNWIB, instead of directly using the uniform (0, 1) deviate u from RNUN, the uniform (0, 1) deviate $1 - u$ is used. In such routines the i-th order statistic from the uniform will yield the $(n + 1 - i)$-th order statistic from the nonuniform distribution.

A similar technique can be used to get antithetic variates. For each uniform deviate u, a second deviate $1 - u$ could be produced by a custom "**RNUN**" or "**RNUNF**". As with order statistics, this technique would only be reasonable for routines that use the inverse CDF technique.

Tests

Extensive empirical tests of some of the uniform random number generators available in **RNUN** and **RNUNF** are reported by Fishman and Moore (1982 and 1986). Results of tests on the generator using the multiplier 16807 with and without shuffling are reported by Learmonth and Lewis (1973b). If the user wishes to perform additional tests, the routines in Chapter 7, "Tests of Goodness of Fit and Randomness" may be of use. The user may also wish to compute some basic statistics or to make some plots of the output of the random number generator being used. The routines in Chapter 1, "Basic Statistics" and Chapter 16, "Line Printer Graphics" may be used for this purpose. Often in Monte Carlo applications, it is appropriate to construct an ad hoc test that is sensitive to departures that are important in the given application. For example, in using Monte Carlo methods to evaluate a one-dimensional integral, autocorrelations of order one may not be harmful, but they may be disastrous in evaluating a two-dimensional integral. Although generally the routines in this chapter for generating random deviates from nonuniform distributions use exact methods, and, hence, their quality depends almost solely on the quality of the underlying uniform generator, it is often advisable to employ an ad hoc test of goodness of fit for the transformations that are to be applied to the deviates from the nonuniform generator.

Copula Generators and Canonical Correlation

With release 7.0, three new subroutines associated with copulas have been added to the Fortran Numerical Library . A copula is a multivariate cumulative probability distribution (CDF) whose arguments are random variables uniformly distributed on the interval [0, 1] corresponding to the probabilities (variates) associated with arbitrarily distributed marginal deviates. The copula structure allows the multivariate CDF to be partitioned into the copula, which has associated with it information characterizing the dependence among the marginal variables, and the set of separate marginal deviates, each of which has its own distribution structure.

Two subroutines, **RNMVGC** and **RNMVTC**, allow the user to specify a correlation structure (in the form of a Cholesky matrix) which can be used to imprint correlation information on a sequence of multivariate random vectors. Each call to one of these methods returns a random vector whose elements (variates) are each uniformly distributed on the interval [0, 1] and correlated according to a user-specified Cholesky matrix. These variate vector sequences may then be inverted to marginal deviate sequences whose distributions and imprinted correlations are user-specified.

Method **RNMVGC** generates a random Gaussian copula vector by inverting a vector of uniform [0, 1] random numbers to a N(0, 1) deviate vector, imprinting the N(0,1) vector with the correlation information by multiplying it with the Cholesky matrix, and then using the N(0,1) CDF to map the Cholesky-imprinted deviate vector back to a vector of imprinted uniform [0, 1] variates.

Method **RNMVTC** inverts a vector of uniform [0, 1] random numbers to a N(0,1) deviate vector, imprints the vector with correlation information by multiplying it with the Cholesky matrix, transforms the imprinted N(0,1) vector to an imprinted Student's *t* vector (where each element is Student's *t* distributed with ν degrees of freedom) by

dividing each element of the imprinted N(0,1) vector by $\sqrt{s/\nu}$, where s is a random deviate taken from a chi-squared distribution with ν degrees of freedom, and finally maps the each element of the resulting imprinted Student's t vector back to a uniform [0, 1] distributed variate using the Student's t CDF.

The third copula subroutine, **CANCOR**, extracts a "canonical correlation" matrix from a sequence of multivariate deviate vectors whose component marginals are arbitrarily distributed. This is accomplished by first extracting the empirical CDF from each of the marginal deviate sequences and then using this empirical CDF to map the deviates to uniform [0, 1] variates which are then inverted to N(0, 1) deviates. Each element $C_{i\,j}$ of the canonical correlation matrix can then be extracted by averaging the products $z_{i\,t}\,z_{j\,t}$ of N(0, 1) deviates i and j over the t-indexed sequence. The utility of subroutine **CANCOR** is that because the canonical correlation matrix is derived from N(0, 1) deviates, the correlation is unbiased, i.e. undistorted by the arbitrary marginal distribution structures of the original deviate vector sequences. This is important in such financial applications as portfolio optimization, where correlation is used to estimate and minimize risk.

The use of subroutines **RNMVGC**, **RNMVTC**, and **CANCOR** is illustrated in the examples following subroutines **RNMVGC** and **RNMVTC**. The example following **RNMVGC** first uses method **RNMVGC** to create a correlation imprinted sequence of random deviate vectors and then uses method **CANCOR** to extract the correlation matrix from the imprinted sequence of vectors. Similarly, The example following **RNMVTC** first uses method **RNMVTC** to create a correlation imprinted sequence of random deviate vectors and then uses method **CANCOR** to extract the correlation matrix from the imprinted sequence of vectors.

Other Notes on Usage

The generators for continuous distributions are available in both single and double precision versions. This is merely for the convenience of the user; the double precision versions should not be considered more "accurate," except possibly for the multivariate distributions.

The names of all of the routines for random number generation begin with "**RN**" for single precision and "**DRN**" for double precision. In most routines, the first argument, **NR**, is the number of variates to generate; and the last variable, either **R** or **IR**, is the vector of random variates.

Error handling and workspace allocation in the routines for random number generation are done somewhat differently than in most other IMSL routines. In general, there is less error checking than in other routines since there is more emphasis on speed in the random number generation routines. Simple checks for gross errors are made in all routines; and the routines for setup do complete checking since it is assumed that they would not be called frequently. Some routines, such as those that construct tables or interpolate from tables, require that the user explicitly provide some work arrays.

Random Number Generation Utility Routines

All of the random number generators in this chapter depend on the generation of uniform (0, 1) numbers, which is done by the routine **RNUN**, or by its function analog **RNUNF**. These basic generators use either a multiplicative congruential method or a generalized feedback shift register (GFSR) method, or the Mersenne Twister method to yield a subsequence of a fixed cyclic sequence. The beginning of the subsequence is determined by the seed.

The utility routines for the random number generators allow the user to select the type of the generator (or to determine the type of the generator being used) and to set or retrieve the seed.

Selection of the Type of the Generator

The uniform pseudorandom number generators use a multiplicative congruential method, with or without shuffling or a GFSR method, or the Mersenne Twister method. Routine **RNOPT** determines which method is used; and in the case of a multiplicative congruential method, it determines the value of the multiplier and whether or not to use shuffling. The description of **RNUN** may provide some guidance in the choice of the form of the generator. If no selection is made explicitly, the generators use the multiplier 16807 without shuffling. This form of the generator has been in use for some time (see Lewis, Goodman, and Miller, 1969). This is the generator formerly known as **GGUBS** in the IMSL Library. It is the "minimal standard generator" discussed by Park and Miller (1988).

Both of the Mersenne Twister generators have a period of $2^{19937} - 1$ and a 624-dimensional equidistribution property. See Matsumoto et al. 1998 for details.

The IMSL Mersenne Twister generators are derived from code copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved. It is subject to the following notice:

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The IMSL 32-bit Mersenne Twister generator is based on the Matsumoto and Nishimura code 'mt19937ar' and the 64-bit code is based on 'mt19937-64'.

The selection of the type of generator is made by calling the routine **RNOPT**, choosing one of nine different options.

RNOPT

```
CALL RNOPT (IOPT)
```

The argument is:

IOPT — The indicator of the generator. (Input)

The random number generator is either a multiplicative congruential generator with modulus $2^{31} - 1$ or a GFSR generator or Mersenne Twister. `IOPT` is used to choose the multiplier and whether or not shuffling is done, or is used to choose the GFSR method, or is used to choose the Mersenne Twister generator.

IOPT	Generator
1	Congruential, with multiplier 16807 is used.
2	Congruential, with multiplier 16807 is used with shuffling.
3	Congruential, with multiplier 397204094 is used.
4	Congruential, with multiplier 397204094 is used with shuffling.
5	Congruential, with multiplier 950706376 is used.
6	Congruential, with multiplier 950706376 is used with shuffling.
7	GFSR, with the recursion $X_t = X_{t-1563} \oplus X_{t-96}$ is used.
8	A 32-bit Mersenne Twister generator is used. The real and double random numbers are generated from 32-bit integers.
9	A 64-bit Mersenne Twister generator is used. The real and double random numbers are generated from 64-bit integers. This ensures that all bits of both float and double are random.

If no selection is made explicitly, a multiplicative generator (with multiplier 16807) is used (equivalent to `IOPT` = 1).

The type of generator being used can be determined by calling the routine `RNOPG`.

RNOPG

```
CALL RNOPG (IOPT)
```

The argument is:

`IOPT`, which is an output variable in `RNOPG`.

Setting the Seed

Before using any of the random number generators, the generator must be initialized by selecting a *seed*, or starting value. The user does not have to do this, but it can done by calling the routine **RNSET**. If the user does not select a seed, one is generated using the system clock. A seed needs to be selected only once in a program unless there is some desire to maintain two separate streams of random numbers.

RNSET

```
CALL RNSET (ISEED)
```

The argument is:

ISEED — The seed of the random number generator. (Input)

ISEED must be in the range (0, 2147483646). If **ISEED** is zero (or if **RNSET** is not called before the generation of random numbers begins), a value is computed using the system clock; and, hence, the results of programs using the IMSL random number generators will be different at different times.

Stopping and Restarting Simulations and Controlling More Than One Stream of Random Numbers

For most purposes, even if several simulations are being run in the same program or if the simulation is being conducted in blocks, it is best to use the sequence of uniform random deviates just as produced by **RNUN** or **RNUNF** without concern for from where in the underlying cyclic sequence the numbers are coming.

If, however, the simulations are being conducted incrementally or if simulations are being run in parallel, it may be necessary to exercise more control over the sequence. The routines that are used in stopping and restarting simulations come in pairs, one to get the current value and one to set the value. The argument for each pair is the same within the pair; it is output in one case and input in the other. (**RNSET** is an exception since it is often used at the beginning of a simulation before any seed is ever set.) If a nonshuffled form of the multiplicative congruential generators is used (that is **IOPT** in **RNOPT** is 1, 3, or 5), the only thing that must be controlled is the seed, so in this case the only routines needed are

RNSET

Initializes the seed used in the generators

RNGET

Retrieves the current value of the seed

The usages are:

CALL RNSET (ISEED) (**ISEED** is input)

CALL RNGET (ISEED) (**ISEED** is output)

ISEED is an integer in the range 1 to 2147483646 (except, as noted above, it can be input to RNSET as 0 to indicate that the system clock is used to generate a seed).

If a shuffled generator, the GFSR generator, or a Mersenne Twister generator is used, in addition to controlling the seed as described above, another array must be maintained if the user wishes to stop and restart the simulation. It is a floating-point array for the shuffled generators and an integer array for the GFSR generator and Mersenne Twister generator. The routines are:

RNSES

Initializes the table used in the shuffled generators.

RNGES

Retrieves the current table used in the shuffled generators.

RNSEF

Initializes the table used in the GFSR generator.

RNGEF

Retrieves the current table used in the GFSR generator.

RNIN32

Initializes the table used in the 32-bit Mersenne Twister generator using an array.

RNSE32

Sets the current table used in the 32-bit Mersenne Twister generator.

RNGE32

Retrieves the current table used in the 32-bit Mersenne Twister generator.

RNIN64

Initializes the table used in the 64-bit Mersenne Twister generator using an array.

RNSE64

Sets the current table used in the 64-bit Mersenne Twister generator.

RNGE64

Retrieves the current table used in the 64-bit Mersenne Twister generator.

There are different tables used in the single and double precision versions of the shuffled generators, so RNGES and RNSES can also be used in double precision.

The usages are:

CALL RNGES (TABLE) (TABLE is output.)

`CALL RNSES (TABLE)` (`TABLE` is input.)

`CALL RNGEF (IARRAY)` (`IARRAY` is output.)

`CALL RNSEF (IARRAY)` (`IARRAY` is input.)

`CALL RNGE32 (MTABLE32)` (`MTABLE` is output.)

`CALL RNSE32 (MTABLE32)` (`MTABLE` is input.)

`CALL RNGE64 (MTABLE64)` (`MTABLE` is output.)

`CALL RNSE64 (MTABLE64)` (`MTABLE` is input.)

The arguments are:

TABLE — Array of length 128 used in the shuffled generators.

IARRAY — Array of length 1565 used in the GFSR generators.

MTABLE32 — Array of length 625 used in the 32-bit Mersenne Twister generators.

MTABLE64 — Array of length 313 used in the 64-bit Mersenne Twister generators.

The values in both `TABLE` and `IARRAY` are initialized by the IMSL random number generators. The values are all positive in both arrays except if the user wishes to reinitialize the array, in which case the first element of the array is input as a nonpositive value. (Usually, one should avoid reinitializing these arrays, but it might be necessary sometimes in restarting a simulation.) If the first element of `TABLE` or `IARRAY` is set to a nonpositive value on the call to `RNSES` or `RNSEF`, on the next invocation of a routine to generate random numbers using shuffling (if `RNSES`) or a GFSR method (if `RNSEF`), the appropriate array will be reinitialized.

In addition to controlling separate streams of random numbers, sometimes it is desirable to insure from the beginning that two streams do not overlap. This can be done with the congruential generators that do not do shuffling by using `RNISD` to get a seed that will generate random numbers beginning 100,000 numbers farther along.

RNISD

```
CALL RNISD (ISEED1, ISEED2)
```

The arguments are:

ISEED1 — The seed that yields the first stream. (Input)

ISEED2 — The seed that yields a stream beginning 100,000 numbers beyond the stream that begins with `ISEED1`. (Output)

Given a seed, **ISEED1**, **RNISD** determines another seed, **ISEED2**, such that if one of the IMSL multiplicative congruential generators, using no shuffling, went through 100,000 generations starting with **ISEED1**, the next number in that sequence would be the first number in the sequence that begins with the seed **ISEED2**. This can be described more simply by stating that **RN1** and **RN2** in the following sequence of FORTRAN are assigned the same values.

```
          CALL RNISD(ISEED1, ISEED2)
          CALL RNSET(ISEED1)
          DO 10 I = 1, 100000
             RN1 = RNUNF()
       10 CONTINUE
          RN1 = RNUNF()
          CALL RNSET(ISEED2)
          RN2 = RNUNF()
```

To obtain seeds that generate sequences with beginning values separated by 200,000 numbers, call **RNISD** twice:

```
    CALL RNISD(ISEED1, ISEED2)
    CALL RNISD(ISEED2, ISEED2)
```

Note that **RNISD** works only when a multiplicative congruential generator without shuffling is used. This means that either the routine **RNOPT** has not been called at all or that it has been last called with **IOPT** taking a value of 1, 3, or 5.

For many of the IMSL generators for nonuniform distributions that do not use the inverse CDF method, the distance between the sequences generated starting with **ISEED1** and starting with **ISEED2** may be less than 100,000. This is because the nonuniform generators that use other techniques may require more than one uniform deviate for each output deviate.

The reason that one may want two seeds that generate sequences a known distance apart is for blocking Monte Carlo experiments or for running parallel streams.

Examples

Example 1: Selecting the Type of Generator and Stopping and Restarting the Simulations

In this example, three separate simulation streams are used, each with a different form of the generator. Each stream is stopped and restarted. (Although this example is obviously an artificial one, there may be reasons for maintaining separate streams and stopping and restarting them because of the nature of the usage of the random numbers coming from the separate streams.)

```
         USE IMSL_LIBRARIES

         IMPLICIT   NONE
         INTEGER    I, IARRAY(1565), ISEED1, ISEED2, ISEED7, NOUT, NR
         REAL       R(5), TABLE(128)
```

```
!
      CALL UMACH (2, NOUT)
      NR     = 5
      ISEED1 = 123457
      ISEED2 = 123457
      ISEED7 = 123457
!                                Begin first stream, IOPT = 1 (by
!                                default)
      CALL RNSET (ISEED1)
      CALL RNUN (R)
      CALL RNGET (ISEED1)
      WRITE (NOUT,99997) (R(I),I=1,NR), ISEED1
!                                Begin second stream, IOPT = 2
      CALL RNOPT (2)
      CALL RNSET (ISEED2)
      CALL RNUN (R)
      CALL RNGET (ISEED2)
      CALL RNGES (TABLE)
      WRITE (NOUT,99998) (R(I),I=1,NR), ISEED2
!                                Begin third stream, IOPT = 7
      CALL RNOPT (7)
      CALL RNSET (ISEED7)
      CALL RNUN (R)
      CALL RNGET (ISEED7)
      CALL RNGEF (IARRAY)
      WRITE (NOUT,99999) (R(I),I=1,NR), ISEED7
!                                Reinitialize seed
!                                Resume first stream
      CALL RNOPT (1)
      CALL RNSET (ISEED1)
      CALL RNUN (R)
      CALL RNGET (ISEED1)
      WRITE (NOUT,99997) (R(I),I=1,NR), ISEED1
!                                Reinitialize seed and table for
!                                shuffling
!                                Resume second stream
      CALL RNOPT (2)
      CALL RNSET (ISEED2)
      CALL RNSES (TABLE)
      CALL RNUN (R)
      CALL RNGET (ISEED2)
      WRITE (NOUT,99998) (R(I),I=1,NR), ISEED2
!                                Reinitialize seed and table for GFSR
!                                Resume third stream
      CALL RNOPT (7)
      CALL RNSET (ISEED7)
      CALL RNSEF (IARRAY)
      CALL RNUN (R)
      CALL RNGET (ISEED7)
      WRITE (NOUT,99999) (R(I),I=1,NR), ISEED7
!
99997 FORMAT (/, '  First stream  ', 5F8.4, /, '  Output seed = ', &
             I11)
99998 FORMAT (/, '  Second stream ', 5F8.4, /, '  Output seed = ', &
             I11)
99999 FORMAT (/, '  Third stream  ', 5F8.4, /, '  Output seed = ', &
             I11)
!
      END
```

Output

```
 First stream    0.9662  0.2607  0.7663  0.5693  0.8448
 Output seed =   1814256879
 Second stream   0.7095  0.1861  0.4794  0.6038  0.3790
 Output seed =   1965912801
 Third stream    0.3914  0.0263  0.7622  0.0281  0.8997
 Output seed =   1932158269
 First stream    0.0443  0.9872  0.6014  0.8964  0.3809
 Output seed =   817878095
 Second stream   0.2557  0.4788  0.2258  0.3455  0.5811
 Output seed =   2108806573
 Third stream    0.7519  0.5084  0.9070  0.0910  0.6917
 Output seed =   1485334679
```

Example 2: Determining Seeds for Separate Streams

In this example, **RNISD** is used to determine seeds for 4 separate streams, each 200,000 numbers apart, for a multiplicative congruential generator without shuffling. (Since **RNOPT** is not invoked to select a generator, the multiplier is 16807.) To get each seed requires two invocations of **RNISD**. All of the streams are non-overlapping, since the period of the underlying generator is 2,147,483,646.

```
      USE UMACH_INT
      USE RNISD_INT

      IMPLICIT   NONE
      INTEGER    ISEED1, ISEED2, ISEED3, ISEED4, NOUT
!
      CALL UMACH (2, NOUT)
      ISEED1 = 123457
      CALL RNISD (ISEED1, ISEED2)
      CALL RNISD (ISEED2, ISEED2)
      CALL RNISD (ISEED2, ISEED3)
      CALL RNISD (ISEED3, ISEED3)
      CALL RNISD (ISEED3, ISEED4)
      CALL RNISD (ISEED4, ISEED4)
      WRITE (NOUT,99999) ISEED1, ISEED2, ISEED3, ISEED4
!
99999 FORMAT ('  Seeds for four separate streams: ', /, '  ', 4I11)
!
      END
```

Output

```
 Seeds for four separate streams:
      123457 2016130173   85016329  979156171
```

RNIN32

Initializes the 32-bit Mersenne Twister generator using an array.

Required Arguments

KEY— Integer array of length `LEN` used to initialize the 32-bit Mersenne Twister generator. (Input)

Optional Arguments

LEN — Length of the array key. (Input)

FORTRAN 90 Interface

Generic: `CALL RNIN32 (KEY [, ...])`

Specific: The specific interface name is `S_RNIN32`.

FORTRAN 77 Interface

Single: `CALL RNIN32 (KEY,LEN)`

Description

By default, the Mersenne Twister random number generator is initialized using the current seed value (see `RNGET`). The seed is limited to one integer for initialization. This function allows an arbitrary length array to be used for initialization. This subroutine completely replaces the use of the seed for initialization of the 32-bit Mersenne Twister generator.

Example

See routine `RNGE32`.

RNGE32

Retrieves the current table used in the 32-bit Mersenne Twister generator.

Required Arguments

MTABLE — Integer array of length 625 containing the table used in the 32-bit Mersenne Twister generator. (Output)

FORTRAN 90 Interface

Generic: CALL RNGE32 (MTABLE)

Specific: The specific interface name is RNGE32

FORTRAN 77 Interface

Single: CALL RNGE32 (MTABLE)

Description

The values in the table contain the state of the 32-bit Mersenne Twister random number generator. The table can be used by RNSE32 to set the generator back to this state.

Example

In this example, four simulation streams are generated. The first series is generated with the seed used for initialization. The second series is generated using an array for initialization. The third series is obtained by resetting the generator back to the state it had at the beginning of the second stream. Therefore, the second and third streams are identical. The fourth stream is obtained by resetting the generator back to its original, uninitialized state, and having it reinitialize using the seed. The first and fourth streams are therefore the same.

```
      USE RNIN32_INT
      USE RNGE32_INT
      USE RNSET_INT
      USE UMACH_INT
      USE RNUN_INT
      IMPLICIT   NONE
      INTEGER    I, ISEED, NOUT
      INTEGER INIT(4)
      DATA INIT/291,564,837,1110/
```

```
      DATA ISEED/123457/
      INTEGER NR
      REAL R(5)
      INTEGER MTABLE(625)
      CHARACTER CLABEL(5)*5, FMT*8, RLABEL(3)*5
      RLABEL(1)='NONE'
      CLABEL(1)='NONE'
      DATA FMT/'(W10.4)'/
      NR=5
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNOPT(8)
      CALL RNSET(ISEED)
      CALL RNUN(R)
      CALL WRRRL('FIRST STREAM OUTPUT',1,5,R,1,0, &
                 FMT, RLABEL, CLABEL)
!     REINITIALIZE MERSENNE TWISTER SERIES WITH AN ARRAY
      CALL RNIN32(INIT)
!     SAVE THE STATE OF THE SERIES
      CALL RNGE32(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('SECOND STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
!     RESTORE THE STATE OF THE TABLE
      CALL RNSE32(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('THIRD STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
!     RESET THE SERIES - IT WILL REINITIALIZE FROM THE SEED
      MTABLE(1)=1000
      CALL RNSE32(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('FOURTH STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
      END
```

Output

```
                    First stream output
      0.4347      0.3522      0.0139      0.2091      0.4956
                   Second stream output
      0.2486      0.2226      0.1111      0.9563      0.9846
                    Third stream output
      0.2486      0.2226      0.1111      0.9563      0.9846
                   Fourth stream output
      0.4347      0.3522      0.0139      0.2091      0.4956
```

RNSE32

Sets the current table used in the 32-bit Mersenne Twister generator.

Required Arguments

MTABLE — Integer array of length 625 containing the table used in the 32-bit Mersenne Twister generator. (Input)

FORTRAN 90 Interface

Generic: `CALL RNSE32 (MTABLE)`

Specific: The specific interface name is `RNSE32`

FORTRAN 77 Interface

Single: `CALL RNSE32 (MTABLE)`

Description

The values in `MTABLE` are the state of the 32-bit Mersenne Twister random number generator obtained by a call to `RNGE32`. The values in the table can be used to restore the state of the generator.

Alternatively, if `MTABLE` [1] > 625 then the generator is set to its original, uninitialized, state.

Example

See routine `RNGE32`.

RNIN64

Initializes the 64-bit Mersenne Twister generator using an array.

Required Arguments

KEY— Integer (kind=8) array of length **LEN** used to initialize the 64-bit Mersenne Twister generator. (Input)

Optional Arguments

LEN — Length of the array key. (Input)

FORTRAN 90 Interface

Generic: **CALL RNIN64** (**KEY** [, ...])

Specific: The specific interface name is **S_RNIN64**.

FORTRAN 77 Interface

Single: **CALL RNIN64** (**KEY**, **LEN**)

Description

By default, the Mersenne Twister random number generator is initialized using the current seed value (see **RNGET**). The seed is limited to one integer for initialization. This function allows an arbitrary length array to be used for initialization. This subroutine completely replaces the use of the seed for initialization of the 64-bit Mersenne Twister generator.

Example

See routine **RNGE64**.

RNGE64

Retrieves the current table used in the 64-bit Mersenne Twister generator.

Required Arguments

MTABLE — Integer (kind=8) array of length 313 containing the table used in the 64-bit Mersenne Twister generator. (Output)

FORTRAN 90 Interface

Generic: CALL RNGE64 (MTABLE)

Specific: The specific interface name is RNGE64

FORTRAN 77 Interface

Single: CALL RNGE64 (MTABLE)

Description

The values in the table contain the state of the 64-bit Mersenne Twister random number generator. The table can be used by RNSE64 to set the generator back to this state.

Example

In this example, four simulation streams are generated. The first series is generated with the seed used for initialization. The second series is generated using an array for initialization. The third series is obtained by resetting the generator back to the state it had at the beginning of the second stream. Therefore, the second and third streams are identical. The fourth stream is obtained by resetting the generator back to its original, uninitialized state, and having it reinitialize using the seed. The first and fourth streams are therefore the same.

```
      USE RNIN64_INT
      USE RNGE64_INT
      USE RNSET_INT
      USE UMACH_INT
      USE RNUN_INT
      IMPLICIT   NONE
      INTEGER    I, ISEED, NOUT
      INTEGER(KIND=8) INIT(4)
      DATA INIT/291,564,837,1110/
```

```
      DATA ISEED/123457/
      INTEGER NR
      REAL R(5)
      INTEGER(KIND=8) MTABLE(313)
      CHARACTER CLABEL(5)*5, FMT*8, RLABEL(3)*5
      RLABEL(1)='NONE'
      CLABEL(1)='NONE'
      DATA FMT/'(W10.4)'/
      NR=5
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNOPT(9)
      CALL RNSET(ISEED)
      CALL RNUN(R)
      CALL WRRRL('FIRST STREAM OUTPUT',1,5,R,1,0, &
                 FMT, RLABEL, CLABEL)
!     REINITIALIZE MERSENNE TWISTER SERIES WITH AN ARRAY
      CALL RNIN64(INIT)
!     SAVE THE STATE OF THE SERIES
      CALL RNGE64(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('SECOND STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
!     RESTORE THE STATE OF THE TABLE
      CALL RNSE64(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('THIRD STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
!     RESET THE SERIES - IT WILL REINITIALIZE FROM THE SEED
      MTABLE(1)=1000
      CALL RNSE64(MTABLE)
      CALL RNUN(R)
      CALL WRRRL('FOURTH STREAM OUTPUT',1,5,R,1,0, &
                  FMT, RLABEL, CLABEL)
      END
```

Output

```
                    First stream output
      0.5799      0.9401      0.7102      0.1640      0.5457
                   Second stream output
      0.4894      0.7397      0.5725      0.0863      0.7588
                    Third stream output
      0.4894      0.7397      0.5725      0.0863      0.7588
                   Fourth stream output
      0.5799      0.9401      0.7102      0.1640      0.5457
```

RNSE64

Sets the current table used in the 64-bit Mersenne Twister generator.

Required Arguments

MTABLE — Integer (kind=8) array of length 313 containing the table used in the 64-bit Mersenne Twister generator. (Input)

FORTRAN 90 Interface

Generic: `CALL RNSE64 (MTABLE)`

Specific: The specific interface name is `RNSE64`

FORTRAN 77 Interface

Single: `CALL RNSE64 (MTABLE)`

Description

The values in `MTABLE` are the state of the 64-bit Mersenne Twister random number generator obtained by a call to `RNGE64`. The values in the table can be used to restore the state of the generator. Alternatively, if `MTABLE[1]` > 313 then the generator is set to its original, uninitialized, state.

Example

See routine `RNGE64`.

RNUN

Generates pseudorandom numbers from a uniform (0, 1) distribution.

Required Arguments

R — Vector of length **NR** containing the random uniform (0, 1) deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNUN** (**R** [, ...])
Specific: The specific interface names are **S_RNUN** and **D_RNUN**.

FORTRAN 77 Interface

Single: **CALL RNUN** (**NR**, **R**)
Double: The double precision name is **DRNUN**.

Description

Routine **RNUN** generates pseudorandom numbers from a uniform (0,1) distribution using either a multiplicative congruential method or a generalized feedback shift register (GFSR) method, or the Mersenne Twister generator. The form of the multiplicative congruential generator is

$$x_i \equiv cx_{i-1} \mathrm{mod}\left(2^{31} - 1\right)$$

Each x_i is then scaled into the unit interval (0,1). The possible values for c in the IMSL generators are 16807, 397204094, and 950706376. The selection is made by the routine **RNOPT**. The choice of 16807 will result in the fastest execution time. If no selection is made explicitly, the routines use the multiplier 16807.

The user can also select a shuffled version of the multiplicative congruential generators. In this scheme, a table is filled with the first 128 uniform (0,1) numbers resulting from the simple multiplicative congruential generator. Then, for each x_i from the simple generator, the low-order bits of x_i are used to select a random integer, j, from 1 to 128. The j-th entry in the table is then delivered as the random number; and x_i, after being scaled into the unit interval, is inserted into the j-th position in the table.

The GFSR method is based on the recursion $X_t = X_{t-1563} \oplus X_{t-96}$. This generator, which is different from earlier GFSR generators, was proposed by Fushimi (1990), who discusses the theory behind the generator and reports on several empirical tests of it.

Mersenne Twister (MT) is a pseudorandom number generating algorithm developed by Makoto Matsumoto and Takuji Nishimura in 1996-1997. MT has far longer period and far higher order of equidistribution than any other implemented generators. The values returned in **R** by **RNUN** are positive and less than 1.0. Values in **R** may be smaller than the smallest relative spacing, however. Hence, it may be the case that some value **R**(*i*) is such that 1.0 **-** **R**(*i*) = 1.0.

Deviates from the distribution with uniform density over the interval (**A**, **B**) can be obtained by scaling the output from **RNUN**. The following statements (in single precision) would yield random deviates from a uniform (**A**, **B**) distribution:

```
          CALL RNUN (NR, R)
          CALL SSCAL (NR, B-A, R, 1)
          CALL SADD (NR, A, R, 1)
```

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNUN** is used to generate five pseudorandom uniform numbers. Since **RNOPT** is not called, the generator used is a simple multiplicative congruential one with a multiplier of 16807.

```
          USE RNUN_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE

          INTEGER    ISEED, NOUT, NR
          REAL       R(5)
    !
          CALL UMACH (2, NOUT)
          NR    = 5
```

```
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNUN (R)
      WRITE (NOUT,99999) R
99999 FORMAT ('      Uniform random deviates: ', 5F8.4)
      END
```

Output

```
    Uniform random deviates:    .9662   .2607   .7663   .5693   .8448
```

RNUNF

This function generates a pseudorandom number from a uniform (0, 1) distribution.

Function Return Value

RNUNF — Function value, a random uniform (0, 1) deviate. (Output)
See Comment 1.

Required Arguments

None.

FORTRAN 90 Interface

Generic: **RNUNF ( )**
Specific: The specific interface names are **S_RNUNF** and **D_RNUNF**.

FORTRAN 77 Interface

Single: **RNUNF ( )**
Double: The double precision name is **DRNUNF**.

Description

Routine **RNUNF** is the function form of **RNUN**. The routine **RNUNF** generates pseudorandom numbers from a uniform (0, 1) distribution. The algorithm used is determined by **RNOPT**. The values returned by **RNUNF** are positive and less than 1.0.

If several uniform deviates are needed, it may be more efficient to obtain them all at once by a call to **RNUN** rather than by several references to **RNUNF**.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

```
X = RNUNF()
Y = SQRT(X)
```

must be used rather than

```
Y = SQRT(RNUNF())
```

If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.
3. This function has a side effect: it changes the value of the seed, which is passed through a common block.

Example

In this example, RNUNF is used to generate five pseudorandom uniform numbers. Since RNOPT is not called, the generator used is a simple multiplicative congruential one with a multiplier of 16807.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNUNF_INT

      IMPLICIT   NONE
      INTEGER    I, ISEED, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNSET (ISEED)
      DO 10  I=1, 5
         R(I) = RNUNF()
   10 CONTINUE
      WRITE (NOUT,99999) R
99999 FORMAT ('      Uniform random deviates: ', 5F8.4)
      END
```

Output

```
 Uniform random deviates:   0.9662  0.2607  0.7663  0.5693  0.8448
```

RNBIN

Generates pseudorandom numbers from a binomial distribution.

Required Arguments

N — Number of Bernoulli trials. (Input)

P — Probability of success on each trial. (Input)
P must be greater than 0.0 and less than 1.0.

IR — Vector of length **NR** containing the random binomial deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**IR**,1).

FORTRAN 90 Interface

Generic: `CALL RNBIN (N, P, IR [, ...])`
Specific: The specific interface name is `S_RNBIN`.

FORTRAN 77 Interface

Single: `CALL RNBIN (NR, N, P, IR)`

Description

Routine **RNBIN** generates pseudorandom numbers from a binomial distribution with parameters **N** and **P**. **N** and **P** must be positive, and **P** must be less than 1. The probability function (with n = **N** and p = **P**) is

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for x = 0, 1, 2, ..., n.

The algorithm used depends on the values of n and p. If $np < 10$ or if p is less than a machine epsilon (**AMACH**(4) (Reference Material)), the inverse CDF technique is used; otherwise, the BTPE algorithm of Kachitvichyanukul and Schmeiser (see Kachitvichyanukul 1982) is used. This is an acceptance/rejection method using a composition of four regions. (TPE = Triangle, Parallelogram, Exponential, left and right.)

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNBIN** is used to generate five pseudorandom binomial variates with parameters 20 and 0.5.

```
      USE RNBIN_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, N, NOUT
      REAL       P
!
      CALL UMACH (2, NOUT)
      N     = 20
      P     = 0.5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNBIN (N, P, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Binomial (20, 0.5) random deviates: ', 5I4)
      END
```

Output

```
 Binomial (20, 0.5) random deviates:   14   9  12  10  12
```

RNGDA

Generates pseudorandom numbers from a general discrete distribution using an alias method.

Required Arguments

IOPT — Indicator of whether the alias vectors are to be initialized. (Input)

IOPT	Action
0	The alias vectors are to be initialized using the probabilities in PROBS. IOPT is set to 0 on the first call to RNGDA.
1	The alias vectors IWK and WK are used but PROBS is not used.

IMIN — Smallest value the random deviate can assume. (Input)
This is the value corresponding to the probability in PROBS(1).

PROBS — Vector of length NMASS containing probabilities associated with the individual mass points. (Input)
The elements of PROBS must be nonnegative and must sum to 1.0.

IWK — Index vector of length NMASS. (Input, if IOPT = 1; Output, if IOPT = 0) IWK is a work vector.

WK — Index vector of length NMASS. (Input, if IOPT = 1; Output, if IOPT = 0) WK is a work vector.

IR — Vector of length NR containing the random discrete deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

NMASS — Number of mass points in the discrete distribution. (Input)
Default: NMASS = size (PROBS,1).

FORTRAN 90 Interface

Generic: CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR [, ...])
Specific: The specific interface names are S_RNGDA and D_RNGDA.

FORTRAN 77 Interface

Single: CALL RNGDA (NR, IOPT, IMIN, NMASS, PROBS, IWK, WK, IR)
Double: The double precision name is DRNGDA.

Description

Routine **RNGDA** generates pseudorandom numbers from a discrete distribution with probability function given in the vector **PROBS**; that is

$$\Pr(X = i) = p_j$$

for

$$i = i_{\min}, i_{\min} + 1, \ \ldots, \ i_{\min} + n_m - 1 \quad \text{where } j = i - i_{min} + 1, \ p_j = \texttt{PROBS}(j),$$
$$i_{\min} = \texttt{IMIN}, \text{ and } n_m = \texttt{NMASS}.$$

The algorithm is the alias method, due to Walker (1974), with modifications suggested by Kronmal and Peterson (1979). The method involves a setup phase, in which the vectors **IWK** and **WK** are filled. After the vectors are filled, the generation phase is very fast.

Comments

1. In the interest of efficiency, this routine does only limited error checking when **IOPT** = 1.
2. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Examples

Example 1

In this example, **RNGDA** is used to generate five pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

When **RNGDA** is called the first time, **IOPT** is input as 0. This causes the work arrays to be initialized. In the next call, **IOPT** is 1, so the setup phase is bypassed.

```
      USE RNGDA_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NMASS, NR
      PARAMETER  (NMASS=5, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, IWK(NMASS), NOUT
      REAL       PROBS(NMASS), WK(NMASS)
!
      CALL UMACH (2, NOUT)
      IMIN     = 1
      PROBS(1) = 0.05
      PROBS(2) = 0.45
      PROBS(3) = 0.31
      PROBS(4) = 0.04
      PROBS(5) = 0.15
      IOPT     = 0
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)
      WRITE (NOUT,99998) IR
99998 FORMAT ('          Random deviates: ', 5I4)
      IOPT = 1
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT ('                           ', 5I4)
      END
```

Output

```
 Random deviates:    3   2   2   3   5
                     1   3   4   5   3
```

Example 2

In this example, **RNGDA** is used to generate five pseudorandom binomial variates with parameters 20 and 0.5.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNGDA_INT
      USE BINPR_INT

      IMPLICIT   NONE
      INTEGER    NMASS, NR
      PARAMETER  (NMASS=21, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, IWK(NMASS), K, N, NOUT
      REAL       P, PROBS(NMASS), WK(NMASS)
!
      CALL UMACH (2, NOUT)
      N    = 20
```

```
      P    = 0.5
      IMIN = 0
      DO 10  K=1, NMASS
         PROBS(K) = BINPR(K-1,N,P)
   10 CONTINUE
      IOPT  = 0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNGDA (IOPT, IMIN, PROBS, IWK, WK, IR)

      WRITE (NOUT,99999) IR
99999 FORMAT ('   Binomial (20, .5) deviates: ', 5I4)
      END
```

Output

```
   Binomial (20, .5) deviates:   12  10  16  12  11
```

RNGDS

Sets up table to generate pseudorandom numbers from a general discrete distribution.

Required Arguments

PRF — User-supplied **FUNCTION** to compute the probability associated with each mass point of the distribution. The form is **PRF**(**IX**), where

IX – Point at which the probability function is to be evaluated. (Input)
IX can range from **IMIN** to the value at which the cumulative probability is greater than or equal to 1.0 – **DEL**.

PRF – Value of the probability function at **IX**. (Output)
PRF must be declared **EXTERNAL** in the calling program.

DEL — Maximum absolute error allowed in computing the cumulative probability. (Input)
Probabilities smaller than **DEL** are ignored; hence, **DEL** should be a small positive number. If **DEL** is too small, however, **CUMPR**(**NMASS**) must be exactly 1.0 since that value is compared to 1.0 **- DEL**.

NNDX — The number of elements of **CUMPR** available to be used as indexes. (Input)
NNDX must be greater than or equal to 1. In general, the larger **NNDX** is, to within sixty or seventy percent of **NMASS**, the more efficient the generation of random numbers using **RNGDS** will be.

IMIN — Smallest value the random deviate can assume. (Input/Output)
IMIN is not used if **IOPT** = 1. If **IOPT** = 0, **PRF** is evaluated at **IMIN**. If this value is less than **DEL**, **IMIN** is incremented by 1 and again **PRF** is evaluated at **IMIN**. This process is continued until **PRF**(**IMIN**) ≥ **DEL**. **IMIN** is output as this value and **CUMPR**(1) is output as **PRF**(**IMIN**).

NMASS — The number of mass points in the distribution. (Input, if **IOPT** = 1; Output, if **IOPT** = 0)
If **IOPT** = 0, **NMASS** is the smallest integer such that **PRF**(**IMIN** + **NMASS -** 1) > 1.0 **- DEL**. **NMASS** does include the points **IMIN**(in) + *j* for which **PRF**(**IMIN**(in) + *j*) < **DEL**, for *j* = 0, 1, ..., **IMIN**(out) **- IMIN**(in), where **IMIN**(in) denotes the input value of **IMIN** and **IMIN**(out) denotes its output value.

CUMPR — Vector of length **NMASS** + **NNDX** containing in the first **NMASS** positions, the cumulative probabilities and in some of the remaining positions, indexes to speed access to the probabilities. (Output, if **IOPT** = 0; Input/Output, otherwise)
CUMPR(**NMASS** + 1) + 1 is the actual number of index positions used.

Optional Arguments

IOPT — Indicator of the extent to which **CUMPR** is initialized prior to calling **RNGDS**. (Input)
Default: **IOPT** = 0.

IOPT	Action
0	RNGDS fills all of CUMPR, using PRF.
1	RNGDS fills only the index portion of CUMPR, using the values in the first NMASS positions. PRF is not used and may be a dummy function; also, IMIN and DEL are not used.

LCUMPR — Dimension of **CUMPR** exactly as specified in the dimension statement in the calling program. (Input)
Since the logical length of **CUMPR** is determined in **RNGDS**, **LCUMPR** is used for error checking.
Default : **LCUMPR** = size (**CUMPR**,1).

FORTRAN 90 Interface

Generic: **CALL RNGDS** (**PRF**, **DEL**, **NNDX**, **IMIN**, **NMASS**, **CUMPR** [, ...])
Specific: The specific interface names are **S_RNGDS** and **D_RNGDS**.

FORTRAN 77 Interface

Single: **CALL RNGDS** (**PRF**, **IOPT**, **DEL**, **NNDX**, **IMIN**, **NMASS**, **CUMPR**, **LCUMPR**)
Double: The double precision name is **DRNGDS**.

Description

Routine **RNGDS** sets up a table that routine **RNGDT** uses to generate pseudorandom deviates from a discrete distribution. The distribution can be specified either by its probability function **PRF** or by a vector of values of the cumulative probability function. Note that **PRF** is *not* the cumulative probability distribution function. If the cumulative probabilities are already available in **CUMPR**, the only reason to call **RNGDS** is to form an index vector in the upper portion of **CUMPR** so as to speed up the generation of random deviates by the routine **RNGDT**.

Comments

1. Informational error

Type	Code	Description
3	1	For some `I`, `CUMPR(I)` is computed to be less than 1.0 - `DEL`, and yet `CUMPR(I + 1)` - 1.0 is greater than 1.0 - `CUMPR(I + 1)`. In this case, the maximum value that the random variable is allowed to take on is `I`; that is, `CUMPR(I)` is set to 1.0.

2. The routine `RNGDT` uses the table set up by **RNGDS** to generate random numbers from the distribution with **CDF** represented in **CUMPR**.

Examples

Example 1

In this example, **RNGDS** is used to set up a table to generate pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

In this simple example, we input the cumulative probabilities directly in **CUMPR** and request 3 indexes to be computed (**NNDX** = 4). Since the number of mass points is so small, the indexes would not have much effect on the speed of the generation of the random variates.

```
      USE RNGDS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LCUMPR
      PARAMETER  (LCUMPR=9)
!
      INTEGER    IMIN, IOPT, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      NMASS    = 5
      CUMPR(1) = 0.05
      CUMPR(2) = 0.50
      CUMPR(3) = 0.81
      CUMPR(4) = 0.85
      CUMPR(5) = 1.00
      IOPT     = 1
      NNDX     = 4
      DEL      = 0.00001
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR, IOPT=IOPT)
      WRITE (NOUT,99999) CUMPR
99999 FORMAT ('   Cumulative probabilities and indexes: ', /, 9F6.2)
      END
!
!                                  Dummy function
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      PRF = 0.0
      RETURN
      END
```

Output

```
   Cumulative probabilities and indexes:
   0.05  0.50  0.81  0.85  1.00  3.00  1.00  2.00  5.00
```

Example 2

This example, **RNGDS** is used to set up a table to generate binomial variates with parameters 20 and 0.5. The routine **BINPR** (see Chapter 17, "Probability Distribution Functions and Inverses" is used to compute the probabilities.

```
      USE RNGDS_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    LCUMPR
      PARAMETER  (LCUMPR=33)
!
      INTEGER    I, IMIN, N, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, P, PRF
      COMMON     /BINCOM/ N, P
```

```
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      N    = 20
      P    = 0.5
      IMIN = 0
      NNDX = 12
      DEL  = 0.00001
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR)
      WRITE (NOUT,99998) IMIN, NMASS
99998 FORMAT (' The smallest point with positive probability using ', &
             /, ' the given DEL is ', I1, ' and all points after ', /, &
              ' point number ', I2, ' (counting from the input value ', &
             /, ' of IMIN) have zero probability.')
      WRITE (NOUT,99999) (CUMPR(I),I=1,NMASS+NNDX)
99999 FORMAT (' Cumulative probabilities and indexes: ', /, (5X,8F8.4))
      END
!
!                                  Compute binomial probabilities
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      INTEGER    N
      REAL       BINPR, P
      COMMON     /BINCOM/ N, P
      EXTERNAL   BINPR
!
      PRF = BINPR(IX,N,P)
      RETURN
      END
```

Output

```
 The smallest point with positive probability using
 the given DEL is 1 and all points after
 point number 19 (counting from the input value
 of IMIN) have zero probability.
 Cumulative probabilities and indexes:
      0.0000  0.0002  0.0013  0.0059  0.0207  0.0577  0.1316  0.2517
      0.4119  0.5881  0.7483  0.8684  0.9423  0.9793  0.9941  0.9987
      0.9998  1.0000  1.0000 11.0000  1.0000  7.0000  8.0000  9.0000
      9.0000 10.0000 11.0000 11.0000 12.0000 13.0000 19.0000
```

RNGDT

Generates pseudorandom numbers from a general discrete distribution using a table lookup method.

Required Arguments

IMIN — Smallest value the random deviate can assume. (Input)
This is the value corresponding to the probability in **CUMPR**(1).

NMASS — Number of mass points in the discrete distribution. (Input)

CUMPR — Vector of length at least **NMASS** + 1 containing in the first **NMASS** positions the cumulative probabilities and, possibly, indexes to speed access to the probabilities. (Input)
IMSL routine **RNGDS** can be used to initialize **CUMPR** properly. If no elements of **CUMPR** are used as indexes, **CUMPR**(**NMASS** + 1) is 0.0 on input. The value in **CUMPR**(1) is the probability of **IMIN**. The value in **CUMPR**(**NMASS**) must be exactly 1.0 (since this is the **CDF** at the upper range of the distribution).

IR — Vector of length **NR** containing the random discrete deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**IR**,1).

FORTRAN 90 Interface

Generic: **CALL RNGDT** (**IMIN**, **NMASS**, **CUMPR**, **IR** [, ...])
Specific: The specific interface names are **S_RNGDT** and **D_RNGDT**.

FORTRAN 77 Interface

Single: **CALL RNGDT** (**NR**, **IMIN**, **NMASS**, **CUMPR**, **IR**)
Double: The double precision name is **DRNGDT**.

Description

Routine **RNGDT** generates pseudorandom deviates from a discrete distribution, using the table **CUMPR**, which contains the cumulative probabilities of the distribution and, possibly, indexes to speed the search of the table. The routine RNGDS can be used to set up the table **CUMPR**. **RNGDT** uses the inverse CDF method to generate the variates.

Comments

1. Informational error

Type	Code	Description
3	1	The value in CUMPR(NMASS) is not exactly 1.0, but it was considered close enough to 1.0 that is was set to that value.

2. In the interest of efficiency, this routine does only limited error checking. If **CUMPR** is generated by the routine RNGDS, the error checking is sufficient.

3. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Examples

Example 1

These examples are the same ones used for the routine RNGDS. In this first example, **RNGDS** is used to set up a table and then **RNGDT** is used to generate five pseudorandom variates from the discrete distribution:

$$\Pr(X = 1) = .05$$
$$\Pr(X = 2) = .45$$
$$\Pr(X = 3) = .31$$
$$\Pr(X = 4) = .04$$
$$\Pr(X = 5) = .15$$

The cumulative probabilities are input directly in **CUMPR**, and three indexes are computed by **RNGDS** (**NNDX** = 4). Since the number of mass points is so small, the indexes would not have much effect on the speed of the generation of the random variates.

```
      USE UMACH_INT
      USE RNGDS_INT
      USE RNSET_INT
      USE RNGDT_INT
```

```
      IMPLICIT   NONE
      INTEGER    LCUMPR, NR
      PARAMETER  (LCUMPR=9, NR=5)
!
      INTEGER    IMIN, IOPT, IR(NR), ISEED, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      IMIN     = 1
      NMASS    = 5
      CUMPR(1) = 0.05
      CUMPR(2) = 0.50
      CUMPR(3) = 0.81
      CUMPR(4) = 0.85
      CUMPR(5) = 1.00
      IOPT     = 1
      NNDX     = 4
      DEL      = 0.00001
!                                Set up table
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR, IOPT=IOPT)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Generate variates
      CALL RNGDT (IMIN, NMASS, CUMPR, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Discrete random deviates: ', 5I4)
      END
!
!                                Dummy function
      REAL FUNCTION PRF (IX)
      INTEGER    IX
!
      PRF = 0.0
      RETURN
      END
```

Output

```
 Discrete random deviates:    5   2   3   3   4
```

Example 2

In this example, **RNGDS** is used to set up a table and then **RNGDT** is used to generate five pseudorandom variates from the binomial distribution with parameters 20 and 0.5. The routine **BINPR** (see Chapter 17, "Probability Distribution Functions and Inverses") is used to compute the probabilities.

```
      USE UMACH_INT
      USE RNGDS_INT
      USE RNSET_INT
      USE RNGDT_INT

      IMPLICIT   NONE
      INTEGER    LCUMPR, NR
      PARAMETER  (LCUMPR=33, NR=5)
```

```
!
      INTEGER    IMIN, IR(NR), ISEED, NMASS, NNDX, NOUT
      REAL       CUMPR(LCUMPR), DEL, PRF
      EXTERNAL   PRF
!
      CALL UMACH (2, NOUT)
      IMIN  = 0
      NMASS = 21
      NNDX  = 12
      DEL   = 0.00001
!                                  Set up table
      CALL RNGDS (PRF, DEL, NNDX, IMIN, NMASS, CUMPR)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                  Generate variates
      CALL RNGDT (IMIN, NMASS, CUMPR, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Binomial (20, 0.5) random deviates: ', 5I4)
      END
!
!                                  Compute binomial probabilities
      REAL FUNCTION PRF (IX)
      USE BINPR_INT
      INTEGER    IX
!
      PRF = BINPR(IX,20,0.5)
      RETURN
      END
```

Output

```
 Binomial (20, 0.5) random deviates:   14   9  12  10  12
```

RNGEO

Generates pseudorandom numbers from a geometric distribution.

Required Arguments

P — Probability of success on each trial. (Input)
P must be positive and less than 1.0.

IR — Vector of length **NR** containing the random geometric deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**IR**,1).

FORTRAN 90 Interface

Generic: **CALL RNGEO** (**P**, **IR** [, ...])
Specific: The specific interface name is **S_RNGEO**.

FORTRAN 77 Interface

Single: **CALL RNGEO** (**NR**, **P**, **IR**)

Description

Routine **RNGEO** generates pseudorandom numbers from a geometric distribution with parameter *P*, where *P* is the probability of getting a success on any trial. A geometric deviate can be interpreted as the number of trials until the first success (including the trial in which the first success is obtained). The probability function is

$$f(x) = P(1 - \mathrm{P})^{\mathrm{x}-1}$$

for $x = 1, 2, \ldots$ and $0 < P < 1$

The geometric distribution as defined above has mean $1/P$.

The i-th geometric deviate is generated as the smallest integer not less than $\log(U_i)/\log(1 - P)$, where the U_i are independent uniform (0, 1) random numbers (see Knuth, 1981).

The geometric distribution is often defined on 0, 1, 2, ..., with mean (1 - *P*)/*P*. Such deviates can be obtained by subtracting 1 from each element of **IR**.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNGEO** is used to generate five pseudorandom deviates from a geometric distribution with parameter **P** equal to 0.3.

```
          USE RNGEO_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    NR
          PARAMETER  (NR=5)
    !
          INTEGER    IR(NR), ISEED, NOUT
          REAL       P
    !
          CALL UMACH (2, NOUT)
          P     = 0.3
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNGEO (P, IR)
          WRITE (NOUT,99999) IR
    99999 FORMAT (' Geometric(0.3) random deviates: ', 5I8)
          END
```

Output

```
    Geometric(0.3) random deviates:        1       4       1       2       1
```

RNHYP

Generates pseudorandom numbers from a hypergeometric distribution.

Required Arguments

N — Number of items in the sample. (Input)
N must be positive.

M — Number of special items in the population, or lot. (Input)
M must be positive.

L — Number of items in the lot. (Input)
L must be greater than both N and M.

IR — Vector of length NR containing the random hypergeometric deviates. (Output)
Each element of IR can be considered to be the number of special items in a sample of size N drawn without replacement from a population of size L that contains M such special items.

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: CALL RNHYP (N, M, L, IR [, ...])
Specific: The specific interface name is S_RNHYP.

FORTRAN 77 Interface

Single: CALL RNHYP (NR, N, M, L, IR)

Description

Routine **RNHYP** generates pseudorandom numbers from a hypergeometric distribution with parameters *N*, *M*, and *L*. The hypergeometric random variable *X* can be thought of as the number of items of a given type in a random sample of size *N* that is drawn without replacement from a population of size *L* containing *M* items of this type. The probability function is

$$f(x) = \frac{\binom{M}{x}\binom{L-M}{N-x}}{\binom{L}{N}}$$

for $x = \max(0, N - L + M), 1, 2, \ldots, \min(N, M)$

If the hypergeometric probability function with parameters *N*, *M*, and *L* evaluated at $N - L + M$ (or at 0 if this is negative) is greater than the machine epsilon (**AMACH**(4) (Reference Material)), and less than 1.0 minus the machine epsilon, then **RNHYP** uses the inverse CDF technique. The routine recursively computes the hypergeometric probabilities, starting at $x = \max(0, N - L + M)$ and using the ratio $f(X = x + 1)/f(X = x)$ (see Fishman 1978, page 457).

If the hypergeometric probability function is too small or too close to 1.0, then **RNHYP** generates integer deviates uniformly in the interval $[1, L - i]$, for $i = 0, 1, \ldots$; and at the *i*-th step, if the generated deviate is less than or equal to the number of special items remaining in the lot, the occurrence of one special item is tallied and the number of remaining special items is decreased by one. This process continues until the sample size or the number of special items in the lot is reached, whichever comes first. This method can be much slower than the inverse CDF technique. The timing depends on *N*. If *N* is more than half of *L* (which in practical examples is rarely the case), the user may wish to modify the problem, replacing *N* by $L - N$, and to consider the deviates in **IR** to be the number of special items *not* included in the sample.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNHYP** is used to generate five pseudorandom deviates from a hypergeometric distribution to simulate taking random samples of size 4 from a lot containing 20 items of which 12 are defective. The resulting hypergeometric deviates represent the numbers of defectives in each of the five samples of size 4.

```
      USE RNHYP_INT
      USE UMACH_INT
```

```
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, L, M, N, NOUT
!
      CALL UMACH (2, NOUT)
      N     = 4
      M     = 12
      L     = 20
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNHYP (N, M, L, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT ('   Hypergeometric random deviates: ', 5I8)
      END
```

Output

```
   Hypergeometric random deviates:        4       2       3       3       3
```

RNLGR

Generates pseudorandom numbers from a logarithmic distribution.

Required Arguments

A — Parameter of the logarithmic distribution. (Input)
A must be positive and less than 1.0.

IR — Vector of length NR containing the random logarithmic deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: CALL RNLGR (A, IR [, ...])
Specific: The specific interface name is S_RNLGR.

FORTRAN 77 Interface

Single: CALL RNLGR (NR, A, IR)

Description

Routine RNLGR generates pseudorandom numbers from a logarithmic distribution with parameter A. The probability function is

$$f(x) = -\frac{a^x}{x\ln(1-a)}$$

for x = 1, 2, 3, ..., and $0 < a < 1$.

The methods used are described by Kemp (1981) and depend on the value of A. If A is less than 0.95, Kemp's algorithm LS, which is a "chop-down" variant of an inverse CDF technique, is used. Otherwise, Kemp's algorithm LK, which gives special treatment to the highly probable values of 1 and 2, is used.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNLGR is used to generate 5 pseudo-random deviates from a logarithmic distribution with parameter A equal to 0.3.

```
      USE RNLGR_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       A
!
      CALL UMACH (2, NOUT)
      A     = 0.3
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNLGR (A, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT ('  Logarithmic (0.3) random deviates: ', 5I8)
      END
```

Output

```
  Logarithmic (0.3) random deviates:        2       1       1       1       2
```

RNNBN

Generates pseudorandom numbers from a negative binomial distribution.

Required Arguments

RK — Negative binomial parameter. (Input)
RK must be positive.

P — Probability of success on each trial. (Input)
P must be greater than the machine epsilon, AMACH(4) (Reference Material) and less than 1.0.

IR — Vector of length NR containing the random negative binomial deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: CALL RNNBN (RK, P, IR [, ...])
Specific: The specific interface name is S_RNNBN.

FORTRAN 77 Interface

Single: CALL RNNBN (NR, RK, P, IR)

Description

Routine RNNBN generates pseudorandom numbers from a negative binomial distribution with parameters RK and P. RK and P must be positive and P must be less than 1. The probability function (with r = RK and p = P) is

$$f(x) = \binom{r+x-1}{x}(1-p)^r p^x$$

for x = 0, 1, 2, ...

If *r* is an integer, the distribution is often called the Pascal distribution and can be thought of as modeling the length of a sequence of Bernoulli trials until *r* successes are obtained, where *p* is the probability of getting a success on any trial. In this form, the random variable takes values *r*, *r* + 1, *r* + 2, … and can be obtained from the negative binomial random variable defined above by adding *r* to the negative binomial variable. This latter form is also equivalent to the sum of *r* geometric random variables defined as taking values 1, 2, 3, … .

If $rp/(1 - p)$ is less than 100 and $(1 - p)^r$ is greater than the machine epsilon, **RNNBN** uses the inverse CDF technique; otherwise, for each negative binomial deviate, **RNNBN** generates a gamma $(r, p/(1 - p))$ deviate *Y* and then generates a Poisson deviate with parameter *Y*.

Comments

1. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.
2. If **RK** is an integer, the deviates in **IR** can be thought of as the number of failures in a sequence of Bernoulli trials before **RK** successes occur.

Example

In this example, **RNNBN** is used to generate five pseudorandom deviates from a negative binomial (Pascal) distribution with parameter *r* equal to 4 and *p* equal to 0.3.

```
      USE RNNBN_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    IR(NR), ISEED, NOUT
      REAL       P, RK
!
      CALL UMACH (2, NOUT)
      P     = 0.3
      RK    = 4.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNBN (RK, P, IR)
      WRITE (NOUT,99999) IR
99999 FORMAT (' Negative binomial (4.0, 0.3) random deviates: ', 5I4)
      END
```

Output

```
 Negative binomial (4.0, 0.3) random deviates:    5   1   3   2   3
```

RNPOI

Generates pseudorandom numbers from a Poisson distribution.

Required Arguments

THETA — Mean of the Poisson distribution. (Input)
THETA must be positive.

IR — Vector of length NR containing the random Poisson deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (IR,1).

FORTRAN 90 Interface

Generic: CALL RNPOI (THETA, IR [, ...])
Specific: The specific interface name is S_RNPOI.

FORTRAN 77 Interface

Single: CALL RNPOI (NR, THETA, IR)

Description

Routine RNPOI generates pseudorandom numbers from a Poisson distribution with parameter THETA. THETA, which is the mean of the Poisson random variable, must be positive. The probability function (with θ = THETA) is

$$f(x) = e^{-\theta}\theta^{x}/x!$$

for x = 0, 1, 2, ...

If THETA is less than 15, RNPOI uses an inverse CDF method; otherwise the PTPE method of Schmeiser and Kachitvichyanukul (1981) (see also Schmeiser 1983) is used.

The **PTPE** method uses a composition of four regions, a triangle, a parallelogram, and two negative exponentials. In each region except the triangle, acceptance/rejection is used. The execution time of the method is essentially insensitive to the mean of the Poisson.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNPOI** is used to generate five pseudorandom deviates from a Poisson distribution with mean equal to 0.5.

```
          USE RNPOI_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    NR
          PARAMETER  (NR=5)
    !
          INTEGER    IR(NR), ISEED, NOUT
          REAL       THETA
    !
          CALL UMACH (2, NOUT)
          THETA = 0.5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNPOI (THETA, IR)
          WRITE (NOUT,99999) IR
    99999 FORMAT ('   Poisson(0.5) random deviates: ', 5I8)
          END
```

Output

```
      Poisson(0.5) random deviates:        2       0       1       0       1
```

RNUND

Generates pseudorandom numbers from a discrete uniform distribution.

Required Arguments

K — Parameter of the discrete uniform distribution. (Input)
The integers 1, 2, ..., **K** occur with equal probability. **K** must be positive.

IR — Vector of length **NR** containing the random discrete uniform deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**IR**,1).

FORTRAN 90 Interface

Generic: `CALL RNUND (K, IR [, ...])`
Specific: The specific interface name is **S_RNUND**.

FORTRAN 77 Interface

Single: `CALL RNUND (NR, K, IR)`

Description

Routine **RNUND** generates pseudorandom numbers from a discrete uniform distribution over the integers 1, 2, ..., **K**. A random integer is generated by multiplying **K** by a uniform (0, 1) random number, adding 1.0, and truncating the result to an integer. This, of course, is equivalent to sampling with replacement from a finite population of size **K**. To do the equivalent of sampling without replacement, the routine **RNSRI** can be used.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNUND** is used to generate five pseudorandom deviates from a discrete uniform distribution over the integers from 1 to 6.

```
          USE RNUND_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    IR(5), ISEED, K, NOUT, NR
    !
          CALL UMACH (2, NOUT)
          K     = 6
          NR    = 5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNUND (K, IR)
          WRITE (NOUT,99999) IR
    99999 FORMAT ('  Discrete uniform (1,6) random deviates: ', 5I7)
          END
```

Output

```
    Discrete uniform (1,6) random deviates:       6      2      5      4      6
```

RNBET

Generates pseudorandom numbers from a beta distribution.

Required Arguments

PIN — First beta distribution parameter. (Input)
PIN must be positive.

QIN — Second beta distribution parameter. (Input)
QIN must be positive.

R — Vector of length **NR** containing the random standard beta deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNBET** (**PIN**, **QIN**, **R** [, ...])
Specific: The specific interface names are **S_RNBET** and **D_RNBET**.

FORTRAN 77 Interface

Single: **CALL RNBET** (**NR**, **PIN**, **QIN**, **R**)
Double: The double precision name is **DRNBET**.

Description

Routine **RNBET** generates pseudorandom numbers from a beta distribution with parameters **PIN** and **QIN**, both of which must be positive. With p = **PIN** and q = **QIN**, the probability density function is

$$f(x) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} x^{p-1}(1-x)^{q-1} \quad \text{for } 0 \le x \le 1$$

where $\Gamma(\cdot)$ is the gamma function.

The algorithm used depends on the values of p and q. Except for the trivial cases of $p = 1$ or $q = 1$, in which the inverse CDF method is used, all of the methods use acceptance/rejection. If p and q are both less than 1, the method of Johnk (1964) is used; if either p or q is less than 1 and the other is greater than 1, the method of Atkinson (1979) is used; if both p and q are greater than 1, algorithm BB of Cheng (1978), which requires very little setup time, is used if **NR** is less than 4; and algorithm B4PE of Schmeiser and Babu (1980) is used if **NR** is greater than or equal to 4. Note that for p and q both greater than 1, calling **RNBET** in a loop getting less than 4 variates on each call will not yield the same set of deviates as calling **RNBET** once and getting all the deviates at once.

The values returned in **R** are less than 1.0 and greater than ε, where ε is the smallest positive number such that 1.0 **-** ε is less than 1.0.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNBET** is used to generate five pseudorandom beta (3, 2) variates.

```
      USE RNBET_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       PIN, QIN, R(NR)
!
      CALL UMACH (2, NOUT)
      PIN   = 3.0
      QIN   = 2.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNBET (PIN, QIN, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('    Beta (3,2) random deviates: ', 5F7.4)
      END
```

Output

```
    Beta (3,2) random deviates:  0.2814 0.9483 0.3984 0.3103 0.8296
```

RNCHI

Generates pseudorandom numbers from a chi-squared distribution.

Required Arguments

DF — Degrees of freedom. (Input)
DF must be positive.

R — Vector of length **NR** containing the random chi-squared deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: `CALL RNCHI (DF, R [, ...])`
Specific: The specific interface names are `S_RNCHI` and `D_RNCHI`.

FORTRAN 77 Interface

Single: `CALL RNCHI (NR, DF, R)`
Double: The double precision name is `DRNCHI`.

Description

Routine **RNCHI** generates pseudorandom numbers from a chi-squared distribution with **DF** degrees of freedom. If **DF** is an even integer less than 17, the chi-squared deviate r is generated as

$$r = -2\ln\left(\prod_{i=1}^{n} u_i\right)$$

where n = **DF**/2 and the u_i are independent random deviates from a uniform (0, 1) distribution. If **DF** is an odd integer less than 17, the chi-squared deviate is generated in the same way, except the square of a normal deviate is added to the expression above. If **DF** is greater than 16 or is not an integer, and if it is not too large to cause

overflow in the gamma random number generator, the chi-squared deviate is generated as a special case of a gamma deviate, using routine RNGAM. If overflow would occur in RNGAM, the chi-squared deviate is generated in the manner described above, using the logarithm of the product of uniforms, but scaling the quantities to prevent underflow and overflow.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNCHI is used to generate five pseudorandom chi-squared deviates with 5 degrees of freedom.

```
          USE RNCHI_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    ISEED, NOUT, NR
          REAL       DF, R(5)
    !
          CALL UMACH (2, NOUT)
          DF    = 5.0
          NR    = 5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNCHI (DF, R)
          WRITE (NOUT,99999) R
    99999 FORMAT ('  Chi-squared random deviates with 5 df: ', 5F7.3)
          END
```

Output

```
      Chi-squared random deviates with 5 df:  12.090  0.481  1.798 14.871  1.748
```

RNCHY

Generates pseudorandom numbers from a Cauchy distribution.

Required Arguments

R — Vector of length **NR** containing the random Cauchy deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNCHY** (**R** [, ...])
Specific: The specific interface names are **S_RNCHY** and **D_RNCHY**.

FORTRAN 77 Interface

Single: **CALL RNCHY** (**NR**, **R**)
Double: The double precision name is **DRNCHY**.

Description

Routine **RNCHY** generates pseudorandom numbers from a standard Cauchy distribution. The probability density function is

$$f(x) = \frac{1}{\pi\left(1 + x^2\right)}$$

Use of the inverse CDF technique would yield a Cauchy deviate from a uniform (0, 1) deviate, u, as tan[**π**(u - 5)]. Rather than evaluating a tangent directly, however, **RNCHY** generates two uniform (-1, 1) deviates, x_1 and x_2. These values can be thought of as sine and cosine values. If

$$x_1^2 + x_2^2$$

is less than or equal to 1, then x_1/x_2 is delivered as the Cauchy deviate; otherwise, x_1 and x_2 are rejected and two new uniform (–1, 1) deviates are generated. This method is also equivalent to taking the ratio of two independent normal deviates.

Deviates from the Cauchy distribution with median *T* and first quartile *T* - *S*, that is, with density

$$f(x) = \frac{S}{\pi\left[S^2 + (x - T)^2\right]}$$

can be obtained by scaling the output from **RNCHY**. The following statements (in single precision) would yield random deviates from this Cauchy distribution.

```
    CALL RNCHY (NR, R)
    CALL SSCAL (NR, S, R, 1)
    CALL SADD (NR, T, R, 1)
```

The Cauchy distribution is a member of the symmetric stable family of distributions. The routine **RNSTA** can be used to generate deviates from this more general family of distributions or even from the stable family not requiring symmetry.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNCHY** is used to generate five pseudorandom deviates from a Cauchy distribution.

```
          USE RNCHY_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    ISEED, NOUT, NR
          REAL       R(5)
    !
          CALL UMACH (2, NOUT)
          NR    = 5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNCHY (R)
          WRITE (NOUT,99999) R
    99999 FORMAT ('      Cauchy random deviates: ', 5F8.4)
          END
```

Output

```
    Cauchy random deviates:   3.5765  0.9353 15.5797  2.0815 -0.1333
```

RNEXP

Generates pseudorandom numbers from a standard exponential distribution.

Required Arguments

R — Vector of length **NR** containing the random standard exponential deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: `CALL RNEXP (R [, ...])`
Specific: The specific interface names are **S_RNEXP** and **D_RNEXP**.

FORTRAN 77 Interface

Single: `CALL RNEXP (NR, R)`
Double: The double precision name is **DRNEXP**.

Description

Routine **RNEXP** generates pseudorandom numbers from a standard exponential distribution. The probability density function is $f(x) = e^{-X}$ for $x > 0$. **RNEXP** uses an antithetic inverse CDF technique; that is, a uniform random deviate U is generated and the inverse of the exponential cumulative distribution function is evaluated at 1.0 - U to yield the exponential deviate.

Deviates from the exponential distribution with mean **THETA** can be generated by using **RNEXP** and then multiplying each entry in **R** by **THETA**. The following statements (in single precision using the routine **SSCAL** (Reference Material)) would yield random deviates from such a distribution:

```
    USE IMSL_LIBRARIES
                  ⋮
    CALL RNEXP (R, NR)
    CALL SSCAL (NR, THETA, R, 1)
```

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNEXP is used to generate five pseudorandom deviates from a standard exponential distribution.

```
      USE RNEXP_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNEXP (R)
      WRITE (NOUT,99999) R
99999 FORMAT ('    Exponential random deviates: ', 5F8.4)
      END
```

Output

```
    Exponential random deviates:   0.0344  1.3443  0.2662  0.5633  0.1686
```

RNEXT

Generates pseudorandom numbers from a mixture of two exponential distributions.

Required Arguments

THETA1 — Mean of the exponential distribution that has the larger mean. (Input)

THETA2 — Mean of the exponential distribution that has the smaller mean. (Input)
THETA2 must be positive and less than or equal to THETA1.

P — Mixing parameter. (Input)
P must be nonnegative and less than or equal to THETA1/(THETA1 - THETA2).

R — Vector of length NR containing the random deviates from a mixture of exponentials. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNEXT (THETA1, THETA2, P, R [, ...])
Specific: The specific interface names are S_RNEXT and D_RNEXT.

FORTRAN 77 Interface

Single: CALL RNEXT (NR, THETA1, THETA2, P, R)
Double: The double precision name is DRNEXT.

Description

Routine RNEXT generates pseudorandom numbers from a mixture of two exponential distributions. The probability density function is

$$f(x) = \frac{p}{\theta_1} e^{-x/\theta_1} + \frac{1-p}{\theta_2} e^{-x/\theta_2} \quad \text{for } x > 0$$

where p = P, θ_1 = THETA1, and θ_2 = THETA2.

In the case of a convex mixture, that is, the case $0 < p < 1$, the mixing parameter p is interpretable as a probability; and RNEXT with probability p generates an exponential deviate with mean θ_1, and with probability $1 - p$ generates an exponential with mean θ_2. When p is greater than 1, but less than $\theta_1/(\theta_1 - \theta_2)$, then either an exponential deviate with mean θ_1 or the sum of two exponentials with means θ_1 and θ_2 is generated. The probabilities are $q = p - (p - 1)\theta_1/\theta_2$ and $1 - q$, respectively, for the single exponential and the sum of the two exponentials.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNEXT is used to generate five pseudorandom deviates from a mixture of exponentials with means 2 and 1, respectively, and with mixing parameter 0.5.

```
          USE RNEXT_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    ISEED, NOUT, NR
          REAL       P, R(5), THETA1, THETA2
    !
          CALL UMACH (2, NOUT)
          THETA1 = 2.0
          THETA2 = 1.0
          P      = 0.5
          NR     = 5
          ISEED  = 123457
          CALL RNSET (ISEED)
          CALL RNEXT (THETA1, THETA2, P, R)
          WRITE (NOUT,99999) R
    99999 FORMAT (' Random deviates from a mixture of exponentials: ', /, &
                  5X, 5F8.4)
          END
```

Output

```
    Random deviates from a mixture of exponentials:
          0.0700  1.3024  0.6301  1.9756  0.3716
```

RNEXV

Generates pseudorandom numbers from an extreme value distribution.

Required Arguments

AMU — The location parameter of the extreme value distribution. (Input)

BETA — The scale parameter of the extreme value distribution. (Input)

R — Vector of length NR containing the random extreme value deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNEXV (AMU, BETA, R [, ...])
Specific: The specific interface names are S_RNEXV and D_RNEXV.

FORTRAN 77 Interface

Single: CALL RNEXV (NR, AMU, BETA, R)
Double: The double precision name is DRNEXV.

Description

Routine RNEXV generates pseudorandom numbers from an extreme value distribution generated by evaluating uniform variates u_i, equating to the CDF, and then solving for x_i by first computing

$$\frac{x_i - \mu}{\beta} = \log\left(-\log\left(1 - u_i\right)\right)$$

Where μ = AMU and β = BETA.

The routine ALNREL is used to accurately evaluate the sub-expression log(1 - u_i).

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNEXV is used to generate five pseudorandom deviates from an extreme value distribution with location parameter equal to 0.0, and scale parameter 1.0.

```
      USE UMACH_INT
      USE RNEXV_INT
      IMPLICIT NONE
      INTEGER NR
      PARAMETER (NR=5)
      INTEGER NOUT
      REAL AAMU, B, R(NR)
      CALL UMACH(2, NOUT)
      CALL RNSET(123457)
      AAMU = 0.0
      B   = 1.0
      CALL RNEXV(AAMU, B, R)
      WRITE (NOUT, 99999) R
99999 FORMAT (' Extreme value random deviates: ', 5F10.4)
      END
```

Output

```
 Extreme value random deviates:    1.2202  -1.1971   0.3740  -0.1715  0.6223
```

RNFDF

Generates pseudorandom numbers from the *F* distribution.

Required Arguments

DFN — Numerator degrees of freedom. (Input)
DFN must be positive.

DFD — Denominator degrees of freedom. (Input)
DFD must be positive.

R — Vector of length NR containing the random F deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNFDF (DFN, DFD, R [, ...])
Specific: The specific interface names are S_RNFDF and D_RNFDF.

FORTRAN 77 Interface

Single: CALL RNFDF (NR, DFN, DFD, R)
Double: The double precision name is DRNFDF.

Description

Routine RNFDF generates pseudorandom numbers from an F distribution (see Chapter 17, "Probability Distribution Functions and Inverses", routine FDF).

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNFDF is used to generate five pseudorandom deviates from an F distribution with parameters DFN = 2 and DFD = 3.

```
      USE UMACH_INT
      USE RNFDF_INT
      IMPLICIT NONE
      INTEGER NR
      PARAMETER (NR=5)
      INTEGER NOUT
      REAL DFD, DFN, R(NR)

      CALL UMACH(2, NOUT)
      CALL RNSET(123457)

      DFN = 2.0e0
      DFD = 3.0e0
      CALL RNFDF(DFN, DFD, R)
      WRITE (NOUT, 99999) R
99999 FORMAT (' F Random deviates: ', 5F10.4)
      END
```

Output

```
 F Random deviates:     0.0814    0.3639    0.1323    1.5415    1.0350
```

RNGAM

Generates pseudorandom numbers from a standard gamma distribution.

Required Arguments

A — The shape parameter of the gamma distribution. (Input)
This parameter must be positive.

R — Vector of length **NR** containing the random standard gamma deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNGAM** (**A**, **R** [, ...])
Specific: The specific interface names are **S_RNGAM** and **D_RNGAM**.

FORTRAN 77 Interface

Single: **CALL RNGAM** (**NR**, **A**, **R**)
Double: The double precision name is **DRNGAM**.

Description

Routine **RNGAM** generates pseudorandom numbers from a gamma distribution with shape parameter a and unit scale parameter. The probability density function is

$$f(x) = \frac{1}{\Gamma(a)} x^{a-1} e^{-x} \quad \text{for } x \geq 0$$

Various computational algorithms are used depending on the value of the shape parameter *a*. For the special case of *a* = 0.5, squared and halved normal deviates are used; and for the special case of *a* = 1.0, exponential deviates (from IMSL routine RNEXP) are used. Otherwise, if *a* is less than 1.0, an acceptance-rejection method due to Ahrens, described in Ahrens and Dieter (1974), is used; if *a* is greater than 1.0, a ten-region rejection procedure developed by Schmeiser and Lal (1980) is used.

Deviates from the two-parameter gamma distribution with shape parameter *a* and scale parameter *b* can be generated by using **RNGAM** and then multiplying each entry in **R** by *b*. The following statements (in single precision) would yield random deviates from a gamma (*a*, *b*) distribution.

```
    CALL RNGAM (NR, A, R)
    CALL SSCAL (NR, B, R, 1)
```

The Erlang distribution is a standard gamma distribution with the shape parameter having a value equal to a positive integer; hence, **RNGAM** generates pseudorandom deviates from an Erlang distribution with no modifications required.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNGAM** is used to generate five pseudorandom deviates from a gamma (Erlang) distribution with shape parameter equal to 3.0.

```
          USE RNGAM_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    NR
          PARAMETER  (NR=5)
    !
          INTEGER    ISEED, NOUT
          REAL       A, R(NR)
    !
          CALL UMACH (2, NOUT)
          A     = 3.0
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNGAM (A, R)
          WRITE (NOUT,99999) R
```

```
99999 FORMAT ('  Gamma(3) random deviates: ', 5F8.4)
      END
```

Output

```

 Gamma(3) random deviates:   6.8428  3.4452  1.8535  3.9992  0.7794
```

RNGCS

Sets up table to generate pseudorandom numbers from a general continuous distribution.

Required Arguments

CDF — User-supplied FUNCTION to compute the cumulative distribution function. The form is CDF(X), where

X — Point at which the distribution function is to be evaluated. (Input)

CDF — Value of the distribution function at X. (Output)
CDF must be declared EXTERNAL in the calling program.

IOPT — Indicator of the extent to which TABLE is initialized prior to calling RNGCS. (Input)

IOPT	Action
0	RNGCS fills the last four columns of TABLE. The user inputs the points at which the CDF is to be evaluated in the first column of TABLE. These must be in ascending order.
1	RNGCS fills the last three columns of TABLE. CDF is not used and may be a dummy function; instead, the cumulative distribution function is specified in the first two columns of TABLE. The abscissas (in the first column) must be in ascending order and the function must be strictly monotonically increasing.

TABLE — NDATA by 5 table to be used for interpolation of the cumulative distribution function. (Input and output)
The first column of TABLE contains abscissas of the cumulative distribution function in ascending order, the second column contains the values of the CDF (which must be strictly increasing), and the remaining columns contain values used in interpolation. The first row of TABLE corresponds to the left limit of the support of the distribution and the last row corresponds to the right limit of the support; that is, TABLE(1, 2) = 0.0 and TABLE(NDATA, 2) = 1.0.

Optional Arguments

NDATA — Number of points at which the CDF is evaluated for interpolation. (Input)
NDATA must be greater than or equal to 4.
Default: NDATA = size (TABLE,1).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: CALL RNGCS (CDF, IOPT, TABLE [, ...])
Specific: The specific interface names are S_RNGCS and D_RNGCS.

FORTRAN 77 Interface

Single: CALL RNGCS (CDF, IOPT, NDATA, TABLE, LDTABL)
Double: The double precision name is DRNGCS.

Description

Routine RNGCS sets up a table that routine RNGCT can use to generate pseudorandom deviates from a continuous distribution. The distribution is specified by its cumulative distribution function, which can be supplied either in tabular form in TABLE or by a FORTRAN function CDF. See the documentation for the routine RNGCT for a description of the method.

Comments

1. Informational error

Type	Code	Description
3	1	The values in TABLE(1, 2) and/or TABLE(NDATA, 2) are not exactly 0.0 and 1.0, respectively, but they are considered close enough to these values that they are set to these values.

2. The routine RNGCT uses the table set up by RNGCS to generate random numbers from the distribution with CDF represented in TABLE.

Example

In this example, RNGCS is used to set up a table to generate pseudorandom variates from a beta distribution. This example is continued in the documentation for routine RNGCT to generate the random variates.

```
      USE RNGCS_INT
      USE UMACH_INT
```

```
      IMPLICIT   NONE
      INTEGER    LDTABL
      PARAMETER  (LDTABL=100)
!
      INTEGER    I, IOPT, NINT, NOUT
      REAL       CDF, PIN, QIN, TABLE(LDTABL,5), X
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   CDF
!
      CALL UMACH (2, NOUT)
      PIN  = 3.0
      QIN  = 2.0
      IOPT = 0
      NINT = 100
      X    = 0.0
!                                Fill the first column of the table
!                                with abscissas for interpolation.
      DO 10  I=1, NINT
         TABLE(I,1) = X
         X          = X + 0.01
   10 CONTINUE
      CALL RNGCS (CDF, IOPT, TABLE)
      WRITE (NOUT,99999) (TABLE(I,1),TABLE(I,2),I=1,10)
99999 FORMAT ('   First few elements of the table: ', F4.2, F8.4, /, &
            (36X,F4.2,F8.4))
      END
!
!                                Beta distribution function
      REAL FUNCTION CDF (X)
      REAL       X
!
      REAL       BETDF, PIN, QIN
      COMMON     /BCOM/ PIN, QIN
      EXTERNAL   BETDF
!
      CDF = BETDF(X,PIN,QIN)
      RETURN
      END
```

Output

```
*** WARNING  ERROR 1 from RNGCS.  The values of the CDF in the second
***          column of TABLE did not begin at 0.0 and end at 1.0, but they
***          have been adjusted. Prior to adjustment,
***          TABLE(1,2) = 0.000000E+00 and TABLE(NDATA,2) = 9.994079E-01.
  First few elements of the table: 0.00  0.0000
                                   0.01  0.0000
                                   0.02  0.0000
                                   0.03  0.0001
                                   0.04  0.0002
                                   0.05  0.0005
                                   0.06  0.0008
                                   0.07  0.0013
                                   0.08  0.0019
                                   0.09  0.0027
```

RNGCT

Generates pseudorandom numbers from a general continuous distribution.

Required Arguments

TABLE — NDATA by 5 table to be used for interpolation of the cumulative distribution function. (Input)
The first column of TABLE contains abscissas of the cumulative distribution function in ascending order, the second column contains the values of the CDF (which must be strictly increasing beginning with 0.0 and ending at 1.0) and the remaining columns contain values used in interpolation. This table is set up using routine RNGCS.

R — Vector of length NR containing the random deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

NDATA — Number of points at which the cumulative distribution function is evaluated for interpolation. (Input)
NDATA must be greater than or equal to 4.
Default: NDATA = size (TABLE,1).

LDTABL — Leading dimension of TABLE exactly as specified in the dimension statement in the calling program. (Input)
Default: LDTABL = size (TABLE,1).

FORTRAN 90 Interface

Generic: CALL RNGCT (TABLE, R [, ...])
Specific: The specific interface names are S_RNGCT and D_RNGCT.

FORTRAN 77 Interface

Single: CALL RNGCT (NR, NDATA, TABLE, LDTABL, R)
Double: The double precision name is DRNGCT.

Description

Routine **RNGCT** generates pseudorandom numbers from a continuous distribution using the inverse CDF technique, by interpolation of points of the distribution function given in **TABLE**, which is set up by routine **RNGCS**. A strictly monotone increasing distribution function is assumed. The interpolation is by an algorithm attributable to Akima (1970), using piecewise cubics. The use of this technique for generation of random numbers is due to Guerra, Tapia, and Thompson (1976), who give a description of the algorithm and accuracy comparisons between this method and linear interpolation. The relative errors using the Akima interpolation are generally considered very good.

Comments

1. The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.
2. In the interest of efficiency, this routine does only limited error checking. If **TABLE** is generated by the routine **RNGCS**, the error checking is sufficient.

Example

In this example, **RNGCS** is used to set up a table for generation of beta pseudorandom deviates. The CDF for this distribution is computed by the routine **BETDF** (see Chapter 17, "Probability Distribution Function and Inverses"). The table contains 100 points at which the CDF is evaluated and that are used for interpolation.

```
          USE RNGCT_INT
          USE UMACH_INT
          USE RNGCS_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    LDTABL, NR
          PARAMETER  (LDTABL=100, NR=5)
    !
          INTEGER    I, IOPT, ISEED, NINT, NOUT
          REAL       CDF, PIN, QIN, R(NR), TABLE(LDTABL,5), X
          COMMON     /BCOM/ PIN, QIN
          EXTERNAL   CDF
    !
          CALL UMACH (2, NOUT)
          PIN  = 3.0
          QIN  = 2.0
          IOPT = 0
          NINT = 100
          X    = 0.0
    !                                Fill the first column of the table
    !                                with abscissas for interpolation.
          DO 10  I=1, NINT
             TABLE(I,1) = X
```

```
            X          = X + 0.01
      10 CONTINUE
         CALL RNGCS (CDF, IOPT, TABLE)
      !                                Initialize seed of random number
      !                                generator.
         ISEED = 123457
         CALL RNSET (ISEED)
      !                                Now generate the random deviates.
         CALL RNGCT (TABLE, R)
         WRITE (NOUT,99999) R
      99999 FORMAT ('  Beta (3,2) random deviates: ', 5F7.4)
         END
      !
      !                                Beta distribution function
         REAL FUNCTION CDF (X)
         REAL       X
      !
         REAL       BETDF, PIN, QIN
         COMMON     /BCOM/ PIN, QIN
         EXTERNAL   BETDF
      !
         CDF = BETDF(X,PIN,QIN)
         RETURN
         END
```

Output

```
 *** WARNING  ERROR 1 from RNGCS.  The values of the CDF in the second
 ***          column of TABLE did not begin at 0.0 and end at 1.0, but they
 ***          have been adjusted. Prior to adjustment,
 ***          TABLE(1,2) = 0.000000E+00 and TABLE(NDATA,2) = 9.994079E-01.
  Beta (3,2) random deviates:  0.9208 0.4641 0.7668 0.6536 0.8171
```

RNLNL

Generates pseudorandom numbers from a lognormal distribution.

Required Arguments

XM — Mean of the underlying normal distribution. (Input)

S — Standard deviation of the underlying normal distribution. (Input)
S must be positive.

R — Vector of length NR containing the random lognormal deviates. (Output)
The log of each element of R has a normal distribution with mean XM and standard deviation S.

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNLNL (XM, S, R [, ...])
Specific: The specific interface names are S_RNLNL and D_RNLNL.

FORTRAN 77 Interface

Single: CALL RNLNL (NR, XM, S, R)
Double: The double precision name is DRNLNL.

Description

Routine RNLNL generates pseudorandom numbers from a lognormal distribution with parameters XM and S. The scale parameter in the underlying normal distribution, S, must be positive. The method is to generate normal deviates with mean XM and standard deviation S and then to exponentiate the normal deviates.

With μ = XM and σ = S, the probability density function for the lognormal distribution is

$$f(x) = \frac{1}{\sigma x\sqrt{2\pi}}\exp\left[-\frac{1}{2\sigma^2}(\ln x - \mu)^2\right] \quad \text{for } x > 0$$

The mean and variance of the lognormal distribution are $\exp(\mu + \sigma^2/2)$ and $\exp(2\mu + 2\sigma^2) - \exp(2\mu + \sigma^2)$, respectively.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNLNL is used to generate five pseudorandom lognormal deviates with $\mu = 0$ and $\sigma = 1$.

```
      USE RNLNL_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       R(NR), S, XM
!
      CALL UMACH (2, NOUT)
      XM    = 0.0
      S     = 1.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNLNL (XM, S, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  Lognormal random deviates: ', 5F8.4)
      END
```

Output

```
 Lognormal random deviates:   7.7801  2.9543  1.0861  3.5885  0.2935
```

RNNOA

Generates pseudorandom numbers from a standard normal distribution using an acceptance/rejection method.

Required Arguments

R — Vector of length **NR** containing the random standard normal deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNNOA** (**R** [, ...])
Specific: The specific interface names are **S_RNNOA** and **D_RNNOA**.

FORTRAN 77 Interface

Single: **CALL RNNOA** (**NR**, **R**)
Double: The double precision name is **DRNNOA**.

Description

Routine **RNNOA** generates pseudorandom numbers from a standard normal (Gaussian) distribution using an acceptance/rejection technique due to Kinderman and Ramage (1976). In this method, the normal density is represented as a mixture of densities over which a variety of acceptance/rejection methods due to Marsaglia (1964), Marsaglia and Bray (1964), and Marsaglia, MacLaren, and Bray (1964) are applied. This method is faster than the inverse CDF technique used in **RNNOR** to generate standard normal deviates.

Deviates from the normal distribution with mean **XM** and standard deviation **XSTD** can be obtained by scaling the output from **RNNOA**. The following statements (in single precision) would yield random deviates from a normal (**XM**, **XSTD****2) distribution.

```
    CALL RNNOA (NR, R)
    CALL SSCAL (NR, XSTD, R, 1)
    CALL SADD (NR, XM, R, 1)
```

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNNOA** is used to generate five pseudorandom deviates from a standard normal distribution.

```
      USE RNNOA_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNOA (R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  Standard normal random deviates: ', 5F8.4)
      END
```

Output

```
 Standard normal random deviates:   2.0516  1.0833  0.0826  1.2777 -1.2260
```

RNNOF

This function generates a pseudorandom number from a standard normal distribution.

Function Return Value

RNNOF — Function value, a random standard normal deviate. (Output)
See Comment 1.

Required Arguments

None.

FORTRAN 90 Interface

Generic: **RNNOF** ()
Specific: The specific interface names are **S_RNNOF** and **D_RNNOF**.

FORTRAN 77 Interface

Single: **RNNOF** ()
Double: The double precision name is **DRNNOF**.

Description

Routine **RNNOF** is the function form of **RNNOR**. If several standard normal deviates are needed, it may be more efficient to obtain them all at once by a call to **RNNOR**, rather than by several references to **RNNOF**.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = RNNOF()
   Y = SQRT(X)
   ```

   must be used rather than

```
Y = SQRT(RNNOF())
```

If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

3. This function has a side effect: it changes the value of the seed, which is passed through a common block.

Example

In this example, RNNOF is used to generate five pseudorandom standard normal numbers.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNNOF_INT

      IMPLICIT   NONE
      INTEGER    I, ISEED, NOUT, NR
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      ISEED = 123457
      CALL RNSET (ISEED)
      NR=5
      DO 10  I=1, NR
         R(I) = RNNOF()
   10 CONTINUE
      WRITE (NOUT,99999) R
99999 FORMAT ('  Standard normal random deviates: ', 5F8.4)
      END
```

Output

```
 Standard normal random deviates:   1.8279  -0.6412  0.7266  0.1747  1.0145
```

RNNOR

Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.

Required Arguments

R — Vector of length **NR** containing the random standard normal deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL RNNOR** (**R** [, ...])
Specific: The specific interface names are **S_RNNOR** and **D_RNNOR**.

FORTRAN 77 Interface

Single: **CALL RNNOR** (**NR**, **R**)
Double: The double precision name is **DRNNOR**.

Description

Routine **RNNOR** generates pseudorandom numbers from a standard normal (Gaussian) distribution using an inverse CDF technique. In this method, a uniform (0,1) random deviate is generated and then the inverse of the normal distribution function is evaluated at that point, using the routine **ANORIN** (see Chapter 17, "Probability Distribution Function and Inverses"). This method is slower than the acceptance/rejection technique used in the routine **RNNOA** to generate standard normal deviates. Deviates from the normal distribution with mean **XM** and standard deviation **XSTD** can be obtained by scaling the output from **RNNOR**. The following statements (in single precision, using the routines **SSCAL** (IMSL MATH/LIBRARY) and **SADD** (IMSL MATH/LIBRARY).) would yield random deviates from a normal (**XM**, **XSTD****2) distribution.

```
    USE IMSL_LIBRARIES
              ⋮
    CALL RNNOR (R, NR)
    CALL SSCAL (NR, XSTD, R, 1)
```

```
    CALL SADD (NR, XM, R, 1)
```

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNNOR** is used to generate five pseudorandom deviates from a standard normal distribution.

```
          USE RNNOR_INT
          USE UMACH_INT
          USE RNSET_INT
          IMPLICIT   NONE
          INTEGER    ISEED, NOUT, NR
          REAL       R(5)
    !
          CALL UMACH (2, NOUT)
          NR    = 5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNNOR (R)
          WRITE (NOUT,99999) R
    99999 FORMAT ('  Standard normal random deviates: ', 5F8.4)
          END
```

Output

```
    Standard normal random deviates:   1.8279 -0.6412  0.7266  0.1747  1.0145
```

RNRAL

Generates pseudorandom numbers from a Rayleigh distribution.

Required Arguments

ALPHA — Parameter of the Rayleigh distribution. (Input)
ALPHA must be greater than 0.

R — Vector of length NR containing the random Rayleigh deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNRAL (ALPHA, R [, ...])
Specific: The specific interface names are S_RNRAL and D_RNRAL.

FORTRAN 77 Interface

Single: CALL RNRAL (NR, ALPHA, R)
Double: The double precision name is DRNRAL.

Description

Routine RNRAL generates pseudorandom numbers from a Rayleigh distribution (see Chapter 17, "Probability Distribution Function and Inverses", routine RALDF).

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNRAL** is used to generate five pseudorandom deviates from a Rayleigh distribution with parameter **ALPHA** = 0.5.

```
      USE UMACH_INT
      USE RNRAL_INT
      IMPLICIT NONE
      INTEGER NR
      PARAMETER (NR=5)
      INTEGER NOUT
      REAL ALPHA, R(NR)
      CALL UMACH(2, NOUT)
      CALL RNSET(123457)
      ALPHA = 0.5
      CALL RNRAL(ALPHA, R)
      WRITE (NOUT, 99999) R
99999 FORMAT (' Rayleigh random deviates: ', 5F10.4)
      END
```

Output

```
 Rayleigh random deviates:     0.1311    0.8199    0.3648    0.5307    0.2904
```

RNSTA

Generates pseudorandom numbers from a stable distribution.

Required Arguments

ALPHA — Characteristic exponent of the stable distribution. (Input)
This parameter must be positive and less than or equal to 2.

BPRIME — Skewness parameter of the stable distribution. (Input)
When BPRIME = 0, the distribution is symmetric. Unless ALPHA = 1, BPRIME is not the usual skewness parameter of the stable distribution. BPRIME must be greater than or equal to – 1 and less than or equal to 1.

R — Vector of length NR containing the random stable deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNSTA (ALPHA, BPRIME, R [, ...])
Specific: The specific interface names are S_RNSTA and D_RNSTA.

FORTRAN 77 Interface

Single: CALL RNSTA (NR, ALPHA, BPRIME, R)
Double: The double precision name is DRNSTA.

Description

Routine RNSTA generates pseudorandom numbers from a stable distribution with parameters ALPHA and BPRIME. ALPHA is the usual characteristic exponent parameter α and BPRIME is related to the usual skewness parameter β of the stable distribution. With the restrictions $0 < \alpha \leq 2$ and $-1 \leq \beta \leq 1$, the characteristic function of the distribution is

$$\varphi(t) = \exp[-|t|^{\alpha} \exp(-\pi i\beta(1 - |1 - \alpha|) \operatorname{sign}(t)/2)] \quad \text{for } \alpha \neq 1$$

and

$$\varphi(t) = \exp[-|t|(1 + 2i\beta \ln|t| \operatorname{sign}(t)/\pi)] \quad \text{for } \alpha = 1$$

When $\beta = 0$, the distribution is symmetric. In this case, if $\alpha = 2$, the distribution is normal with mean 0 and variance 2; and if $\alpha = 1$, the distribution is Cauchy.

The parameterization using **BPRIME** and the algorithm used here are due to Chambers, Mallows, and Stuck (1976). The relationship between **BPRIME** = β' and the standard β is

$$\beta' = -\tan(\pi(1 - \alpha)/2) \tan(-\pi\beta(1 - |1 - \alpha|)/2) \quad \text{for } \alpha \neq 1$$

and

$$\beta' = \beta \quad \text{for } \alpha = 1$$

The algorithm involves formation of the ratio of a uniform and an exponential random variate.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNSTA** is used to generate five pseudorandom symmetric stable variates with characteristic exponent 1.5. The tails of this distribution are heavier than those of a normal distribution, but not so heavy as those of a Cauchy distribution. The variance of this distribution does not exist, however. (This is the case for any stable distribution with characteristic exponent less than 2.)

```
          USE RNSTA_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    NR
          PARAMETER  (NR=5)
    !
          INTEGER    ISEED, NOUT
          REAL       ALPHA, BPRIM, R(NR)
    !
          CALL UMACH (2, NOUT)
          ALPHA = 1.5
          BPRIM = 0.0
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNSTA (ALPHA, BPRIM, R)
          WRITE (NOUT,99999) R
```

```
99999 FORMAT (' Stable random deviates: ', 5F9.4)
      END
```

Output

```
 Stable random deviates:    4.4091   1.0564   2.5463   5.6724   2.1656
```

RNSTT

Generates pseudorandom numbers from a Student's *t* distribution.

Required Arguments

DF — Degrees of freedom. (Input)
DF must be positive.

R — Vector of length NR containing the random Student's *t* deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNSTT (DF, R [, ...])
Specific: The specific interface names are S_RNSTT and D_RNSTT.

FORTRAN 77 Interface

Single: CALL RNSTT (NR, DF, R)
Double: The double precision name is DRNSTT.

Description

Routine RNSTT generates pseudo-random numbers from a Student's *t* distribution with DF degrees of freedom, using a method suggested by Kinderman, Monahan, and Ramage (1977). The method ("TMX" in the reference) involves a representation of the *t* density as the sum of a triangular density over (–2, 2) and the difference of this and the *t* density. The mixing probabilities depend on the degrees of freedom of the *t* distribution. If the triangular density is chosen, the variate is generated as the sum of two uniforms; otherwise, an acceptance/rejection method is used to generate a variate from the difference density.

For degrees of freedom less than 100, **RNSTT** requires approximately twice the execution time as routine **RNNOA** which generates pseudorandom normal deviates. The execution time of **RNSTT** increases very slowly as the degrees of freedom increase. Since for very large degrees of freedom the normal distribution and the *t* distribution are very similar, the user may find that the difference in the normal and the *t* does not warrant the additional generation time required to use **RNSTT** instead of **RNNOA**.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNSTT** is used to generate 5 pseudo-random *t* variates with 10 degrees of freedom.

```
      USE RNSTT_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       DF, R(NR)
!
      CALL UMACH(2, NOUT)
      DF = 10.0
      ISEED = 123457
      CALL RNSET(ISEED)
      CALL RNSTT(DF, R)
      WRITE(NOUT, 99999) R
99999 FORMAT ('  t (10) random deviates: ', 5F8.4)
      END
```

Output

```
    t (10) random deviates:   0.6152  1.1528  0.0881  1.3382 -0.9893
```

RNTRI

Generates pseudorandom numbers from a triangular distribution on the interval (0, 1).

Required Arguments

R — Vector of length **NR** containing the random triangular deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: `CALL RNTRI (R [, ...])`
Specific: The specific interface names are `S_RNTRI` and `D_RNTRI`.

FORTRAN 77 Interface

Single: `CALL RNTRI (NR, R)`
Double: The double precision name is `DRNTRI`.

Description

Routine **RNTRI** generates pseudorandom numbers from a triangular distribution over the unit interval. The probability density function is $f(x) = 4x$, for $0 \le x \le 0.5$, and $f(x) = 4(1 - x)$, for $0.5 < x \le 1$. **RNTRI** uses an inverse CDF technique.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNTRI** is used to generate five pseudorandom deviates from a triangular distribution.

```
          USE RNTRI_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    ISEED, NOUT, NR
          REAL       R(5)
    !
          CALL UMACH (2, NOUT)
          NR    = 5
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNTRI (R)
          WRITE (NOUT,99999) R
    99999 FORMAT ('    Triangular random deviates: ', 5F8.4)
          END
```

Output

```
    Triangular random deviates:   0.8700  0.3610  0.6581  0.5360  0.7215
```

RNVMS

Generates pseudorandom numbers from a von Mises distribution.

Required Arguments

C — Parameter of the von Mises distribution. (Input)
This parameter must be greater than one half of machine epsilon. (On many machines, the lower bound for **C** is 10^{-3}.)

R — Vector of length **NR** containing the random von Mises deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: `CALL RNVMS (C, R [, ...])`
Specific: The specific interface names are `S_RNVMS` and `D_RNVMS`.

FORTRAN 77 Interface

Single: `CALL RNVMS (NR, C, R)`
Double: The double precision name is `DRNVMS`.

Description

Routine **RNVMS** generates pseudorandom numbers from a von Mises distribution with parameter **C**, which must be positive. With c = **C**, the probability density function is

$$f(x) = \frac{1}{2\pi I_0(c)} \exp\left[c \cos(x)\right] \quad \text{for } -\pi < x < \pi$$

where $I_0(c)$ is the modified Bessel function of the first kind of order 0. The probability density equals 0 outside the interval (−**π**, **π**).

The algorithm is an acceptance/rejection method using a wrapped Cauchy distribution as the majorizing distribution. It is due to Best and Fisher (1979).

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNVMS is used to generate five pseudorandom von Mises variates with $c = 1$.

```
      USE RNVMS_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NR
      PARAMETER  (NR=5)
!
      INTEGER    ISEED, NOUT
      REAL       C, R(NR)
!
      CALL UMACH (2, NOUT)
      C     = 1.0
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNVMS (C, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  Von Mises random deviates: ', 5F8.4)
      END
```

Output

```
 Von Mises random deviates:   0.2472 -2.4326 -1.0216 -2.1722 -0.5029
```

RNWIB

Generates pseudorandom numbers from a Weibull distribution.

Required Arguments

A — The shape parameter of the Weibull distribution. (Input)
This parameter must be positive.

R — Vector of length **NR** containing the random Weibull deviates. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: **NR** = size (**R**,1).

FORTRAN 90 Interface

Generic: `CALL RNWIB` (A, R [, ...])
Specific: The specific interface names are `S_RNWIB` and `D_RNWIB`.

FORTRAN 77 Interface

Single: `CALL RNWIB` (NR, A, R)
Double: The double precision name is `DRNWIB`.

Description

Routine **RNWIB** generates pseudorandom numbers from a Weibull distribution with shape parameter **A** and unit scale parameter. The probability density function is

$$f(x) = Ax^{A-1}e^{-x^A} \quad \text{for } x \geq 0$$

Routine **RNWIB** uses an antithetic inverse CDF technique to generate a Weibull variate; that is, a uniform random deviate *U* is generated and the inverse of the Weibull cumulative distribution function is evaluated at 1.0 - *U* to yield the Weibull deviate.

Deviates from the two-parameter Weibull distribution with shape parameter **A** and scale parameter *B* can be generated by using **RNWIB** and then multiplying each entry in **R** by *B*. The following statements (using routine **SSCAL** (IMSL MATH/LIBARY) in single precision) would yield random deviates from a two-parameter Weibull distribution.

```
CALL RNWIB (NR, A, R)
CALL SSCAL (NR, B, R, 1)
```

The Rayleigh distribution with probability density function,

$$r(x) = \frac{1}{\alpha^2} x e^{-\left(x^2/2\alpha^2\right)} \quad \text{for } x \geq 0$$

is the same as a Weibull distribution with shape parameter **A** equal to 2 and scale parameter *B* equal to

$$\sqrt{2}\,\alpha$$

hence, **RNWIB** and **SSCAL** (or simple multiplication) can be used to generate Rayleigh deviates.

Comments

1. Informational error

Type	Code	Description
3	1	The value of A is so small that the proportion of values from the Weibull that are too large to represent is greater than machine epsilon.

2. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNWIB** is used to generate five pseudorandom deviates from a two-parameter Weibull distribution with shape parameter equal to 2.0 and scale parameter equal to 6.0, a Rayleigh distribution with parameter

$$\alpha = 3\sqrt{2}$$

```
      USE RNWIB_INT
      USE UMACH_INT
      USE RNSET_INT
      USE SSCAL_INT

      IMPLICIT   NONE
      INTEGER    ISEED, NOUT, NR
      REAL       A, B, R(5)
```

```
!
      CALL UMACH (2, NOUT)
      A     = 2.0
      B     = 6.0
      NR    = 5
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNWIB (A, R)
      CALL SSCAL (NR, B, R, 1)
      WRITE (NOUT,99999) R
99999 FORMAT ('      Weibull(2,6) random deviates: ', 5F8.4)
      END
```

Output

```
 Weibull(2,6) random deviates:   1.1122  6.9568  3.0959  4.5031  2.4638
```

RNCOR

Generates a pseudorandom orthogonal matrix or a correlation matrix.

Required Arguments

A — N by N random orthogonal matrix. (Output, if IOPT = 0; workspace if IOPT = 1; Input/Output, if IOPT = 2. If IOPT = 2, A is destroyed.)

Optional Arguments

N — The order of the matrices to be generated. (Input)
N must be at least two.
Default: N = size (A,2).

IOPT — Option indicator. (Input)
Default: IOPT = 0.

IOPT	Action
0	A random orthogonal matrix is generated in A.
1	A random correlation matrix is generated in COR. (A is used as workspace.)
2	A random correlation matrix is generated in COR using the orthogonal matrix input in A.

EV — If IOPT = 1 or 2, a vector of length N containing the eigenvalues of the correlation matrix to be generated. (Input, if IOPT = 1 or 2; not used otherwise.)
The elements of EV must be positive, they must sum to N, and they cannot all be equal.

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

COR — N by N random correlation matrix. (Output, if IOPT = 1 or 2; not used otherwise.)

LDCOR — Leading dimension of COR exactly as specified in the dimension statement in the calling program. (Input)
Default: LDCOR=size(COR, 1)

FORTRAN 90 Interface

Generic: CALL RNCOR (A [, ...])

Specific: The specific interface names are S_RNCOR and D_RNCOR.

FORTRAN 77 Interface

Single: CALL RNCOR (N, IOPT, EV, A, LDA, COR, LDCOR)

Double: The double precision name is DRNCOR.

Description

Routine RNCOR generates a pseudorandom orthogonal matrix A from the invariant Haar measure. For each column of A, a random vector from a uniform distribution on a hypersphere is selected and then is projected onto the orthogonal complement of the columns of A already formed. The method is described by Heiberger (1978). (See also Tanner and Thisted 1982.)

A correlation matrix is formed by applying a sequence of planar rotations to the matrix $\mathbf{A}^T D\mathbf{A}$, where D = diag(EV(1), ..., EV(N)), so as to yield ones along the diagonal. The planar rotations are applied in such an order that in the two by two matrix that determines the rotation, one diagonal element is less than 1.0 and one is greater than 1.0. This method is discussed by Bendel and Mickey (1978) and by Lin and Bendel (1985).

The distribution of the correlation matrices produced by this method is not known. Bendel and Mickey (1978) and Johnson and Welch (1980) discuss the distribution.

For larger matrices, rounding can become severe; and the double precision results may differ significantly from single precision results.

Comments

1. Workspace may be explicitly provided, if desired, by use of R2COR/DR2COR. The reference is:

   CALL R2COR (N, IOPT, EV, A, LDA, COR, LDCOR, IWK, WK)

   The additional arguments are as follows:

   IWK — Work vector of length 3 * N.

   WK — Work vector of length N.

2. Informational error

Type	Code	Description
3	1	Considerable loss of precision occurred in the rotations used to form the correlation matrix. Some of the diagonals of COR differ from 1.0 by more than the machine epsilon.

3. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNCOR is used to generate a 4 by 4 pseudorandom correlation matrix with eigenvalues in the ratio 1:2:3:4. (Note that the eigenvalues must sum to 4.) Routines MXTXF (IMSL MATH/LIBRARY) and EVCSF (IMSL MATH/LIBRARY) are used to check the output.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    I, IOPT, ISEED, J, LDA, LDCOR, N, NOUT
      REAL       A(4,4), COR(4,4), EV(4), EVAL(4), EVEC(4,4), FLOAT, &
                 SUM, XID(4,4)
      INTRINSIC  FLOAT
!
      CALL UMACH (2, NOUT)
      N     = 4
      LDA   = 4
      LDCOR = 4
      EV(1) = 1.0
      EV(2) = 2.0
      EV(3) = 3.0
      EV(4) = 4.0
!                                Scale the eigenvalues to sum to N.
      SUM = SSUM(N,EV,1)
      CALL SSCAL (N, FLOAT(N)/SUM, EV, 1)
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Generate an orthogonal matrix.
      CALL RNCOR (A)
      WRITE (NOUT,99996) ((A(I,J),J=1,N),I=1,N)
99996 FORMAT (' A random orthogonal matrix: ', /, (5X,4F8.4))
!                                Check it for orthogonality.
      CALL MXTXF (A, XID)
      WRITE (NOUT,99997) ((XID(I,J),J=1,N),I=1,N)
99997 FORMAT (' The identity matrix?:       ', /, (5X,4F8.4))
!
!                                Now get a correlation matrix using
!                                the orthogonal matrix in A, which
!                                will be destroyed.
      IOPT = 2
      CALL RNCOR (A,IOPT=IOPT, EV=EV, COR=COR)
      WRITE (NOUT,99998) ((COR(I,J),J=1,N),I=1,N)
99998 FORMAT (' A random correlation matrix: ', /, (5X,4F8.4))
!                                Check the eigenvalues.
```

```
      CALL EVCSF (COR, EVAL, EVEC)
      WRITE (NOUT,99999) (EVAL(I),I=1,N)
99999 FORMAT (' The computed eigenvalues:', 4F8.4)
      END
```

Output

```
 A random orthogonal matrix:
       -0.8804 -0.2417  0.4065 -0.0351
        0.3088 -0.3002  0.5520  0.7141
       -0.3500  0.5256 -0.3874  0.6717
       -0.0841 -0.7584 -0.6165  0.1941
 The identity matrix?:
        1.0000  0.0000  0.0000  0.0000
        0.0000  1.0000  0.0000  0.0000
        0.0000  0.0000  1.0000  0.0000
        0.0000  0.0000  0.0000  1.0000
 A random correlation matrix:
        1.0000 -0.2358 -0.3258 -0.1101
       -0.2358  1.0000  0.1906 -0.0172
       -0.3258  0.1906  1.0000 -0.4353
       -0.1101 -0.0172 -0.4353  1.0000
 The computed eigenvalues:  1.6000  1.2000  0.8000  0.4000
```

RNDAT

Generates pseudorandom numbers from a multivariate distribution determined from a given sample.

Required Arguments

X — NSAMP by K matrix containing the given sample. (Input/Output)
If IDO = 0 or 1, on output the rows of X are rearranged by routine QUADT to form a *k*-*d* tree.

NN — Number of nearest neighbors of the randomly selected point in X that are used to form the output point in R. (Input)

R — NR by K matrix containing the random multivariate vectors in its rows. (Output)

Optional Arguments

IDO — Generator option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNDAT with the sample in X and all desired pseudo-random numbers are to be generated in this call.
1	This is the first invocation, and additional calls to RNDAT will be made to generate additional random numbers using the same given sample.
2	This is an intermediate invocation of RNDAT. The work vectors have been set up in a previous call, but they are not to be released because additional calls will be made.
3	This is the final invocation of RNDAT. The work vectors have been set up in a previous call and they are to be released.

NR — Number of random multivariate vectors to generate. (Input)
If NR = 0, only initialization or wrap up operations are performed. (This would make sense only if IDO = 1 or 3.)
Default: NR = size (R,1).

K — The length of the multivariate vectors, that is, the number of dimensions. (Input)
Default: K = size (R,2).

NSAMP — Number of given data points from the distribution to be simulated. (Input)
Default: NSAMP = size (X,1).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

LDR — Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNDAT (X, NN, R [, ...])
Specific: The specific interface names are S_RNDAT and D_RNDAT.

FORTRAN 77 Interface

Single: CALL RNDAT (IDO, NR, K, NSAMP, X, LDX, NN, R, LDR)
Double: The double precision name is DRNDAT.

Description

Given a sample of size *n* (= NSAMP) of observations of a *k*-variate random variable, RNDAT generates a pseudo-random sample with approximately the same moments as the given sample. The sample obtained is essentially the same as if sampling from a Gaussian kernel estimate of the sample density. (See Thompson 1989.) Routine RNDAT uses methods described by Taylor and Thompson (1986).

Assume that the (vector-valued) observations x_i are in the rows of *X*. An observation, x_j, is chosen randomly; its nearest *m* (= NN) neighbors,

$$x_{j_1}, x_{j_2}, \ \ldots, x_{j_m}$$

are determined; and the mean

$$\overline{x}_j$$

of those nearest neighbors is calculated. Next, a random sample

$u_1, u_2, \ldots, u_m$ is generated from a uniform distribution with lower bound

$$\frac{1}{m} - \sqrt{\frac{3\left(m-1\right)}{m^2}}$$

and upper bound

$$\frac{1}{m} + \sqrt{\frac{3(m-1)}{m^2}}$$

The random variate delivered is

$$\sum_{l=1}^{m} u_l \left(x_{jl} - \overline{x}_j \right) + \overline{x}_j$$

The process is then repeated until **NR** such simulated variates are generated and stored in the rows of **R**.

When **RNDAT** is invoked for the first time for a given sample, a search tree is computed for the rows of **X**. During the generation process, this tree is used to find the nearest neighbors of the randomly selected row. The argument **IDO** is used to determine whether or not the tree must be computed and whether workspace has to be allocated to store the tree.

Comments

1. Workspace may be explicitly provided, if desired, by use of **R2DAT/DR2DAT**. The reference is:

   CALL R2DAT (IDO, NR, K, NSAMP, X, LDX, NN, R, LDR, IWK, WK)

   The additional arguments are as follows:

   IWK — Work vector of length equal to 2 * NSAMP + 3 * LEN + K + NN.

   WK — Work vector of length equal to 2 * NSAMP + 2 * K * LEN + K + 2 * NN.

   R2DAT allows alternating calls for two different populations (see Comment 3). **Warning:** **R2DAT** does no error checking.

2. The rows of **X** are rearranged on output from either **RNDAT** or **R2DAT**.

3. When more than one call is to be made to **RNDAT** to generate more than one **R** matrix using the same sample in **X**, **IDO** should be set to 1 for the first call, to 2 for all subsequent calls except the last one, and to 3 for the last call. If more than one population is to be simulated (that is, there is more than one sample, **X**), it is necessary to generate all of the observations from each population at one time because data is stored in the work vectors. If the user provides work vectors for each population to be simulated, **R2DAT** can be used to simulate different population alternatively.

4. The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNDAT** is used to generate 5 pseudorandom vectors of length 4 using the initial and final systolic pressure and the initial and final diastolic pressure from Data Set A in Afifi and Azen (1979) as the fixed sample from the population to be modeled. (Values of these four variables are in the seventh, tenth, twenty-first, and twenty-fourth columns of data set number nine in routine GDATA).

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDR, LDRDAT, LDX, NDR, NDRDAT, NDX
      PARAMETER  (LDR=5, LDRDAT=113, LDX=113, NDR=4, NDRDAT=34, NDX=4)
!
      INTEGER    ISEED, NN, NRCOL, NRROW
      REAL       R(LDR,NDR), RDATA(LDRDAT,NDRDAT), X(LDX,NDX)
      CHARACTER * 6 NUMBER(1)
!                                Afifi and Azen Data Set A
      DATA NUMBER(1)/'NUMBER'/
      CALL GDATA (9, RDATA, NRROW, NRCOL)
      CALL SCOPY (NRROW, RDATA(1:,7), 1,  X(1:,1), 1)
      CALL SCOPY (NRROW, RDATA(1:,10), 1, X(1:,2), 1)
      CALL SCOPY (NRROW, RDATA(1:,21), 1, X(1:,3), 1)
      CALL SCOPY (NRROW, RDATA(1:,24), 1, X(1:,4), 1)
!
      ISEED = 123457
      CALL RNSET (ISEED)
!                                Set input values
      NN    = 5
!                                Generate random variates
      CALL RNDAT (X, NN, R)
!                                Print results
      CALL WRRRL ('Random variates', R, NUMBER, NUMBER, FMT='(F15.4)')
!
      END
```

Output

```
                          Random variates
                1               2               3               4
1         162.7668         90.5057        153.7173        104.8768
2         153.3533         78.3180        176.6643         85.2155
3          93.6958         48.1675        153.5495         71.3688
4         101.7508         54.1855        113.1215         56.2916
5          91.7403         58.7684         48.4368         28.0994
```

RNMTN

Generates pseudorandom numbers from a multinomial distribution.

Required Arguments

N — Multinomial parameter indicating the number of independent trials. (Input)

P — Vector of length K containing the probabilities of the possible outcomes. (Input)
The elements of P must be positive and must sum to 1.0.

IR — NR by K matrix containing the random multinomial vectors in its rows. (Output)

Optional Arguments

NR — Number of random multinomial vectors to generate. (Input)
Default: NR = size (IR,1).

K — The number of mutually exclusive outcomes on any trial. (Input)
K is the length of the multinomial vectors. K must be greater than or equal to 2.
Default: K = size (IR,2).

LDIR — Leading dimension of IR exactly as specified in the dimension statement of the calling program. (Input)
Default: LDIR = size (IR,1).

FORTRAN 90 Interface

Generic: CALL RNMTN (N, P, IR [, ...])
Specific: The specific interface name is S_RNMTN.

FORTRAN 77 Interface

Single: CALL RNMTN (NR, N, K, P, IR, LDIR)

Description

Routine **RNMTN** generates pseudorandom numbers from a **K**-variate multinomial distribution with parameters **N** and **P**. **K** and **N** must be positive. Each element of **P** must be positive and the elements must sum to 1. The probability function (with n = **N**, k = **K**, and p_i = **P(I)**) is

$$f\left(x_1, x_2, \ldots, x_k\right) = \frac{n!}{x_1! x_2! \ \ldots \ x_k!} p_1^{x_1} p_2^{x_2} \ \ldots \ p_k^{x_k}$$

for $x_i \geq 0$ and

$$\sum_{i=1}^{k} x_i = n$$

The deviate in each row of **IR** is produced by generation of the binomial deviate x_1 with parameters n and p_i and then by successive generations of the conditional binomial deviates x_j given $x_1, x_2, \ldots, x_{j-1}$ with parameters $n - x_1 - x_2 - \ldots - x_{j-1}$ and $p_j / (1 - p_1 - p_2 - \ldots - p_{j-1})$.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNMTN** is used to generate five pseudorandom 3-dimensional multinomial variates with parameters **N** = 20 and **P** = (0.1, 0.3, 0.6).

```
          USE RNMTN_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    K, LDIR
          PARAMETER  (K=3, LDIR=5)
    !
          INTEGER    I, IR(LDIR,K), ISEED, J, N, NOUT, NR
          REAL       P(K)
    !
          CALL UMACH (2, NOUT)
          N     = 20
          NR    = 5
          P(1)  = 0.1
          P(2)  = 0.3
          P(3)  = 0.6
```

```
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNMTN (N, P, IR)
      WRITE (NOUT,99999) ((IR(I,J),J=1,K),I=1,NR)
99999 FORMAT (' Multinomial random deviates: ', 3I4, /, (30X,3I4))
      END
```

Output

```
 Multinomial random deviates:    5   4  11
                                 3   6  11
                                 3   3  14
                                 5   5  10
                                 4   5  11
```

RNMVN

Generates pseudorandom numbers from a multivariate normal distribution.

Required Arguments

RSIG — Upper triangular matrix, K by K, containing the Cholesky factor of the variance-covariance matrix. (Input)
The variance-covariance matrix is equal to the product of the transpose of RSIG and RSIG. RSIG can be obtained from the variance-covariance matrix using routine CHFAC.

R — NR by K matrix containing the random multivariate normal vectors in its rows. (Output)

Optional Arguments

NR — Number of random multivariate normal vectors to generate. (Input)
Default: NR = size (R,1).

K — Length of the multivariate normal vectors. (Input)
Default: K = size (R,2).

LDRSIG — Leading dimension of RSIG exactly as specified in the dimension statement in the calling program. (Input)
Default: LDRSIG = size (RSIG,1).

LDR — Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: CALL RNMVN (RSIG, R [, ...])
Specific: The specific interface names are S_RNMVN and D_RNMVN.

FORTRAN 77 Interface

Single: CALL RNMVN (NR, K, RSIG, LDRSIG, R, LDR)
Double: The double precision name is DRNMVN.

Description

Routine **RNMVN** generates pseudorandom numbers from a multivariate normal distribution with mean vector consisting of all zeroes and variance-covariance matrix whose Cholesky factor (or "square root") is **RSIG**; that is, **RSIG** is an upper triangular matrix such that the transpose of **RSIG** times **RSIG** is the variance-covariance matrix. First, independent random normal deviates with mean 0 and variance 1 are generated, and then the matrix containing these deviates is postmultiplied by **RSIG**. The independent normals are generated into the columns of a matrix, which has **NR** rows; hence, if **RNSET** is called with different values of **NR**, the output is different even if the seed is the same in the calls.

Deviates from a multivariate normal distribution with means other than zero can be generated by using **RNMVN** and then by adding the vector of means to each row of **R**.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, **RNMVN** is used to generate five pseudorandom multivariate normal vectors of length 2 with variance-covariance matrix equal to

$$\begin{matrix} 0.500 & 0.375 \\ 0.375 & 0.500 \end{matrix}$$

The routine **CHFAC** is first called to compute the Cholesky factorization of the variance-covariance matrix.

```
          USE RNMVN_INT
          USE UMACH_INT
          USE CHFAC_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    I, IRANK, ISEED, J, K, LDR, LDRSIG, NOUT, NR
          REAL       COV(2,2), R(5,2), RSIG(2,2)
     !
          CALL UMACH (2, NOUT)
          NR       = 5
          K        = 2
          LDRSIG   = 2
          LDR      = 5
          COV(1,1) = 0.5
          COV(1,2) = 0.375
          COV(2,1) = 0.375
          COV(2,2) = 0.5
     !                                Obtain the Cholesky factorization.
```

```
      CALL CHFAC (COV, IRANK, RSIG)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNMVN (RSIG, R)
      WRITE (NOUT,99999) ((R(I,J),J=1,K),I=1,NR)

99999 FORMAT ('    Multivariate normal random deviates: ', /, &
            (1X,2F8.4))
      END
```

Output

```
       Multivariate normal random deviates:
     1.4507  1.2463
     0.7660 -0.0429
     0.0584 -0.6692
     0.9035  0.4628
    -0.8669 -0.9334
```

RNSPH

Generates pseudorandom points on a unit circle or K-dimensional sphere.

Required Arguments

Z — NR by K matrix containing the random Cartesian coordinates on the unit circle or sphere. (Output)

Optional Arguments

NR — Number of random numbers to generate. (Input)
Default: NR = size (Z,1).

K — Dimension of the circle (K = 2) or of the sphere. (Input)
Default: K = size (Z,2).

LDZ — Leading dimension of Z exactly as specified in the dimension statement of the calling program. (Input)
Default: LDZ = size (Z,1).

FORTRAN 90 Interface

Generic: CALL RNSPH (Z [, ...])
Specific: The specific interface names are S_RNSPH and D_RNSPH.

FORTRAN 77 Interface

Single: CALL RNSPH (NR, K, Z, LDZ)
Double: The double precision name is DRNSPH.

Description

Routine RNSPH generates pseudorandom coordinates of points that lie on a unit circle or a unit sphere in K-dimensional space. For points on a circle (K = 2), pairs of uniform (– 1, 1) points are generated and accepted only if they fall within the unit circle (the sum of their squares is less than 1), in which case they are scaled so as to lie on the circle.

For spheres in three or four dimensions, the algorithms of Marsaglia (1972) are used. For three dimensions, two independent uniform (– 1, 1) deviates U_1 and U_2 are generated and accepted only if the sum of their squares S_1 is less than 1. Then, the coordinates

$$Z_1 = 2U_1\sqrt{1-S_1}, Z_2 = 2U_2\sqrt{1-S_1}, \quad \text{and } Z_3 = 1-2S_1$$

are formed. For four dimensions, U_1, U_2, and S_1 are produced as described above. Similarly, U_3, U_4, and S_2 are formed. The coordinates are then

$$Z_1 = U_1, Z_2 = U_2, Z_3 = U_3\sqrt{\left(1-S_1\right)/S_2}$$

and

$$Z_4 = U_4\sqrt{\left(1-S_1\right)/S_2}$$

For spheres in higher dimensions, **K** independent normal deviates are generated and scaled so as to lie on the unit sphere in the manner suggested by Muller (1959).

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNSPH** is used to generate two uniform random deviates from the surface of the unit sphere in three space.

```
          USE UMACH_INT
          USE RNSET_INT
          USE RNSPH_INT

          IMPLICIT   NONE
          INTEGER    K, LDZ, NR
          PARAMETER  (K=3, LDZ=2)
    !
          INTEGER    I, ISEED, J, NOUT
          REAL       Z(LDZ,K)
    !
          CALL UMACH (2, NOUT)
          NR    = 2
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNSPH (Z)
          WRITE (NOUT,99999) ((Z(I,J),J=1,K),I=1,NR)
    99999 FORMAT ('      Coordinates of first point: ', 3F8.4, /, &
```

```
                '      Coordinates of second point:', 3F8.4)
          END
```

Output

```

          Coordinates of first point:   0.8893  0.2316  0.3944
          Coordinates of second point:  0.1901  0.0396 -0.9810
```

RNTAB

Generates a pseudorandom two-way table.

Required Arguments

NRTOT — Vector of length NROW containing the row totals. (Input)

NCTOT — Vector of length NCOL containing the column totals. (Input)
The elements of NRTOT and NCTOT must be nonnegative and must sum to the same quantity.

ITAB — NROW by NCOL random matrix with the given row and column totals. (Output)

Optional Arguments

IDO — Generator option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNTAB with these input specifications of the two-way table.
1	This is the first invocation, and additional calls to RNTAB will be made to generate random tables with the same specifications.
2	This is an intermediate invocation of RNTAB. The work vectors have been set up in a previous call, but they are not to be released because additional calls will be made.
3	This is the final invocation of RNTAB. The work vectors have been set up in a previous call and they are to be released.

NROW — Number of rows in the table. (Input)
Default: NROW = size (ITAB,1).

NCOL — Number of columns in the table. (Input)
Default: NCOL = size (ITAB,2).

LDITAB — Leading dimension of ITAB exactly as specified in the dimension statement in the calling program. (Input)
Default: LDITAB = size (ITAB,1).

FORTRAN 90 Interface

Generic: CALL RNTAB (NRTOT, NCTOT, ITAB [, ...])

Specific: The specific interface name is S_RNTAB.

FORTRAN 77 Interface

Single: CALL RNTAB (IDO, NROW, NCOL, NRTOT, NCTOT, ITAB, LDITAB)

Description

Routine RNTAB generates pseudorandom entries for a two-way contingency table with fixed row and column totals. The method depends on the size of the table and the total number of entries in the table. If the total number of entries is less than twice the product of the number of rows and columns, the method described by Boyette (1979) and by Agresti, Wackerly, and Boyette (1979) is used. In this method, a work vector is filled with row indices so that the number of times each index appears equals the given row total. This vector is then randomly permuted and used to increment the entries in each row so that the given row total is attained.

For tables with larger numbers of entries, the method of Patefield (1981) is used. This method can be considerably faster in these cases. The method depends on the conditional probability distribution of individual elements, given the entries in the previous rows. The probabilities for the individual elements are computed starting from their conditional means.

On the first call to RNTAB with a given set of row and column totals, certain checking is done, and the work vector is allocated and initialized. On the final call, the work vector is released. The argument IDO indicates the nature of the call. In a simulation study, RNTAB would typically be called first with IDO = 1, then would be called several times with IDO = 2, and then finally would be called with IDO = 3. If only one table is needed, IDO should be set to 0.

Comments

1. Let IRSUM = the sum of the elements in NRTOT. If IRSUM + 1 is less than 2 * NROW * NCOL, automatic workspace usage is IRSUM; otherwise, automatic workspace usage is 2 * IRSUM + 1 because a different algorithm is used. Workspace may be explicitly provided, if desired, by use of R2TAB. R2TAB allows selection of the algorithm to be used and it allows alternating calls for two different problems (see Comment 3). The reference is:

   CALL R2TAB (IDO, NROW, NCOL, NRTOT, NCTOT, ITAB, LDITAB, IOPT, IRSUM, IWK, WK).

   The additional arguments are as follows:

IOPT — Option indicator. (Input)
If IOPT = 1, Boyette's method is used.
If IOPT = 2, Patefield's method is used.

IRSUM — Sum of the elements in NRTOT. (Output)

IWK — Work vector of length equal to the sum of the elements in NRTOT.

WK — Work vector of length equal to the sum of the elements in NRTOT plus one, used only if IOPT = 2.
WARNING: R2TAB does no error checking.

2. Informational error

Type	Code	Description
3	1	The values of NRTOT and/or of NCTOT are such that the probability distribution of tables is degenerate, that is, only one such table is possible.

3. When more than one table with the same marginal totals is to be generated, IDO should be set to 1 for the first call, to 2 for all subsequent calls except the last one, and to 3 for the last call. If several tables of different sizes or with different marginal totals are to be generated, it is necessary to generate all of each type together because of the data stored in the work vectors. If the user provides work vectors for each type of table to be generated, R2TAB can be used to generate different types of tables alternatively.

4. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNTAB is used to generate a two by three table with row totals 3 and 5, and column totals 2, 4, and 2.

```
          USE RNTAB_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    I, ISEED, ITAB(2,3), IWK, J, NCTOT(3), NOUT,&
                     NRTOT(2), NROW, NCOL
    !
          CALL UMACH (2, NOUT)
          NROW = 2
          NCOL = 3
          NRTOT(1) = 3
          NRTOT(2) = 5
          NCTOT(1) = 2
          NCTOT(2) = 4
          NCTOT(3) = 2
          ISEED    = 123457
```

```
      CALL RNSET (ISEED)
      CALL RNTAB (NRTOT, NCTOT, ITAB)
      WRITE (NOUT,99999) ((ITAB(I,J),J=1,NCOL),I=1,NROW)
99999 FORMAT (' A random contingency table with fixed marginal totals:' &
            , /, (5X,3I5))
      END
```

Output

```
 A random contingency table with fixed marginal totals:
        0    2    1
        2    2    1
```

RNMVGC

Given a Cholesky factorization of a correlation matrix, generates pseudorandom numbers from a Gaussian Copula distribution.

Required Arguments

CHOL — Array of size *n* by *n* containing the upper-triangular Cholesky factorization of the correlation matrix of order *n* where *n* = size(**CHOL**,1). (Input)

R — Array of length *n* containing the pseudorandom numbers from a multivariate Gaussian Copula distribution. (Output)

FORTRAN 90 Interface

Generic: **RNMVGC** (**CHOL**, **R**)

Specific: The specific interface names are **S_RNMVGC** and **D_RNMVGC**.

Description

RNMVGC generates pseudorandom numbers from a multivariate Gaussian Copula distribution which are uniformly distributed on the interval (0,1) representing the probabilities associated with standard normal N(0,1) deviates imprinted with correlation information from input upper-triangular Cholesky matrix **CHOL**. Cholesky matrix **CHOL** is defined as the "square root" of a user-defined correlation matrix, that is **CHOL** is an upper triangular matrix such that the transpose of **CHOL** times **CHOL** is the correlation matrix. First, a length *n* array of independent random normal deviates with mean 0 and variance 1 is generated, and then this deviate array is post-multiplied by Cholesky matrix **CHOL**. Finally, the Cholesky-imprinted random N(0,1) deviates are mapped to output probabilities using the N(0,1) cumulative distribution function (CDF).

Random deviates from arbitrary marginal distributions which are imprinted with the correlation information contained in Cholesky matrix **CHOL** can then be generated by inverting the output probabilities using user-specified inverse CDF functions.

Example: Using Copulas to Imprint and Extract Correlation Information

This example uses subroutine **RNMVGC** to generate a multivariate sequence `gcdevt` whose marginal distributions are user-defined and imprinted with a user-specified input correlation matrix `corrin` and then uses subroutine CANCOR to extract an output canonical correlation matrix `corrout` from this multivariate random sequence.

This example illustrates two useful copula related procedures. The first procedure generates a random multivariate sequence with arbitrary user-defined marginal deviates whose dependence is specified by a user-defined correlation matrix. The second procedure is the inverse of the first: an arbitrary multivariate deviate input sequence is first mapped to a corresponding sequence of empirically derived variates, i.e. cumulative distribution function values representing the probability that each random variable has a value less than or equal to the input deviate. The variates are then inverted, using the inverse standard normal CDF function, to N(0,1) deviates; and finally, a canonical covariance matrix is extracted from the multivariate N(0,1) sequence using the standard sum of products.

This example demonstrates that subroutine **RNMVGC** correctly imbeds the user-defined correlation information into an arbitrary marginal distribution sequence by extracting the canonical correlation from these sequences and showing that they differ from the original correlation matrix by a small relative error, which generally decreases as the number of multivariate sequence vectors increases.

```
          use rnmvgc_int
          use cancor_int
          use anorin_int
          use chiin_int
          use fin_int
          use amach_int
          use rnopt_int
          use rnset_int
          use umach_int
          use chfac_int
          implicit none

          integer, parameter :: lmax=15000, nvar=3
          real corrin(nvar,nvar), tol, chol(nvar,nvar), gcvart(nvar), &
             gcdevt(lmax,nvar), corrout(nvar,nvar), relerr
          integer irank, k, kmax, kk, i, j, nout

          data corrin /&
            1.0, -0.9486832, 0.8164965, &
            -0.9486832, 1.0, -0.6454972, &
            0.8164965, -0.6454972,  1.0/

          call umach (2, nout)

          write(nout,*)
          write(nout,*) "Off-diagonal Elements of Input " // &
             "Correlation Matrix: "
          write(nout,*)
```

```
      do i = 2, nvar
         do j = 1, i-1
            write(nout,'(" CorrIn(",i2,",",i2,") = ", f10.6)') &
               i, j, corrin(i,j)
         end do
      end do

      write(nout,*)
      write(nout,*) "Off-diagonal Elements of Output Correlation " // &
         "Matrices calculated from"
      write(nout,*) "Gaussian Copula imprinted multivariate sequence:"

!     Compute the Cholesky factorization of CORRIN.
      tol=amach(4)
      tol=100.0*tol
      call chfac (corrin, irank, chol, tol=tol)

      kmax = lmax/100
      do kk = 1, 3
         write (*, '(/" # vectors in multivariate sequence:  ", &
             i7/)') kmax

         call rnopt(1)
         call rnset (123457)

         do k = 1, kmax

!     Generate an array of Gaussian Copula random numbers.
            call rnmvgc (chol, gcvart)
            do j = 1, nvar
!     Invert Gaussian Copula probabilities to deviates.
               if (j .eq. 1) then
!     ChiSquare(df=10) deviates:
                  gcdevt(k, j) = chiin(gcvart(j), 10.e0)
               else if (j .eq. 2) then
!     F(dfn=15,dfd=10) deviates:
                  gcdevt(k, j) = fin(gcvart(j), 15.e0, 10.e0)
               else
!     Normal(mean=0,variance=1) deviates:
                  gcdevt(k, j) = anorin(gcvart(j))
               end if
            end do
         end do

!     Extract Canonical Correlation matrix.
         call cancor (gcdevt(:kmax,:), corrout)

         do i = 2, nvar
            do j = 1, i-1
               relerr = abs(1.0 - (corrout(i,j) / corrin(i,j)))
               write(nout,'(" CorrOut(",i2,",",i2,") = ", '// &
                 'f10.6, "; relerr = ", f10.6)') &
                 i, j, corrout(i,j), relerr
            end do
         end do
         kmax = kmax*10
      end do
      end
```

Output

```
 Off-diagonal Elements of Input Correlation Matrix:

  CorrIn( 2, 1) =  -0.948683
  CorrIn( 3, 1) =   0.816496
  CorrIn( 3, 2) =  -0.645497

  Off-diagonal Elements of Output Correlation Matrices calculated from
  Gaussian Copula imprinted multivariate sequence:

  # vectors in multivariate sequence:      150

  CorrOut( 2, 1) =  -0.940215; relerr =   0.008926
  CorrOut( 3, 1) =   0.794511; relerr =   0.026927
  CorrOut( 3, 2) =  -0.616082; relerr =   0.045569

  # vectors in multivariate sequence:     1500

  CorrOut( 2, 1) =  -0.947443; relerr =   0.001308
  CorrOut( 3, 1) =   0.808307; relerr =   0.010031
  CorrOut( 3, 2) =  -0.635650; relerr =   0.015256

  # vectors in multivariate sequence:    15000

  CorrOut( 2, 1) =  -0.948267; relerr =   0.000439
  CorrOut( 3, 1) =   0.817261; relerr =   0.000936
  CorrOut( 3, 2) =  -0.646208; relerr =   0.001101
```

RNMVTC

Given a Cholesky factorization of a correlation matrix, generates pseudorandom numbers from a Student's *t* Copula distribution.

Required Arguments

DF — Degrees of freedom. (Input)
DF must be greater than 2.

CHOL — Array of size *n* by *n* containing the upper-triangular Cholesky factorization of the correlation matrix of order *n*. (Input)

R — Array of length *n* containing the pseudorandom numbers from a multivariate a Student's *t* Copula distribution. (Output)

FORTRAN 90 Interface

Generic: **RNMVTC** (**DF**, **CHOL**, **R**)
Specific: The specific interface names are **S_RNMVTC** and **D_RNMVTC**.

Description

RNMVTC generates pseudorandom numbers from a multivariate Student's *t* Copula distribution which are uniformly distributed on the interval (0,1) representing the probabilities associated with Student's *t* deviates with **DF** degrees of freedom imprinted with correlation information from input upper-triangular Cholesky matrix **CHOL**. Cholesky matrix **CHOL** is defined as the "square root" of a user-defined correlation matrix. That is, **CHOL** is an upper triangular matrix such that the transpose of **CHOL** times **CHOL** is the correlation matrix. First, a length *n* array of independent random normal deviates with mean 0 and variance 1 is generated, and then this deviate array is post-multiplied by Cholesky matrix **CHOL**. Each of the *n* elements of the resulting vector of Cholesky-imprinted random deviates is then divided by $\sqrt{s/\nu}$, where ν = **DF** and *s* is a random deviate taken from a chi-squared distribution with **DF** degrees of freedom. Each element of the Cholesky-imprinted standard normal N(0,1) array is a linear combination of normally distributed random numbers and is therefore itself normal, and the division of each element by $\sqrt{s/\nu}$ therefore insures that each element of the resulting array is Student's *t* distributed. Finally, each element of the Cholesky-imprinted Student's *t* array is mapped to an output probability using the Student's *t* cumulative distribution function (CDF) with **DF** degrees of freedom.

Random deviates from arbitrary marginal distributions which are imprinted with the correlation information contained in Cholesky matrix **CHOL** can then be generated by inverting the output probabilities using user-specified inverse CDF functions.

Example: Using Student's t Copulas to Imprint and Extract Correlation Information

This example uses **RNMVTC** to generate a multivariate sequence **tcdevt** whose marginal distributions are user-defined and imprinted with a user-specified input correlation matrix **corrin** and then uses CANCOR to extract an output canonical correlation matrix **corrout** from this multivariate random sequence.

This example illustrates two useful copula related procedures. The first procedure generates a random multivariate sequence with arbitrary user-defined marginal deviates whose dependence is specified by a user-defined correlation matrix. The second procedure is the inverse of the first: an arbitrary multivariate deviate input sequence is first mapped to a corresponding sequence of empirically derived variates, i.e. cumulative distribution function values representing the probability that each random variable has a value less than or equal to the input deviate. The variates are then inverted, using the inverse standard normal CDF function, to N(0,1) deviates; and finally, a canonical covariance matrix is extracted from the multivariate N(0,1) sequence using the standard sum of products.

This example demonstrates that subroutine **RNMVTC** correctly imbeds the user-defined correlation information into an arbitrary marginal distribution sequence by extracting the canonical correlation from these sequences and showing that they differ from the original correlation matrix by a small relative error.

Recall that a Gaussian Copula array sequence, whose probabilities are mapped directly from Cholesky-imprinted N(0,1) deviates, has the property that the relative error between the input and output correlation matrices generally decreases as the number of multivariate sequence vectors increases. This is understandable because the correlation imprinting and extraction processes both act upon N(0,1) marginal distributions, and one would expect that a larger sample would therefore result in more accurate imprinting and extraction of correlation information.

In contrast, the imprinting of correlation information onto Student's *t* vector sequence is accomplished by imprinting onto an N(0,1) array and then dividing the array components by a scaled chi-squared random deviate, thereby introducing noise into the imprinting process. (An array of Student's *t* deviates cannot be Cholesky-imprinted directly, because a linear combination of Student's *t* deviates is not Student's *t* distributed.) A larger sample would thus contain additional correlation information and additional noise, so the accuracy would be expected to plateau. This is illustrated in the following example, which should be compared with the Gaussian Copula example given for FNL routine RNMVGC.

```
          use rnmvtc_int
          use cancor_int
          use anorin_int
```

```
      use chiin_int
      use fin_int
      use amach_int
      use rnopt_int
      use rnset_int
      use umach_int
      use chfac_int
      implicit none

      integer, parameter:: lmax=15000, nvar=3
      real corrin(nvar,nvar), tol, chol(nvar,nvar), &
         tcvart(nvar), tcdevt(lmax,nvar), corrout(nvar,nvar), &
         relerr, df
      integer irank, k, kmax, kk, i, j, nout

      data corrin /&
        1.0, -0.9486832, 0.8164965, &
        -0.9486832, 1.0, -0.6454972, &
        0.8164965, -0.6454972,  1.0/

      call umach (2, nout)

      write(nout,*) "Off-diagonal Elements of Input " // &
         "Correlation Matrix: "
      write(nout,*)

      do i = 2, nvar
         do j = 1, i-1
            write(nout,'(" CorrIn(",i2,",",i2,") = ", f10.6)') &
               i, j, corrin(i,j)
         end do
      end do

      df = 5.0
      write(nout,'(/" Degrees of freedom df = ", f6.2/)') df
      write(nout,*) "Imprinted random sequences distributions:"
      write(nout,*) "1: Chi, 2: F, 3: Normal:"

      write(nout,*)
      write(nout,*) "Off-diagonal Elements of Output Correlation " //&
         "Matrices calculated from"
      write(nout,*) "Student's t Copula imprinted multivariate sequence:"

!     Compute the Cholesky factorization of CORRIN.
      tol=amach(4)
      tol=100.0*tol
      call chfac (corrin, irank, chol, tol=tol)

      kmax = lmax/100
      do kk = 1, 3
         write (nout, '(/" # vectors in multivariate sequence:  ", &
             i7/)') kmax

         call rnopt(1)
         call rnset (123457)

         do k = 1, kmax
!     Generate an array of Gaussian Copula random numbers.
            call rnmvtc (df, chol, tcvart)

```

```
            do j = 1, nvar
!     Invert Student's t Copula probabilities to deviates.

               if (j .eq. 1) then
!     ChiSquare(df=10) deviates:
                  tcdevt(k, j) = chiin(tcvart(j), 10.e0)
               else if (j .eq. 2) then
!     F(dfn=15,dfd=10) deviates:
                  tcdevt(k, j) = fin(tcvart(j), 15.e0, 10.e0)
               else
!     Normal(mean=0,variance=1) deviates:
                  tcdevt(k, j) = anorin(tcvart(j))
               end if
            end do
         end do
!     Extract Canonical Correlation matrix.
         call cancor (tcdevt(:kmax,:), corrout)

         do i = 2, nvar
            do j = 1, i-1
               relerr = abs(1.0 - (corrout(i,j) / corrin(i,j)))
               write(nout,'(" CorrOut(",i2,",",i2,") = ",&
                 f10.6, "; relerr = ", f10.6)')&
                 i, j, corrout(i,j), relerr
            end do
         end do
         kmax = kmax*10
      end do
      end
```

Output

```
 Off-diagonal Elements of Input Correlation Matrix:

 CorrIn( 2, 1) =  -0.948683
 CorrIn( 3, 1) =   0.816496
 CorrIn( 3, 2) =  -0.645497

 Degrees of freedom df =   5.00

 Imprinted random sequences distributions:
 1: Chi, 2: F, 3: Normal:

 Off-diagonal Elements of Output Correlation Matrices calculated from
 Student's t Copula imprinted multivariate sequence:

 # vectors in multivariate sequence:      150

 CorrOut( 2, 1) =  -0.953573; relerr =   0.005154
 CorrOut( 3, 1) =   0.774720; relerr =   0.051166
 CorrOut( 3, 2) =  -0.621419; relerr =   0.037302

 # vectors in multivariate sequence:     1500

 CorrOut( 2, 1) =  -0.944316; relerr =   0.004603
 CorrOut( 3, 1) =   0.810163; relerr =   0.007757
 CorrOut( 3, 2) =  -0.636348; relerr =   0.014174
```

```
     # vectors in multivariate sequence:    15000

     CorrOut( 2, 1) =  -0.946770; relerr =   0.002017
     CorrOut( 3, 1) =   0.808562; relerr =   0.009718
     CorrOut( 3, 2) =  -0.636322; relerr =   0.014215
```

CANCOR

Given an input array of deviate values, generates a canonical correlation array.

Required Arguments

DEVT — Array of size ***m*** by ***n*** containing ***m*** sequence elements for each of ***n*** variables. (Input)

CORR — Array of size ***n*** by ***n*** containing canonical correlation array. (Output)

FORTRAN 90 Interface

Generic: CANCOR(**DEVT**, **CORR**)

Specific: The specific interface names are **S_CANCOR** and **D_CANCOR**.

Description

CANCOR generates a canonical correlation matrix from an arbitrarily distributed multivariate deviate sequence **DEVT** with *n* deviate variables, *m* elements in each deviate sequence, and a Gaussian Copula dependence structure.

Subroutine **CANCOR** first maps each of the **J=1**…*n* input deviate sequences **DEVT(K=1**…*m*, **J)** into a corresponding sequence of variates, say V_{KJ} (where variates are values of the empirical cumulative probability function, *CDF*(*x*), defined as the probability that random deviate variable $X < x$). The variate matrix element V_{KJ} is then mapped into standard normal N(0,1) distributed deviates z_{kj} using the inverse standard normal CDF **ANORIN**(V_{KJ}) and then the standard covariance estimator

$$C_{ij} = \frac{1}{m}\sum_{k=1}^{m} z_{ki} z_{kj}$$

is used to calculate the canonical correlation matrix **CORR**, where C_{ij} = **CORR**(**I**,**J**).

If a multivariate distribution has Gaussian marginal distributions, then the standard "empirical" correlation matrix given above is "unbiased", i.e. an accurate measure of dependence among the variables. But when the marginal distributions depart significantly from Gaussian, i.e. are skewed or flattened, then the empirical correlation may become biased. One way to remove such bias from dependence measures is to map the non-Gaussian-distributed marginal deviates to N(0,1) deviates (by mapping the non-Gaussian marginal deviates to empirically derived marginal CDF variate values, then inverting the variates to N(0,1) deviates as described above), and calculating the

standard empirical correlation matrix from these N(0,1) deviates as in the equation above. The resulting "canonical correlation" matrix thereby avoids the bias that would occur if the empirical correlation matrix were extracted from the non-Gaussian marginal distributions directly.

The canonical correlation matrix may be of value in such applications as Markowitz portfolio optimization, where an unbiased measure of dependence is required to evaluate portfolio risk, defined in terms of the portfolio variance which is in turn defined in terms of the correlation among the component portfolio instruments.

The utility of the canonical correlation derives from the observation that a "copula" multivariate distribution with uniformly-distributed deviates (corresponding to the CDF probabilities associated with the marginal deviates) may be mapped to arbitrarily distributed marginals, so that an unbiased dependence estimator derived from one set of marginals (N(0,1) distributed marginals) can be used to represent the dependence associated with arbitrarily-distributed marginals. The "Gaussian Copula" (whose variate arguments are derived from N(0,1) marginal deviates) is a particularly useful structure for representing multivariate dependence.

Example: Using Copulas to Imprint and Extract Correlation Information

This example uses subroutine `RNMVGC` to generate a multivariate sequence `gcdevt` whose marginal distributions are user-defined and imprinted with a user-specified input correlation matrix `corrin` and then uses subroutine **CANCOR** to extract an output canonical correlation matrix `corrout` from this multivariate random sequence.

This example illustrates two useful copula related procedures. The first procedure generates a random multivariate sequence with arbitrary user-defined marginal deviates whose dependence is specified by a user-defined correlation matrix. The second procedure is the inverse of the first: an arbitrary multivariate deviate input sequence is first mapped to a corresponding sequence of empirically derived variates, i.e. cumulative distribution function values representing the probability that each random variable has a value less than or equal to the input deviate. The variates are then inverted, using the inverse standard normal CDF function, to N(0,1) deviates; and finally, a canonical covariance matrix is extracted from the multivariate N(0,1) sequence using the standard sum of products.

This example demonstrates that subroutine **RNMVGC** correctly imbeds the user-defined correlation information into an arbitrary marginal distribution sequence by extracting the canonical correlation from these sequences and showing that they differ from the original correlation matrix by a small relative error, which generally decreases as the number of multivariate sequence vectors increases.

```
          use rnmvgc_int
          use cancor_int
          use anorin_int
          use chiin_int
          use fin_int
          use amach_int
```

```
      use rnopt_int
      use rnset_int
      use umach_int
      use chfac_int
      implicit none

      integer, parameter :: lmax=15000, nvar=3
      real corrin(nvar,nvar), tol, chol(nvar,nvar), gcvart(nvar), &
         gcdevt(lmax,nvar), corrout(nvar,nvar), relerr
      integer irank, k, kmax, kk, i, j, nout

      data corrin /&
        1.0, -0.9486832, 0.8164965, &
        -0.9486832, 1.0, -0.6454972, &
        0.8164965, -0.6454972,  1.0/

      call umach (2, nout)

      write(nout,*)
      write(nout,*) "Off-diagonal Elements of Input " // &
         "Correlation Matrix: "
      write(nout,*)

      do i = 2, nvar
         do j = 1, i-1
            write(nout,'(" CorrIn(",i2,",",i2,") = ", f10.6)') &
               i, j, corrin(i,j)
         end do
      end do

      write(nout,*)
      write(nout,*) "Off-diagonal Elements of Output Correlation " // &
         "Matrices calculated from"
      write(nout,*) "Gaussian Copula imprinted multivariate sequence:"

!     Compute the Cholesky factorization of CORRIN.
      tol=amach(4)
      tol=100.0*tol
      call chfac (corrin, irank, chol, tol=tol)

      kmax = lmax/100
      do kk = 1, 3
         write (*, '(/" # vectors in multivariate sequence:  ", &
             i7/)') kmax

         call rnopt(1)
         call rnset (123457)

         do k = 1, kmax

!     Generate an array of Gaussian Copula random numbers.
            call rnmvgc (chol, gcvart)
            do j = 1, nvar
!     Invert Gaussian Copula probabilities to deviates.
               if (j .eq. 1) then
!     ChiSquare(df=10) deviates:
                  gcdevt(k, j) = chiin(gcvart(j), 10.e0)
               else if (j .eq. 2) then
!     F(dfn=15,dfd=10) deviates:
                  gcdevt(k, j) = fin(gcvart(j), 15.e0, 10.e0)
```

```
                  else
!     Normal(mean=0,variance=1) deviates:
                     gcdevt(k, j) = anorin(gcvart(j))
                  end if
               end do
            end do

!     Extract Canonical Correlation matrix.
            call cancor (gcdevt(:kmax,:), corrout)

            do i = 2, nvar
               do j = 1, i-1
                  relerr = abs(1.0 - (corrout(i,j) / corrin(i,j)))
                  write(nout,'(" CorrOut(",i2,",",i2,") = ", '// &
                    'f10.6, "; relerr = ", f10.6)') &
                    i, j, corrout(i,j), relerr
               end do
            end do
            kmax = kmax*10
         end do
         end
```

Output

```
    Off-diagonal Elements of Input Correlation Matrix:

     CorrIn( 2, 1) =  -0.948683
     CorrIn( 3, 1) =   0.816496
     CorrIn( 3, 2) =  -0.645497

     Off-diagonal Elements of Output Correlation Matrices calculated from
     Gaussian Copula imprinted multivariate sequence:

     # vectors in multivariate sequence:      150

     CorrOut( 2, 1) =  -0.940215; relerr =   0.008926
     CorrOut( 3, 1) =   0.794511; relerr =   0.026927
     CorrOut( 3, 2) =  -0.616082; relerr =   0.045569

     # vectors in multivariate sequence:     1500

     CorrOut( 2, 1) =  -0.947443; relerr =   0.001308
     CorrOut( 3, 1) =   0.808307; relerr =   0.010031
     CorrOut( 3, 2) =  -0.635650; relerr =   0.015256

     # vectors in multivariate sequence:    15000

     CorrOut( 2, 1) =  -0.948267; relerr =   0.000439
     CorrOut( 3, 1) =   0.817261; relerr =   0.000936
     CorrOut( 3, 2) =  -0.646208; relerr =   0.001101
```

RNNOS

Generates pseudorandom order statistics from a standard normal distribution.

Required Arguments

IFIRST — First order statistic to generate. (Input)

ILAST — Last order statistic to generate. (Input)
`ILAST` must be greater than or equal to `IFIRST`. The full set of order statistics from `IFIRST` to `ILAST` is generated. If only one order statistic is desired, set `ILAST` = `IFIRST`.

N — Size of the sample from which the order statistics arise. (Input)

R — Vector of length `ILAST` + 1 `- IFIRST` containing the random order statistics in ascending order. (Output)
The first element of `R` is the `IFIRST`-th order statistic in a random sample of size `N` from the standard normal distribution.

FORTRAN 90 Interface

Generic: `CALL RNNOS (IFIRST, ILAST, N, R)`
Specific: The specific interface names are `S_RNNOS` and `D_RNNOS`.

FORTRAN 77 Interface

Single: `CALL RNNOS (IFIRST, ILAST, N, R)`
Double: The double precision name is `DRNNOS`.

Description

Routine `RNNOS` generates the `IFIRST` through the `ILAST` order statistics from a pseudorandom sample of size `N` from a normal (0, 1) distribution. Routine `RNNOS` uses the routine `RNUNO` to generate order statistics from the uniform (0, 1) distribution and then obtains the normal order statistics using the inverse CDF transformation.

Each call to `RNNOS` yields an independent event so order statistics from different calls may not have the same order relations with each other.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Example

In this example, RNNOS is used to generate the fifteenth through the nineteenth order statistics from a sample of size twenty.

```
      USE RNNOS_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    IFIRST, ILAST, ISEED, N, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      IFIRST = 15
      ILAST  = 19
      N      = 20
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNNOS (IFIRST, ILAST, N, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  The 15th through the 19th order statistics from a', &
            /, '  random sample of size 20 from a normal distribution' &
            , /, 5F8.4)
      END
```

Output

```
 The 15th through the 19th order statistics from a
 random sample of size 20 from a normal distribution
 0.4056  0.4681  0.4697  0.9067  0.9362
```

RNUNO

Generates pseudorandom order statistics from a uniform (0, 1) distribution.

Required Arguments

IFIRST — First order statistic to generate. (Input)

ILAST — Last order statistic to generate. (Input)
`ILAST` must be greater than or equal to `IFIRST`. The full set of order statistics from `IFIRST` to `ILAST` is generated. If only one order statistic is desired, set `ILAST` = `IFIRST`.

N — Size of the sample from which the order statistics arise. (Input)

R — Vector of length `ILAST` + 1 `- IFIRST` containing the random order statistics in ascending order. (Output)
The first element of `R` is the `IFIRST`-th order statistic in a random sample of size `N` from the uniform (0, 1) distribution.

FORTRAN 90 Interface

Generic: `CALL RNUNO (IFIRST, ILAST, N, R)`
Specific: The specific interface names are `S_RNUNO` and `D_RNUNO`.

FORTRAN 77 Interface

Single: `CALL RNUNO (IFIRST, ILAST, N, R)`
Double: The double precision name is `DRNUNO`.

Description

Routine `RNUNO` generates the `IFIRST` through the `ILAST` order statistics from a pseudorandom sample of size `N` from a uniform (0, 1) distribution. Depending on the values of `IFIRST` and `ILAST`, different methods of generation are used to achieve greater efficiency. If `IFIRST` = 1 and `ILAST` = `N`, that is, if the full set of order statistics are desired, the spacings between successive order statistics are generated as ratios of exponential variates. If the full set is not desired, a beta variate is generated for one of the order statistics, and the others are generated as extreme order statistics from conditional uniform distributions. Extreme order statistics from a uniform distribution can be obtained by raising a uniform deviate to an appropriate power.

Each call to **RNUNO** yields an independent event. This means, for example, that if on one call the fourth order statistic is requested and on a second call the third order statistic is requested, the "fourth" may be smaller than the "third". If both the third and fourth order statistics from a given sample are desired, they should be obtained from a single call to **RNUNO** (by specifying **IFIRST** less than or equal to 3 and **ILAST** greater than or equal to 4).

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNUNO** is used to generate the fifteenth through the nineteenth order statistics from a sample of size twenty.

```
      USE RNUNO_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    IFIRST, ILAST, ISEED, N, NOUT
      REAL       R(5)
!
      CALL UMACH (2, NOUT)
      IFIRST = 15
      ILAST  = 19
      N      = 20
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNUNO (IFIRST, ILAST, N, R)
      WRITE (NOUT,99999) R
99999 FORMAT ('  The 15th through the 19th order statistics from a', &
            /, '  random sample of size 20 from a uniform ', &
            'distribution', /, 5F8.4)
      END
```

Output

```
  The 15th through the 19th order statistics from a
  random sample of size 20 from a uniform distribution
  0.6575  0.6802  0.6807  0.8177  0.8254
```

RNARM

Generates a time series from a specified ARMA model.

Required Arguments

CNST — Overall constant. (Input)
See Comments.

PAR — Vector of length NPAR containing the autoregressive parameters. (Input)

LAGAR — Vector of length NPAR containing the order of the autoregressive parameters. (Input)
The elements of LAGAR must be greater than or equal to one.

PMA — Vector of length NPMA containing the moving average parameters. (Input)

LAGMA — Vector of length NPMA containing the order of the moving average parameters. (Input)
The elements of LAGMA must be greater than or equal to one.

IADIST — Option for normally distributed innovations. (Input)

IADIST	Action
0	Innovations are generated from a normal distribution (white noise) with mean 0 and variance AVAR.
1	Innovations are specifed by the user.

AVAR — Variance of the normal distribution, if used. (Input)
For IADIST = 0, AVAR is input; and for IADIST = 1, AVAR is unused.

A — Vector of length NW + max(LAGMA(*j*)) containing the innovations. (Input or output)
For IADIST = 1, A is input; and for IADIST = 0, A is output.

WI — Vector of length max(LAGAR(*i*)) containing the initial values of the time series. (Input)

W — Vector of length NW containing the generated time series. (Output)

Optional Arguments

NW — Number of observations of the time series to generate. (Input)
NW must be greater than or equal to one.
Default: NW = size (W,1).

NPAR — Number of autoregressive parameters. (Input)
NPAR must be greater than or equal to zero.
Default: NPAR = size (PAR,1).

NPMA — Number of moving average parameters. (Input)
NPMA must be greater than or equal to zero.
Default: NPMA = size (PMA,1).

FORTRAN 90 Interface

Generic: CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, IADIST, AVAR, A, WI, W [, ...])
Specific: The specific interface names are S_RNARM and D_RNARM.

FORTRAN 77 Interface

Single: CALL RNARM (NW, CNST, NPAR, PAR, LAGAR, NPMA, PMA, LAGMA, IADIST, AVAR, A, WI, W)
Double: The double precision name is DRNARM.

Description

Routine RNARM simulates an ARMA(p, q) process, $\{W_t\}$ for t = 1, 2, ..., n (with n = NW, p = NPAR, and q = NPMA). The model is

$$\phi(B)W_t = \theta_0 + \theta(B)A_t \qquad t \in ZZ$$

where B is the backward shift operator,

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q$$

Let μ be the mean of the time series $\{W_t\}$. The overall constant θ_0 (CNST) is

$$\theta_0 = \begin{cases} \mu & p = 0 \\ \mu\left(1 - \sum_{i=1}^{p}\phi_i\right) & p > 0 \end{cases}$$

Comments

1. The time series is generated according to the following model:

$$X(i) = CNST + PAR(1) * X(i - LAGAR(1)) + \ldots + PAR(NPAR) * X(i - LAGAR(NPAR)) + A(i) - PMA(1) * A(i - LAGMA(1)) - \ldots - PMA(NPMA) * A(i - LAGAR(NPMA))$$

where

$$X(t) = W(t), t = 1, 2, \ldots, NW$$

and

$$W(t) = WI(t + p), t = 1 - p, 2 - p, \ldots, -1, 0$$

with $p = \max(LAGAR(k))$.

The constant is related to the mean of the series, **WMEAN**, as follows:

$$CNST = WMEAN * (1 - PAR(1) - \ldots - PAR(NPAR))$$

2. Time series whose innovations have a nonnormal distribution may be simulated by setting **IADIST** = 1 and by providing the appropriate innovations in **A** and start values in **WI**.

3. The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Examples

Example 1

In this example, **RNARM** is used to generate a time series of length five, using an ARMA model with three autoregressive parameters and two moving average parameters. The start values are 0.1000, 0.0500, and 0.0375.

```
      USE RNARM_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    NPAR, NPMA, NW
      PARAMETER  (NPAR=3, NPMA=2, NW=5)
!
      INTEGER    I, IADIST, ISEED, LAGAR(NPAR), LAGMA(NPMA), NOUT
      REAL       A(NW+2), AVAR, CNST, PAR(NPAR), PMA(NPMA), W(NW), &
                 WI(3)
!
      CALL UMACH (2, NOUT)
      LAGAR(1) = 1
      LAGAR(2) = 2
      LAGAR(3) = 3
      PAR(1)   = 0.500
      PAR(2)   = 0.250
      PAR(3)   = 0.125
      LAGMA(1) = 1
      LAGMA(2) = 2
      PMA(1)   = -0.500
      PMA(2)   = -0.250
```

```
      IADIST   = 0
      CNST    = 1.0
      AVAR     = 0.1
      WI(1)    = 0.1
      WI(2)    = 0.05
      WI(3)    = 0.0375
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
                  IADIST, AVAR, A, WI, W)
      WRITE (NOUT,99999) (W(I),I=1,NW)
99999 FORMAT ('   Simulated ARMA(3,2) series ', 5F7.4)
      END
```

Output

```
   Simulated ARMA(3,2) series  1.4033 2.2200 2.2864 2.8878 2.8322
```

Example 2

In this example, 500 observations from an ARMA(2, 2) process are simulated using **RNARM**; and then routine NSPE is used to estimate the parameters of the model. The model is used as an example by Priestley (1981), page 139.

```

      USE RNARM_INT
      USE RNSET_INT
      USE NSPE_INT

      IMPLICIT   NONE
      INTEGER    NPAR, NPMA, NW
      PARAMETER  (NPAR=2, NPMA=2, NW=500)
!
      INTEGER    IADIST, ISEED, LAGAR(NPAR), LAGMA(NPMA)
      REAL       A(NW+2), AVAR, AVAR1, CNST, CNST1, PAR(NPAR), &
                 PAR1(NPAR), PMA(NPMA), PMA1(NPMA), W(NW), WI(2), WMEAN
!
      LAGAR(1) = 1
      LAGAR(2) = 2
      PAR(1)   = -1.4
      PAR(2)   = -0.5
      LAGMA(1) = 1
      LAGMA(2) = 2
      PMA(1)   = 0.2
      PMA(2)   = 0.1
      IADIST   = 0
      CNST    = 0.0
      AVAR     = 1.0
      WI(1)    = 0.0
      WI(2)    = 0.0
      ISEED    = 123457
      CALL RNSET (ISEED)
      CALL RNARM (CNST, PAR, LAGAR, PMA, LAGMA, &
                  IADIST, AVAR, A, WI, W)
      CALL NSPE (W, CNST1, PAR1, PMA1, AVAR1, IPRINT=1)
      END
```

Output

```
 Results from NSPE/N2PE
 WMEAN =     .02192622
 CONST =      .0695866
 AVAR  =    1.0936457
        PAR
       1       2
  -1.533  -0.641
         PMA
        1        2
   0.0560   0.1294
```

RNNPP

Generates pseudorandom numbers from a nonhomogeneous Poisson process.

Required Arguments

TIMBEG — Lower endpoint of the time interval of the process. (Input)
TIMBEG must be nonnegative. Usually, TIMBEG = 0.

TIMEND — Upper endpoint of the time interval of the process. (Input)
TIMEND must be greater than TIMBEG.

FTHETA — User-supplied FUNCTION to provide the value of the rate of the process as a function of time. This function must be defined over the interval from TIMBEG to TIMEND and must be nonnegative in that interval. The form is FTHETA(TIME),
where

TIME — Time at which the rate function is evaluated. (Input)

FTHETA — Value of the rate function. (Output)

FTHETA must be declared EXTERNAL in the calling program.

THEMIN — Minimum value of the rate function FTHETA in the interval (TIMBEG, TIMEND). (Input)
If the actual minimum is unknown, set THEMIN = 0.0.

THEMAX — Maximum value of the rate function FTHETA in the interval (TIMBEG, TIMEND). (Input)
If the actual maximum is unknown, set THEMAX to a known upper bound of the maximum. The efficiency of RNNPP is less the greater THEMAX exceeds the true maximum.

NEUB — Upper bound on the number of events to be generated. (Input)
In order to be reasonably sure that the full process through time TIMEND is generated, calculate NEUB as NEUB = X + 10.0 * SQRT(X), where X = THEMAX * (TIMEND - TIMBEG). The only penalty in setting NEUB too large is that the output vector must be dimensioned of length NEUB.

NE — Number of events actually generated. (Output)
If NE is less that NEUB, the time TIMEND is reached before NEUB events are realized.

R — Vector of length NE containing the times to events. (Output)
R must be dimensioned to be of length NEUB.

FORTRAN 90 Interface

Generic: CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
Specific: The specific interface names are S_RNNPP and D_RNNPP.

FORTRAN 77 Interface

Single: CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
Double: The double precision name is DRNNPP.

Description

Routine RNNPP simulates a one-dimensional nonhomogeneous Poisson process with rate function THETA in a fixed interval (TIMBEG, TIMEND].

Let $\lambda(t)$ be the rate function and t_0 = TIMBEG and t_1 = TIMEND. Routine RNNPP uses a method of thinning a nonhomogeneous Poisson process $\{N^*(t), t \geq t_0\}$ with rate function $\lambda^*(t) \geq \lambda(t)$ in $(t_0, t_1]$, where the number of events, N^*, in the interval $(t_0, t_1]$ has a Poisson distribution with parameter

$$\mu_0 = \int_{t_0}^{t_1} \lambda(t)\, dt$$

The function

$$\Lambda(t') = \int_0^{t'} \lambda(t)\, dt$$

is called the *integrated rate function*. In RNNPP, $\lambda^*(t)$ is taken to be a constant λ^*(= THEMAX) so that at time t_i, the time of the next event t_{i+1} is obtained by generating and cumulating exponential random numbers

$$E^*_{1,i},\ E^*_{2,i},\ \ \ldots\ ,$$

with parameter λ^*, until for the first time

$$u_{j,i} \leq \left(t_i + E^*_{1,i} + \ \cdots\ + E^*_{j,i} \right) / \lambda^*$$

where the $u_{j,i}$ are independent uniform random numbers between 0 and 1. This process is continued until the specified number of events, NEUB, is realized or until the time, TIMEND, is exceeded. This method is due to Lewis and Shedler (1979), who also review other methods. The most straightforward (and most efficient) method is by inverting the integrated rate function, but often this is not possible.

If **THEMAX** is actually greater than the maximum of $\lambda(t)$ in $(t_0, t_1]$, the routine will work, but less efficiently. Also, if $\lambda(t)$ varies greatly within the interval, the efficiency is reduced. In that case, it may be desirable to divide the time interval into subintervals within which the rate function is less variable. This is possible because the process is without memory.

If no time horizon arises naturally, **TIMEND** must be set large enough to allow for the required number of events to be realized. Care must be taken, however, that **FTHETA** is defined over the entire interval.

After simulating a given number of events, the next event can be generated by setting **TIMBEG** to the time of the last event (the sum of the elements in **R**) and calling **RNNPP** again. Cox and Lewis (1966) discuss modeling applications of nonhomogeneous Poisson processes.

Comments

The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.

Examples

Example 1

In this example, **RNNPP** is used to generate the first five events in the time 0 to 20 (if that many events are realized) in a nonhomogeneous process with rate function

$$\lambda(t) = 0.6342\, e^{0.001427t}$$

for $0 < t \leq 20$.

Since this is a monotonically increasing function of t, the minimum is at $t = 0$ and is 0.6342, and the maximum is at $t = 20$ and is $0.6342\, e^{0.02854} = 0.652561$.

```
          USE RNNPP_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    NEUB
          PARAMETER  (NEUB=5)
    !
          INTEGER    I, ISEED, NE, NOUT
          REAL       FTHETA, R(NEUB), THEMAX, THEMIN, TIMBEG, TIMEND
          EXTERNAL   FTHETA
    !
          CALL UMACH (2, NOUT)
          TIMBEG = 0.0
          TIMEND = 20.0
```

```
      THEMIN = 0.6342
      THEMAX = 0.652561
      ISEED  = 123457
      CALL RNSET (ISEED)
      CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
      WRITE (NOUT,99999) NE, (R(I),I=1,NE)
99999 FORMAT ('   Inter-event times for the first ', I1, ' events', /, &
             '   in the process: ', 5F7.4)
      END
!
      REAL FUNCTION FTHETA (T)
      REAL       T
!
      REAL       EXP
      INTRINSIC  EXP
!
      FTHETA = 0.6342*EXP(0.001427*T)
      RETURN
      END
```

Output

```
 Inter-event times for the first 5 events
 in the process:  0.0527 0.4080 0.2584 0.0198 0.1676
```

Example 2

As it turns out in the simulation above, the first five events are realized before time equals 20. If it is desired to continue the simulation to time equals 20, setting NEUB to 49 (that is,

$$\lambda^*\left(t_1 - t_0\right) + 10\sqrt{\lambda^*\left(t_1 - t_0\right)}$$

would likely ensure that the time is reached. In the following example, we see that there are twelve events realized by time equals 20.

```
      USE UMACH_INT
      USE RNSET_INT
      USE RNNPP_INT
      USE SSUM_INT

      IMPLICIT   NONE
      INTEGER    NEUB
      PARAMETER  (NEUB=49)
!
      INTEGER    ISEED, NE, NOUT
      REAL       FTHETA, R(NEUB), T, THEMAX, THEMIN, TIMBEG, TIMEND
      EXTERNAL   FTHETA
!
      CALL UMACH (2, NOUT)
      TIMBEG = 0.0
      TIMEND = 20.0
      THEMIN = 0.6342
      THEMAX = 0.652561
      ISEED  = 123457
      CALL RNSET (ISEED)
```

```
      CALL RNNPP (TIMBEG, TIMEND, FTHETA, THEMIN, THEMAX, NEUB, NE, R)
      T = TIMBEG + SSUM(NE,R,1)
      IF (NE .LT. NEUB) THEN
         WRITE (NOUT,99998) NE, T
99998    FORMAT ('   Only ', I2, ' events occurred before the time', &
               /, '   limit expired.  The last event occurred at', /, &
               '   time = ', F6.3)
      ELSE
         WRITE (NOUT,99999) NE, T
99999    FORMAT ('   Possibly more than ', I2, ' events would have', &
               /, '   occurred before the time limit expired.', /, &
               '   The last event occurred at time = ', F6.3)
      END IF
      END
!
      REAL FUNCTION FTHETA (T)
      REAL       T
!
      REAL       EXP
      INTRINSIC  EXP
!
      FTHETA = 0.6342*EXP(0.001427*T)
      RETURN
      END
```

Output

```
   Only 12 events occurred before the time
   limit expired.  The last event occurred at
   time = 18.809
```

RNPER

Generates a pseudorandom permutation.

Required Arguments

IPER — Vector of length **K** containing the random permutation of the integers from 1 to **K**. (Output)

Optional Arguments

K — Number of integers to be permuted. (Input)
Default: **K** = size (**IPER**,1).

FORTRAN 90 Interface

Generic: **CALL RNPER** (**IPER** [, ...])
Specific: The specific interface name is **S_RNPER**.

FORTRAN 77 Interface

Single: **CALL RNPER** (**K**, **IPER**)

Description

Routine **RNPER** generates a pseudorandom permutation of the integers from 1 to **K**. It begins by filling a vector of length **K** with the consecutive integers 1 to **K**. Then, with *M* initially equal to **K**, a random index *J* between 1 and *M* (inclusive) is generated. The element of the vector with the index *M* and the element with index *J* swap places in the vector. *M* is then decremented by 1 and the process repeated until *M* = 1.

Comments

The routine **RNSET** can be used to initialize the seed of the random number generator. The routine **RNOPT** can be used to select the form of the generator.

Example

In this example, **RNPER** is called to produce a pseudorandom permutation of the integers from 1 to 10.

```
      USE RNPER_INT
      USE UMACH_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    IPER(10), ISEED, NOUT
!
      CALL UMACH (2, NOUT)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNPER (IPER)
      WRITE (NOUT,99999) IPER
99999 FORMAT ('   Random permutation of the integers from 1 to 10', /, &
            10I5)
      END
```

Output

```
   Random permutation of the integers from 1 to 10
    5    9    2    8    1    6    4    7    3   10
```

RNSRI

Generates a simple pseudorandom sample of indices.

Required Arguments

NPOP — Number of items in the population. (Input)

INDEX — Vector of length NSAMP containing the indices of the sample. (Output)
INDEX is a random sample (without replacement) of the integers from 1 to NPOP, in increasing order.

Optional Arguments

NSAMP — Sample size desired. (Input)
Default: NSAMP = size (INDEX,1).

FORTRAN 90 Interface

Generic: CALL RNSRI (NPOP, INDEX [, ...])
Specific: The specific interface name is S_RNSRI.

FORTRAN 77 Interface

Single: CALL RNSRI (NSAMP, NPOP, INDEX)

Description

Routine RNSRI generates the indices of a pseudorandom sample,without replacement, of size NSAMP numbers from a population of size NPOP. If NSAMP is greater than NPOP/2, the integers from 1 to NPOP are selected sequentially with a probability conditional on the number selected and the number remaining to be considered. If, when the *i*-th population index is considered, *j* items have been included in the sample, then the index *i* is included with probability
(NSAMP - *j*)/(NPOP + 1 - *i*).

If NSAMP is not greater than NPOP/2, a *O*(NSAMP) algorithm due to Ahrens and Dieter (1985) is used. Of the methods discussed by Ahrens and Dieter, the one called SG* is used in RNSRI. It involves a preliminary selection of *q* indices using a geometric distribution for the distances between each index and the next one. If the preliminary sample size *q* is less than NSAMP, a new preliminary sample is chosen, and this is continued until a preliminary sample greater in size than NSAMP is chosen. This preliminary sample is then thinned using the same kind of sampling as described above for the case in which the sample size is greater than half of the population size. Routine RNSRI does not store the preliminary sample indices, but rather restores the state of the generator used in selecting the sample initially, and then passes through once again, making the final selection as the preliminary sample indices are being generated.

Comments

1. The routine RNSET can be used to initialize the seed of the random number generator. If NSAMP is greater than NPOP/2, RNSRI uses two different generators in an algorithm due to Ahrens and Dieter (1985). The routine RNOPT can be used to select the form of the generator used for uniform deviates in the algorithm. The generator used for exponential deviates in the algorithm is a nonshuffled generator that is different from the one for the uniform. If IOPTU is the option indicator for the uniform generator (see documentation for RNOPT), then the option indicator for the exponential generator is MOD((2 * INT((IOPTU + 1)/2) + 1), 6).
2. The routine RNSRS can be used to select a sample from a population of unknown size.

Example

In this example, RNSRI is used to generate the indices of a pseudorandom sample of size 5 from a population of size 100.

```
          USE RNSRI_INT
          USE UMACH_INT
          USE RNSET_INT

          IMPLICIT   NONE
          INTEGER    INDEX(5), ISEED, NOUT, NPOP
    !
          CALL UMACH (2, NOUT)
          NPOP  = 100
          ISEED = 123457
          CALL RNSET (ISEED)
          CALL RNSRI (NPOP, INDEX)
          WRITE (NOUT,99999) INDEX
    99999 FORMAT ('      Random sample: ', 5I4)
          END
```

Output

```
          Random sample:    2   22   53   61  79
```

RNSRS

Generates a simple pseudorandom sample from a finite population.

Required Arguments

POP — NROW by NVAR matrix containing the population to be sampled. (Input) If IDO = 0, POP contains the entire population; otherwise, POP contains a different part of the population on each invocation of RNSRS.

NPOP — The number of items in the population. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2.) If IDO = 0, NPOP = NROW on output. If the population is input a few items at a time, it is not necessary to know the number of items in the population in advance. NPOP is used to cumulate the population size and should not be changed between calls to RNSRS. If, on output, NPOP is greater than or equal to NSAMP, the sampling can be considered complete for a population of size NPOP.

SAMP — NSAMP by NVAR matrix containing the sample. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2.)

INDEX — Vector of length NSAMP containing the indices of the sample in the population. (Output, if IDO = 0 or 1; Input/Output, if IDO = 2.) The INDEX(I)-th item in the population is the I-th item in the sample. INDEX is not necessarily in increasing order.

Optional Arguments

IDO — Processing option. (Input)
Default: IDO = 0.

IDO	Action
0	This is the only invocation of RNSRS for this data set, and the entire population is input at once.
1	This is the first invocation, and additional calls to RNSRS will be made. Initialization and updating for the subpopulation in POP are performed.
2	This is an additional invocation of RNSRS, and updating for the subpopulation in POP is performed.

NROW — Number of rows of data currently input in POP. (Input)
NROW must be nonnegative.
Default: NROW = size (POP,1).

NVAR — Number of variables in the population and in the sample. (Input)
Default: NVAR = size (POP,2).

LDPOP — Leading dimension of POP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDPOP = size (POP,1).

NSAMP — The sample size desired. (Input)
Default: NSAMP = size (SAMP,1).

LDSAMP — Leading dimension of SAMP exactly as specified in the dimension statement in the calling program. (Input)
Default: LDSAMP = size (SAMP,1).

FORTRAN 90 Interface

Generic: CALL RNSRS (POP, NPOP, SAMP, INDEX [, ...])
Specific: The specific interface names are S_RNSRS and D_RNSRS.

FORTRAN 77 Interface

Single: CALL RNSRS (IDO, NROW, NVAR, POP, LDPOP, NSAMP, NPOP, SAMP, LDSAMP, INDEX)
Double: The double precision name is DRNSRS.

Description

Routine RNSRS generates a pseudorandom sample from a given population, without replacement, using an algorithm due to McLeod and Bellhouse (1983).

The first NSAMP items in the population are included in the sample. Then, for each successive item from the population, a random item in the sample is replaced by that item from the population with probability equal to the sample size divided by the number of population items that have been encountered at that time.

Comments

1. The routine RNSET can be used to initialize the seed of the random number generator. The routine RNOPT can be used to select the form of the generator.
2. The routine RNSRI can be used to select a sample of indices in increasing order.

Examples

Example 1

In this example, **RNSRS** is used to generate a sample of size 5 from a population stored in the matrix **POP**. All of the data are available at once, so default **IDO** = 0 is used.

```
      USE RNSRS_INT
      USE UMACH_INT
      USE GDATA_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    I, INDEX(5), ISEED, J, NOUT, NPOP, NROW, NVAR

      REAL       POP(176,2), SAMP(5,2)
!
      CALL UMACH (2, NOUT)
!                                Get Wolfer sunspot data to use
!                                as "population".
      CALL GDATA (2, POP, NROW, NVAR)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET (ISEED)
      CALL RNSRS (POP, NPOP, SAMP, INDEX)
      WRITE (NOUT,99999) NPOP, INDEX, ((SAMP(I,J),I=1,5),J=1,2)
99999 FORMAT ('     The population size is ', I5, /, '   Indices of ', &
            'random sample: ', 5I8, /, '                 The sample: ' &
            , 5F8.0, /, '                            ', 5F8.0)
      END
```

Output

```
 The population size is   176
 Indices of random sample:       16      80     175      25      21
              The sample:    1764.   1828.   1923.   1773.   1769.
                               36.     62.      6.     35.    106.
```

Example 2

Routine **RNSRS** is now used to generate a sample of size 5 from the same population as in the example above except the data are input to **RNSRS** one observation at a time. This is the way **RNSRS** may be used to sample from a file on disk or tape. Notice that the number of records need not be known in advance.

```
      USE RNSRS_INT
      USE UMACH_INT
      USE GDATA_INT
      USE RNSET_INT

      IMPLICIT   NONE
      INTEGER    ISEED, NOUT, IDO, NROW, NVAR, NPOP, INDEX(5), I, J
      REAL       POP(176,2), SAMP(5,2), X(2, 1)
```

```
      CALL UMACH(2, NOUT)
!                                Get Wolfer sunspot data to use
!                                as “population”.
      CALL GDATA (2, POP, NROW, NVAR)
!                                Initialize seed of random number
!                                generator.
      ISEED = 123457
      CALL RNSET(ISEED)
      IDO = 1
      DO 10 I=1,176
!                                In this DO-loop, the data would
!                                generally be read from a file,
!                                one observation at a time.  This
!                                program simulates this by copying
!                                the observations one at a time into
!                                X from POP.
         X(1,1) = POP(I,1)
         X(2,1) = POP(I,2)
         CALL RNSRS (X, NPOP, SAMP, INDEX, IDO=IDO, &
                     NROW=1, NVAR=NVAR, LDPOP=1)
         IDO = 2
   10 CONTINUE
      WRITE(NOUT, 20) NPOP, INDEX, ((SAMP(I,J),I=1,5),J=1,2)
   20 FORMAT ('     The population size is ', I5,/, &
              '   Indices of random sample: ', 5I8,/, &
              '                The sample: ', 5F8.0,/, &
              '                            ', 5F8.0)
      END
```

Output

```
 The population size is   176
 Indices of random sample:       16      80     175      25      21
              The sample:    1764.   1828.   1923.   1773.   1769.
                                36.     62.      6.     35.    106.
```

FAURE_INIT

Shuffled Faure sequence initialization.

Required Arguments

NDIM — The dimension of the hyper-rectangle. (Input)

STATE — An `IMSL_FAURE` pointer for the derived type created by the call to `FAURE_INIT`. The output contains information about the sequence. Use `?_IMSL_FAURE` as the type, where `?_` is `S_` or `D_` depending on precision. (Output)

Optional Arguments

NBASE — The base of the Faure sequence. (Input)

Default: The smallest prime number greater than or equal to `NDIM`.

NSKIP — The number of points to be skipped at the beginning of the Faure sequence. (Input)

Default: $\lfloor NBASE^{m/2-1} \rfloor$, where $m = \lfloor \log(B)/\log(NBASE) \rfloor$ and B is the largest machine representable integer.

FORTRAN 90 Interface

Generic: `CALL FAURE_INIT (NDIM, STATE [, ...])`

Specific: The specific interface names are `S_FAURE_INIT` and `D_FAURE_INIT`.

FAURE_FREE

Frees the structure containing information about the Faure sequence.

Required Arguments

STATE — An `IMSL_FAURE` pointer containing the structure created by the call to `FAURE_INIT`. (Input/Output)

FORTRAN 90 Interface

Generic: `CALL FAURE_FREE (STATE)`

Specific: The specific interface names are `S_FAURE_FREE` and `D_FAURE_FREE`.

FAURE_NEXT

Computes a shuffled Faure sequence.

Required Arguments

STATE — An IMSL_FAURE pointer containing the structure created by the call to FAURE_INIT. The structure contains information about the sequence. The structure should be freed using FAURE_FREE after it is no longer needed. (Input/Output)

NEXT_PT — Vector of length NDIM containing the next point in the shuffled Faure sequence, where NDIM is the dimension of the hyper-rectangle specified in FAURE_INIT. (Output)

Optional Arguments

IMSL_RETURN_SKIP — Returns the current point in the sequence. The sequence can be restarted by calling FAURE_INIT using this value for NSKIP, and using the same value for NDIM. (Input)

FORTRAN 90 Interface

Generic: CALL FAURE_NEXT (STATE, NEXT_PT [, ...])

Specific: The specific interface names are S_FAURE_NEXT and D_FAURE_NEXT.

Description

Discrepancy measures the deviation from uniformity of a point set.

The discrepancy of the point set $x_1, ..., x_n \in [0,1]^d$, $d \geq 1$, is defined

$$D_n^{(d)} = \sup_E \left| \frac{A(E;n)}{n} - \lambda(E) \right|,$$

where the supremum is over all subsets of $[0, 1]^d$ of the form

$$E = \left[0, t_1\right) \times ... \times \left[0, t_d\right), \quad 0 \leq t_j \leq 1, \quad 1 \leq j \leq d$$

λ is the Lebesque measure, and $A(E;n)$ is the number of the x_j contained in E.

The sequence $x_1, x_2, \ldots$ of points $[0,1]^d$ is a low-discrepancy sequence if there exists a constant $c(d)$, depending only on d, such that

$$D_n^{(d)} \le c(d)\frac{\left[\log(n)\right]^d}{n}$$

for all $n > 1$.

Generalized Faure sequences can be defined for any prime base $b \ge d$. The lowest bound for the discrepancy is obtained for the smallest prime $b \ge d$, so the optional argument **NBASE** defaults to the smallest prime greater than or equal to the dimension.

The generalized Faure sequence $x_1, x_2, \ldots$, is computed as follows:

Write the positive integer n in its b-ary expansion,

$$n = \sum_{i=0}^{\infty} a_i(n)\, b^i$$

where $a_i(n)$ are integers, $0 \le a_i(n) < b$.

The j-th coordinate of x_n is

$$x_n^{(j)} = \sum_{k=0}^{\infty}\sum_{d=0}^{\infty} c_{kd}^{(j)}\, a_d(n)\, b^{-k-1}, \quad 1 \le j \le d$$

The generator matrix for the series, $c_{kd}^{(j)}$, is defined to be

$$c_{kd}^{(j)} = j^{d-k} c_{kd}$$

and c_{kd} is an element of the Pascal matrix,

$$c_{kd} = \begin{cases} \frac{d!}{c!(d-c)!} & k \le d \\ 0 & k > d \end{cases}$$

It is faster to compute a shuffled Faure sequence than to compute the Faure sequence itself. It can be shown that this shuffling preserves the low-discrepancy property.

The shuffling used is the b-ary Gray code. The function $G(n)$ maps the positive integer n into the integer given by its b-ary expansion.

The sequence computed by this function is $x(G(n))$, where x is the generalized Faure sequence.

Example

In this example, five points in the Faure sequence are computed. The points are in the three-dimensional unit cube.

Note that FAURE_INIT is used to create a structure that holds the state of the sequence. Each call to FAURE_NEXT returns the next point in the sequence and updates the IMSL_FAURE structure. The final call to FAURE_FREE frees data items, stored in the structure, that were allocated by FAURE_INIT.

```
           use faure_int
           implicit none
           type (s_imsl_faure), pointer  :: state
           real(kind(1e0))          :: x(3)
           integer,parameter :: ndim=3
           integer           :: k
    !                                CREATE THE STRUCTURE THAT HOLDS
    !                                THE STATE OF THE SEQUENCE.
           call faure_init(ndim, state)
    !                                GET THE NEXT POINT IN THE SEQUENCE
           do k=1,5
              call faure_next(state, x)
              write(*,'(3F15.3)') x(1), x(2) , x(3)
           enddo
    !                                 FREE DATA ITEMS STORED IN
    !                                 state STRUCTURE
           call faure_free(state)
                       end
```

Output

```
          0.334      0.493      0.064
          0.667      0.826      0.397
          0.778      0.270      0.175
          0.111      0.604      0.509
          0.445      0.937      0.842
```

Utilities

Routines

19.6 Time, Date, and Version

19.7 Retrieval of Data Sets

WRRRN

Prints a real rectangular matrix with integer row and column labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title. Use "% /" within the title to create a new line. Long titles are automatically wrapped.

A — NRA by NCA matrix to be printed. (Input)

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A,1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of A is printed, including the diagonal.
2	Upper triangle of A excluding the diagonal of A is printed.
-1	Lower triangle of A is printed, including the diagonal.
-2	Lower triangle of A excluding the diagonal of A is printed.

FORTRAN 90 Interface

Generic: CALL WRRRN (TITLE, A [, ...])

Specific: The specific interface names are **S_WRRRN** and **D_WRRRN** for two dimensional arrays, and **S_WRRRN1D** and **D_WRRRN1D** for one dimensional arrays.

FORTRAN 77 Interface

Single: **CALL WRRRN** (**TITLE**, **NRA**, **NCA**, **A**, **LDA**, **ITRING**)

Double: The double precision name is **DWRRRN**.

Description

Routine **WRRRN** prints a real rectangular matrix with the rows and columns labeled 1, 2, 3, and so on. **WRRRN** can restrict printing to the elements of the upper or lower triangles of matrices via the **ITRING** option. Generally, **ITRING** ≠ 0 is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set **NRA** to the length of the array and set **NCA** = 1. For a row vector, set **NRA** = 1 and set **NCA** to the length of the array. In both cases, set **LDA** = **NRA** and set **ITRING** = 0.

Comments

1. A single **D**, **E**, or **F** format is chosen automatically in order to print 4 significant digits for the largest element of **A** in absolute value. Routine **WROPT** can be used to change the default format.
2. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking **WROPT**.
3. A page width of 78 characters is used. Page width and page length can be reset by invoking **PGOPT**.
4. Output is written to the unit specified by **UMACH** (see the Reference Material).

Example

The following example prints all of a 3 × 4 matrix *A* where $a_{ij} = i + j/10$.

```
          USE WRRRN_INT
          
          IMPLICIT   NONE
          INTEGER    ITRING, LDA, NCA, NRA
          PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
    !
          INTEGER    I, J
          REAL       A(LDA,NCA)
    !
```

```
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                  Write A matrix.
      CALL WRRRN ('A', A, NRA=NRA)
      END
```

Output

```
                    A
          1       2       3       4
1   1.100   1.200   1.300   1.400
2   2.100   2.200   2.300   2.400
3   3.100   3.200   3.300   3.400
```

WRRRL

Print a real rectangular matrix with a given format and labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
TITLE set equal to a blank character(s) suppresses printing of the title.

A — NRA by NCA matrix to be printed. (Input)

RLABEL — CHARACTER * (*) vector of labels for rows of A. (Input)
If rows are to be numbered consecutively 1, 2, ..., NRA, use RLABEL(1) = 'NUMBER'. If no row labels are desired, use RLABEL(1) = 'NONE'. Otherwise, RLABEL is a vector of length NRA containing the labels.

CLABEL — CHARACTER * (*) vector of labels for columns of A. (Input)
If columns are to be numbered consecutively 1, 2, ..., NCA, use CLABEL(1) = 'NUMBER'. If no column labels are desired, use CLABEL(1) = 'NONE'. Otherwise, CLABEL(1) is the heading for the row labels, and either CLABEL(2) must be 'NUMBER' or 'NONE', or CLABEL must be a vector of length NCA + 1 with CLABEL(1 + *j*) containing the column heading for the *j*-th column.

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A,1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

ITRING — Triangle option. (Input)
Default: ITRING = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of A is printed, including the diagonal.
2	Upper triangle of A excluding the diagonal of A is printed.
-1	Lower triangle of A is printed, including the diagonal.
-2	Lower triangle of A excluding the diagonal of A is printed.

FMT — Character string containing formats. (Input)

If FMT is set to a blank character(s), the format used is specified by WROPT. Otherwise, FMT must contain exactly one set of parentheses and one or more edit descriptors. For example, FMT = '(F10.3)' specifies this F format for the entire matrix. FMT = '(2E10.3, 3F10.3)' specifies an E format for columns 1 and 2 and an F format for columns 3, 4 and 5. If the end of FMT is encountered and if some columns of the matrix remain, format control continues with the first format in FMT. Even though the matrix A is real, an I format can be used to print the integer part of matrix elements of A. The most useful formats are special formats, called the "V and W formats," that can be used to specify pretty formats automatically. Set FMT = '(V10.4)' if you want a single D, E, or F format selected automatically with field width 10 and with 4 significant digits. Set FMT = '(W10.4)' if you want a single D, E, F, or I format selected automatically with field width 10 and with 4 significant digits. While the V format prints trailing zeroes and a trailing decimal point, the W format does not. See Comment 4 for general descriptions of the V and W formats. FMT may contain only D, E, F, G, I, V, or W edit descriptors, e.g., the X descriptor is not allowed.
Default: FMT = ' '.

FORTRAN 90 Interface

Generic: CALL WRRRL (TITLE, A, RLABEL, CLABEL [, ...])

Specific: The specific interface names are S_WRRRL and D_WRRRL for two dimensional arrays, and S_WRRRL1D and D_WRRRL1D for one dimensional arrays.

FORTRAN 77 Interface

Single: CALL WRRRL (TITLE, NRA, NCA, A, LDA, ITRING, FMT, RLABEL, CLABEL)

Double: The double precision name is DWRRRL.

Description

Routine **WRRRL** prints a real rectangular matrix (stored in *A*) with row and column labels (specified by **RLABEL** and **CLABEL**, respectively) according to a given format (stored in **FMT**). **WRRRL** can restrict printing to the elements of upper or lower triangles of matrices via the **ITRING** option. Generally, **ITRING** ≠ 0 is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set **NRA** to the length of the array and set **NCA** = 1. For a row vector, set **NRA** = 1 and set **NCA** to the length of the array. In both cases, set **LDA** = **NRA**, and set **ITRING** = 0.

Comments

1. Workspace may be explicitly provided, if desired, by use of **W2RRL/DW2RRL**. The reference is:

   CALL W2RRL (TITLE, NRA, NCA, A, LDA, ITRING, FMT, RLABEL, CLABEL, CHWK)

   The additional argument is:

   CHWK — CHARACTER * 10 work vector of length **NCA**. This workspace is referenced only if all three conditions indicated at the beginning of this comment are met. Otherwise, **CHWK** is not referenced and can be a **CHARACTER *** 10 vector of length one.

2. The output appears in the following form:

   TITLE

CLABEL(1)	CLABEL(2)	CLABEL(3)	CLABEL(4)
RLABEL(1)	Xxxxx	xxxxx	Xxxxx
RLABEL(2)	Xxxxx	xxxxx	Xxxxx

3. Use "% /" within titles or labels to create a new line. Long titles or labels are automatically wrapped.

4. For printing numbers whose magnitudes are unknown, the **G** format in FORTRAN is useful; however, the decimal points will generally not be aligned when printing a column of numbers. The **V** and **W** formats are special formats used by this routine to select a **D**, **E**, **F**, or **I** format so that the decimal points will be aligned. The **V** and **W** formats are specified as *Vn.d* and *Wn.d*. Here, *n* is the field width and *d* is the number of significant digits generally printed. Valid values for *n* are 3, 4,..., 40. Valid values for *d* are 1, 2, ..., *n* - 2. If **FMT** specifies one format and that format is a **V** or **W** format, all elements of the matrix **A** are examined to determine one FORTRAN format for printing. If **FMT** specifies more than one format, FORTRAN formats are generated separately from each **V** or **W** format.

5. A page width of 78 characters is used. Page width and page length can be reset by invoking **PGOPT**.

6. Horizontal centering, method for printing large matrices, paging, method for printing NaN (not a number), printing a title on each page, and many other options can be selected by invoking **WROPT**.

7. Output is written to the unit specified by **UMACH** (see the Reference Material).

Example

The following example prints all of a 3 × 4 matrix *A* where $a_{ij} = (i + j/10)10^{j-3}$.

```
      USE WRRRL_INT
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
!
      INTEGER    I, J
      REAL       A(LDA,NCA)
      CHARACTER  CLABEL(5)*5, FMT*8, RLABEL(3)*5
!
      DATA FMT/'(W10.6)'/
      DATA CLABEL/'   ', 'Col 1', 'Col 2', 'Col 3', 'Col 4'/
      DATA RLABEL/'Row 1', 'Row 2', 'Row 3'/
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = (I+J*0.1)*10.0**(J-3)
   10    CONTINUE
   20 CONTINUE
!                                Write A matrix.
      CALL WRRRL ('A', A, RLABEL, CLABEL, NRA=NRA, FMT=FMT)
      END
```

Output

```
                              A
            Col 1       Col 2       Col 3       Col 4
Row 1       0.011       0.120       1.300      14.000
Row 2       0.021       0.220       2.300      24.000
Row 3       0.031       0.320       3.300      34.000
```

WRIRN

Prints an integer rectangular matrix with integer row and column labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
`TITLE` set equal to a blank character(s) suppresses printing of the title. Use "% /" within the title to create a new line. Long titles are automatically wrapped.

MAT — `NRMAT` by `NCMAT` matrix to be printed. (Input)

Optional Arguments

NRMAT — Number of rows. (Input)
Default: `NRMAT` = size (`MAT`,1).

NCMAT — Number of columns. (Input)
Default: `NCMAT` = size (`MAT`,2).

LDMAT — Leading dimension of `MAT` exactly as specified in the dimension statement in the calling program. (Input)
Default: `LDMAT` = size (`MAT`,1).

ITRING — Triangle option. (Input)
Default: `ITRING` = 0.

`ITRING`	Action
0	Full matrix is printed.
1	Upper triangle of `MAT` is printed, including the diagonal.
2	Upper triangle of `MAT` excluding the diagonal of `MAT` is printed.
-1	Lower triangle of `MAT` is printed, including the diagonal.
-2	Lower triangle of `MAT` excluding the diagonal of `MAT` is printed.

FORTRAN 90 Interface

Generic: `CALL WRIRN (TITLE, MAT [, ...])`
Specific: The specific interface name is `S_WRIRN`.

FORTRAN 77 Interface

Single: CALL WRIRN (TITLE, NRMAT, NCMAT, MAT, LDMAT, ITRING).

Description

Routine **WRIRN** prints an integer rectangular matrix with the rows and columns labeled 1, 2, 3, and so on. **WRIRN** can restrict printing to elements of the upper and lower triangles of matrices via the **ITRING** option. Generally, **ITRING** ≠ 0 is used with symmetric matrices.

In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set **NRMAT** to the length of the array and set **NCMAT** = 1. For a row vector, set **NRMAT** = 1 and set **NCMAT** to the length of the array. In both cases, set **LDMAT** = **NRMAT** and set **ITRING** = 0.

Comments

1. All the entries in **MAT** are printed using a single **I** format. The field width is determined by the largest absolute entry.
2. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking **WROPT**.
3. A page width of 78 characters is used. Page width and page length can be reset by invoking **PGOPT**.
4. Output is written to the unit specified by **UMACH** (see the Reference Material).

Example

The following example prints all of a 3 × 4 matrix A = **MAT** where $a_{ij} = 10i + j$.

```
      USE WRIRN_INT

      IMPLICIT   NONE
      INTEGER    ITRING, LDMAT, NCMAT, NRMAT
      PARAMETER  (ITRING=0, LDMAT=10, NCMAT=4, NRMAT=3)
!
      INTEGER    I, J, MAT(LDMAT,NCMAT)
!
      DO 20  I=1, NRMAT
         DO 10  J=1, NCMAT
            MAT(I,J) = I*10 + J
   10    CONTINUE
   20 CONTINUE
!                                Write MAT matrix.
      CALL WRIRN ('MAT', MAT, NRMAT=NRMAT)
      END
```

Output

```
          MAT
        1   2   3   4
    1  11  12  13  14
    2  21  22  23  24
    3  31  32  33  34
```

WRIRL

Print an integer rectangular matrix with a given format and labels.

Required Arguments

TITLE — Character string specifying the title. (Input)
`TITLE` set equal to a blank character(s) suppresses printing of the title.

MAT — `NRMAT` by `NCMAT` matrix to be printed. (Input)

RLABEL — `CHARACTER *` (`*`) vector of labels for rows of `MAT`. (Input)
If rows are to be numbered consecutively 1, 2, ..., `NRMAT`, use `RLABEL`(1) = `'NUMBER'`. If no row labels are desired, use `RLABEL`(1) = `'NONE'`. Otherwise, `RLABEL` is a vector of length `NRMAT` containing the labels.

CLABEL — `CHARACTER *` (`*`) vector of labels for columns of `MAT`. (Input)
If columns are to be numbered consecutively 1, 2, ..., `NCMAT`, use `CLABEL`(1) = `'NUMBER'`. If no column labels are desired, use `CLABEL`(1) = `'NONE'`. Otherwise, `CLABEL`(1) is the heading for the row labels, and either `CLABEL`(2) must be `'NUMBER'` or `'NONE'`, or `CLABEL` must be a vector of length `NCMAT` + 1 with `CLABEL`(1 + *j*) containing the column heading for the *j*-th column.

Optional Arguments

NRMAT — Number of rows. (Input)
Default: `NRMAT` = size (`MAT`,1).

NCMAT — Number of columns. (Input)
Default: `NCMAT` = size (`MAT`,2).

LDMAT — Leading dimension of `MAT` exactly as specified in the dimension statement in the calling program. (Input)
Default: `LDMAT` = size (`MAT`,1).

ITRING — Triangle option. (Input)
Default: `ITRING` = 0.

ITRING	Action
0	Full matrix is printed.
1	Upper triangle of MAT is printed, including the diagonal.
2	Upper triangle of MAT excluding the diagonal of MAT is printed.
-1	Lower triangle of MAT is printed, including the diagonal.
-2	Lower triangle of MAT excluding the diagonal of MAT is printed.

FMT — Character string containing formats. (Input)

If FMT is set to a blank character(s), the format used is a single I format with field width determined by the largest absolute entry. Otherwise, FMT must contain exactly one set of parentheses and one or more I edit descriptors. For example, FMT = '(I10)' specifies this I format for the entire matrix. FMT = '(2I10, 3I5)' specifies an I10 format for columns 1 and 2 and an I5 format for columns 3, 4 and 5. If the end of FMT is encountered and if some columns of the matrix remain, format control continues with the first format in FMT. FMT may only contain the I edit descriptor, e.g., the X edit descriptor is not allowed.

Default: FMT = ' '.

FORTRAN 90 Interface

Generic: CALL WRIRL (TITLE, MAT, RLABEL, CLABEL [, ...])

Specific: The specific interface name is S_WRIRL.

FORTRAN 77 Interface

Single: CALL WRIRL (TITLE, NRMAT, NCMAT, MAT, LDMAT, ITRING, FMT, RLABEL, CLABEL)

Description

Routine WRIRL prints an integer rectangular matrix (stored in MAT) with row and column labels (specified by RLABEL and CLABEL, respectively), according to a given format (stored in FMT). WRIRL can restrict printing to the elements of upper or lower triangles of matrices via the ITRING option. Generally, ITRING $\neq$ 0 is used with symmetric matrices. In addition, one-dimensional arrays can be printed as column or row vectors. For a column vector, set NRMAT to the length of the array and set NCMAT = 1. For a row vector, set NRMAT = 1 and set NCMAT to the length of the array. In both cases, set LDMAT = NRMAT, and set ITRING = 0.

Comments

1. The output appears in the following form:

```
                                TITLE

CLABEL(1)   CLABEL(2)    CLABEL(3)    CLABEL(4)

RLABEL(1)   Xxxxx        xxxxx        xxxxx

RLABEL(2)   Xxxxx        xxxxx        xxxxx
```

2. Use "% /" within titles or labels to create a new line. Long titles or labels are automatically wrapped.
3. A page width of 78 characters is used. Page width and page length can be reset by invoking **PGOPT**.
4. Horizontal centering, a method for printing large matrices, paging, printing a title on each page, and many other options can be selected by invoking **WROPT**.
5. Output is written to the unit specified by **UMACH** (see the Reference Material).

Example

The following example prints all of a 3 × 4 matrix A = **MAT** where $a_{ij} = 10i + j$.

```
      USE WRIRL_INT

      IMPLICIT   NONE
      INTEGER    ITRING, LDMAT, NCMAT, NRMAT

      PARAMETER  (ITRING=0, LDMAT=10, NCMAT=4, NRMAT=3)
!
      INTEGER    I, J, MAT(LDMAT,NCMAT)
      CHARACTER  CLABEL(5)*5, FMT*8, RLABEL(3)*5
!
      DATA FMT/'(I2)'/
      DATA CLABEL/'     ', 'Col 1', 'Col 2', 'Col 3', 'Col 4'/
      DATA RLABEL/'Row 1', 'Row 2', 'Row 3'/
!
      DO 20  I=1, NRMAT
         DO 10  J=1, NCMAT
            MAT(I,J) = I*10 + J
   10    CONTINUE
   20 CONTINUE
!                                Write MAT matrix.
      CALL WRIRL ('MAT', MAT, RLABEL, CLABEL, NRMAT=NRMAT)
      END
```

Output

```
                    MAT
        Col 1  Col 2  Col 3  Col 4
```

```
Row 1    11    12    13    14
Row 2    21    22    23    24
Row 3    31    32    33    34
```

WROPT

Sets or retrieves an option for printing a matrix.

Required Arguments

IOPT — Indicator of option type. (Input)

IOPT	Description of Option Type
-1, 1	Horizontal centering or left justification of matrix to be printed
-2, 2	Method for printing large matrices
-3, 3	Paging
-4, 4	Method for printing NaN (not a number), and negative and positive machine infinity.
-5, 5	Title option
-6, 6	Default format for real and complex numbers
-7, 7	Spacing between columns
-8, 8	Maximum horizontal space reserved for row labels
-9, 9	Indentation of continuation lines for row labels
-10, 10	Hot zone option for determining line breaks for row labels
-11, 11	Maximum horizontal space reserved for column labels
-12, 12	Hot zone option for determining line breaks for column labels
-13, 13	Hot zone option for determining line breaks for titles
-14, 14	Option for the label that appears in the upper left hand corner that can be used as a heading for the row numbers or a label for the column headings for WR**N routines
-15, 15	Option for skipping a line between invocations of WR**N routines, provided a new page is not to be issued
-16, 16	Option for vertical alignment of the matrix values relative to the associated row labels that occupy more than one line
0	Reset all the current settings saved in internal variables back to their last setting made with an invocation of WROPT with ISCOPE = 1. (This option is used internally by routines printing a matrix and is not useful otherwise.)

If **IOPT** is negative, **ISET** and **ISCOPE** are input and are saved in internal variables. If **IOPT** is positive, **ISET** is output and receives the currently active setting for the option (if **ISCOPE** = 0) or the last global setting for the option (if **ISCOPE** = 1). If **IOPT** = 0, **ISET** and **ISCOPE** are not referenced.

ISET — Setting for option selected by **IOPT**. (Input, if **IOPT** is negative; output, if **IOPT** is positive; not referenced if **IOPT** = 0)

IOPT	ISET	Meaning
-1, 1	0	Matrix is left justified
	1	Matrix is centered horizontally on page
-2, 2	0	A complete row is printed before the next row is printed. Wrapping is used if necessary.

IOPT	ISET	Meaning
	m	Here, *m* is a positive integer. Let n_1 be the maximum number of columns beginning with column 1 that fit across the page (as determined by the widths of the printing formats). First, columns 1 through n_1 are printed for rows 1 through *m*. Let n_2 be the maximum number of columns beginning with column $n_1 + 1$ that fit across the page. Second, columns $n_1 + 1$ through $n_1 + n_2$ are printed for rows 1 through *m*. This continues until the last columns are printed for rows 1 through *m*. Printing continues in this fashion for the next *m* rows, etc.
-3, 3	-2	Printing begins on the next line, and no paging occurs.
	-1	Paging is on. Every invocation of a `WR***` routine begins on a new page, and paging occurs within each invocation as is needed
	0	Paging is on. The first invocation of a `WR***` routine begins on a new page, and subsequent paging occurs as is needed. With this option, every invocation of a `WR***` routine ends with a call to `WROPT` to reset this option to *k*, a positive integer giving the number of lines printed on the current page.
	k	Here, k is a positive integer. Paging is on, and *k* lines have been printed on the current page. If *k* is less than the page length `IPAGE` (see PGOPT), then `IPAGE - k` lines are printed before a new page instruction is issued. If k is greater than or equal to `IPAGE`, then the first invocation of a `WR***` routine begins on a new page. In any case, subsequent paging occurs as is needed. With this option, every invocation of a `WR***` routine ends with a call to `WROPT` to reset the value of *k*.
-4, 4	0	NaN is printed as a series of decimal points, negative machine infinity is printed as a series of minus signs, and positive machine infinity is printed as a series of plus signs.
	1	NaN is printed as a series of blank characters, negative machine infinity is printed as a series of minus signs, and positive machine infinity is printed as a series of plus signs.
	2	NaN is printed as "NaN," negative machine infinity is printed as "`-Inf`" and positive machine infinity is printed as "`Inf`."
	3	NaN is printed as a series of blank characters, negative machine infinity is printed as "`-Inf`," and positive machine infinity is printed as "`Inf`."
-5, 5	0	Title appears only on first page.
	1	Title appears on the first page and all continuation pages.
-6, 6	0	Format is (`W10.4`). See Comment 2.
	1	Format is (`W12.6`). See Comment 2.
	2	Format is (`1PE12.5`).
	3	Format is *Vn*.4 where the field width *n* is determined. See Comment 2.
	4	Format is *Vn*.6 where the field width *n* is determined. See Comment 2.

IOPT	ISET	Meaning
	5	Format is 1PE*n*.*d* where *n* = *d* + 7, and *d* + 1 is the maximum number of significant digits.
-7, 7	k_1	Number of characters left blank between columns. k_1 must be between 0 and 5, inclusively.
-8, 8	k_2	Maximum width (in characters) reserved for row labels. k_2 = 0 means use the default.
-9, 9	k_3	Number of characters used to indent continuation lines for row labels. k_3 must be between 0 and 10, inclusively.
-10, 10	k_4	Width (in characters) of the hot zone where line breaks in row labels can occur. k_4 = 0 means use the default. k_4 must not exceed 50.
-11, 11	k_5	Maximum width (in characters) reserved for column labels. k_5 = 0 means use the default.
-12, 12	k_6	Width (in characters) of the hot zone where line breaks in column labels can occur. k_6 = 0 means use the default. k_6 must not exceed 50.
-13, 13	k_7	Width (in characters) of the hot zone where line breaks in titles can occur. k_7 must be between 1 and 50, inclusively.
-14	0	There is no label in the upper left hand corner.
	1	The label in the upper left hand corner is “Component” if a row vector or column vector is printed; the label is “Row/Column” if both the number of rows and columns are greater than one; otherwise, there is no label.
-15	0	A blank line is printed on each invocation of a WR**N routine before the matrix title provided a new page is not to be issued.
	1	A blank line is not printed on each invocation of a WR**N routine before the matrix title.
-16, 16	0	The matrix values are aligned vertically with the last line of the associated row label for the case IOPT = 2 and ISET is positive.
	1	The matrix values are aligned vertically with the first line of the associated row label.

ISCOPE — Indicator of the scope of the option. (Input if IOPT is nonzero; not referenced if IOPT = 0)

ISCOPE	Action
0	Setting is temporarily active for the next invocation of a WR*** matrix printing routine.
1	Setting is active until it is changed by another invocation of WROPT.

FORTRAN 90 Interface

Generic: CALL WROPT (IOPT, ISET, ISCOPE)

Specific: The specific interface name is **S_WROPT**.

FORTRAN 77 Interface

Single: **CALL WROPT** (**IOPT**, **ISET**, **ISCOPE**)

Description

Routine **WROPT** allows the user to set or retrieve an option for printing a matrix. The options controlled by **WROPT** include the following: horizontal centering, a method for printing large matrices, paging, method for printing NaN (not a number) and positive and negative machine infinities, printing titles, default formats for numbers, spacing between columns, maximum widths reserved for row and column labels, indentation of row labels that continue beyond one line, widths of hot zones for breaking of labels and titles, the default heading for row labels, whether to print a blank line between invocations of routines, and vertical alignment of matrix entries with respect to row labels continued beyond one line. (NaN and positive and negative machine infinities can be retrieved by **AMACH** and **DMACH** that are documented in the section "Machine-Dependent Constants" in the Reference Material.) Options can be set globally (**ISCOPE** = 1) or temporarily for the next call to a printing routine (**ISCOPE** = 0).

Comments

1. This program can be invoked repeatedly before using a **WR*** ** routine to print a matrix. The matrix printing routines retrieve these settings to determine the printing options. It is not necessary to call **WROPT** if a default value of a printing option is desired. The defaults are as follows.

	ISET	**Meaning**
1	0	Left justified
2	1000000	Number lines before wrapping
3	-2	No paging
4	2	NaN is printed as "NaN," negative machine infinity is printed as "`-Inf`" and positive machine infinity is printed as "`Inf`."
5	0	Title only on first page.
6	3	Default format is *Vn*.4.
7	2	2 spaces between columns.

	ISET	Meaning
8	0	Maximum row label width MAXRLW = 2 * IPAGEW/3 if matrix has one column; MAXRLW = IPAGEW/4 otherwise.
9	3	3 character indentation of row labels continued beyond one line.
10	0	Width of row label hot zone is MAXRLW/3 characters.
11	0	Maximum column label width MAXCLW = min{max (NW + NW/2, 15), 40} for integer and real matrices, where NW is the field width for the format corresponding to the particular column. MAXCLW = min{max(NW + NW/2, 15), 83} for complex matrices, where NW is the sum of the two field widths for the formats corresponding to the particular column plus 3.
12	0	Width of column label hot zone is MAXCLW/3 characters.
13	10	Width of hot zone for titles is 10 characters.
14	0	There is no label in the upper left hand corner.
15	0	Blank line is printed.
16	0	The matrix values are aligned vertically with the last line of the associated row label.

For **IOPT** = 8, the default depends on the current value for the page width, **IPAGEW** (see **PGOPT**).

2. The **V** and **W** formats are special formats that can be used to select a **D**, **E**, **F**, or **I** format so that the decimal points will be aligned. The **V** and **W** formats are specified as *Vn.d* and *Wn.d*. Here, *n* is the field width and *d* is the number of significant digits generally printed. Valid values for *n* are 3, 4, ..., 40. Valid values for *d* are 1, 2, ..., *n* - 2. While the **V** format prints trailing zeroes and a trailing decimal point, the **W** format does not.

Example

The following example illustrates the effect of **WROPT** when printing a 3 × 4 real matrix *A* with **WRRRN** where $a_{ij} = i + j/10$. The first call to **WROPT** sets horizontal printing so that the matrix is first printed horizontally centered on the page. In the next invocation of **WRRRN**, the left-justification option has been set via routine **WROPT** so the matrix is left justified when printed. Finally, because the scope of left justification was only for the next call to a printing routine, the last call to **WRRRN** results in horizontally centered printing.

```
        USE WROPT_INT
        USE WRRRN_INT
```

```
      IMPLICIT   NONE
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=10, NCA=4, NRA=3)
!
      INTEGER    I, IOPT, ISCOPE, ISETNG, J
      REAL       A(LDA,NCA)
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                Activate centering option.
!                                Scope is global.
      IOPT   = -1
      ISETNG = 1
      ISCOPE = 1
!
      CALL WROPT (IOPT, ISETNG, ISCOPE)
!                                Write A matrix.
      CALL WRRRN ('A', A, NRA=NRA)
!                                Activate left justification.
!                                Scope is local.
      IOPT   = -1
      ISETNG   = 0
      ISCOPE = 0
      CALL WROPT (IOPT, ISETNG, ISCOPE)
      CALL WRRRN ('A', A, NRA=NRA)
      CALL WRRRN ('A', A, NRA=NRA)
      END
```

Output

```
                                    A
                              1       2       3       4
                      1   1.100   1.200   1.300   1.400
                      2   2.100   2.200   2.300   2.400
                      3   3.100   3.200   3.300   3.400

                A
          1       2       3       4
1   1.100   1.200   1.300   1.400
2   2.100   2.200   2.300   2.400
3   3.100   3.200   3.300   3.400

                                    A
                              1       2       3       4
                      1   1.100   1.200   1.300   1.400
                      2   2.100   2.200   2.300   2.400
                      3   3.100   3.200   3.300   3.400
```

PGOPT

Sets or retrieves page width and length for printing.

Required Arguments

IOPT — Page attribute option. (Input)

IOPT	Description of Attribute
-1, 1	Page width.
-2, 2	Page length.

Negative values of IOPT indicate the setting IPAGE is input. Positive values of IOPT indicate the setting IPAGE is output.

IPAGE — Value of page attribute. (Input, if IOPT is negative; output, if IOPT is positive.)

IOPT	Description of Attribute	Settings for IPAGE
-1, 1	Page width (in characters)	10, 11, …
-2, 2	Page length (in lines)	10, 11, …

FORTRAN 90 Interface

Generic: CALL PGOPT (IOPT, IPAGE)

Specific: The specific interface name is S_PGOPT.

FORTRAN 77 Interface

Single: CALL PGOPT (IOPT, IPAGE)

Description

Routine PGOPT is used to set or retrieve the page width or the page length for routines that perform printing.

Example

The following example illustrates the use of **PGOPT** to set the page width at 20 characters. Routine **WRRRN** is then used to print a 3 × 4 matrix *A* where $a_{ij} = i + j/10$.

```
      USE PGOPT_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    ITRING, LDA, NCA, NRA
      PARAMETER  (ITRING=0, LDA=3, NCA=4, NRA=3)
!
      INTEGER    I, IOPT, IPAGE, J
      REAL       A(LDA,NCA)
!
      DO 20  I=1, NRA
         DO 10  J=1, NCA
            A(I,J) = I + J*0.1
   10    CONTINUE
   20 CONTINUE
!                                  Set page width.
      IOPT  = -1
      IPAGE = 20
      CALL PGOPT (IOPT, IPAGE)
!                                  Print the matrix A.
      CALL WRRRN ('A', A)
      END
```

Output

```
           A
         1       2
1   1.100   1.200
2   2.100   2.200
3   3.100   3.200

         3       4
1   1.300   1.400
2   2.300   2.400
3   3.300   3.400
```

PERMU

Rearranges the elements of an array as specified by a permutation.

Required Arguments

X — Real vector of length **N** containing the array to be permuted. (Input)

IPERMU — Integer vector of length **N** containing a permutation **IPERMU**(1), ..., **IPERMU**(**N**) of the integers 1, ..., **N**. (Input)

XPERMU — Real vector of length **N** containing the array **X** permuted. (Output)
If **X** is not needed, **X** and **XPERMU** can share the same storage locations.

Optional Arguments

N — Length of the arrays **X** and **XPERMU**. (Input)
Default: **N** = size (**X**,1).

IPATH — Integer flag. (Input)
Default: **IPATH** = 1.
IPATH = 1 means **IPERMU** represents a forward permutation, i.e., **X**(**IPERMU**(**I**)) is moved to **XPERMU**(**I**). **IPATH** = 2 means **IPERMU** represents a backward permutation, i.e., **X**(**I**) is moved to **XPERMU**(**IPERMU**(**I**)).

FORTRAN 90 Interface

Generic: **CALL PERMU** (**X**, **IPERMU**, **XPERMU** [, ...])
Specific: The specific interface names are **S_PERMU** and **D_PERMU**.

FORTRAN 77 Interface

Single: **CALL PERMU** (**N**, **X**, **IPERMU**, **IPATH**, **XPERMU**)
Double: The double precision name is **DPERMU**.

Description

Routine **PERMU** rearranges the elements of an array according to a permutation vector. It has the option to do both forward and backward permutations.

Example

This example rearranges the array *X* using **IPERMU**; forward permutation is performed.

```
      USE PERMU_INT
      USE UMACH_INT

      IMPLICIT   NONE
!                                  Declare variables
      INTEGER    IPATH, N
      PARAMETER  (IPATH=1, N=4)
!
      INTEGER    IPERMU(N), J, NOUT
      REAL       X(N), XPERMU(N)
!                                  Set values for  X, IPERMU
!
!                            X = ( 5.0  6.0  1.0  4.0 )
!                            IPERMU = ( 3 1 4 2 )
!
      DATA X/5.0, 6.0, 1.0, 4.0/, IPERMU/3, 1, 4, 2/
!                                  Permute X into XPERMU
      CALL PERMU (X, IPERMU, XPERMU)
!                                  Get output unit number
      CALL UMACH (2, NOUT)
!                                  Print results
      WRITE (NOUT,99999) (XPERMU(J),J=1,N)
!
99999 FORMAT ('  The output vector is:', /, 10(1X,F10.2))
      END
```

Output

```
 The Output vector is:
 1.00       5.00       4.00       6.00
```

PERMA

Permutes the rows or columns of a matrix.

Required Arguments

A — NRA by NCA matrix to be permuted. (Input)

IPERMU — Vector of length K containing a permutation IPERMU(1), ..., IPERMU(K) of the integers 1, ..., K where K = NRA if the rows of A are to be permuted and K = NCA if the columns of A are to be permuted. (Input)

APER — NRA by NCA matrix containing the permuted matrix. (Output)
If A is not needed, A and APER can share the same storage locations.

Optional Arguments

NRA — Number of rows. (Input)
Default: NRA = size (A,1).

NCA — Number of columns. (Input)
Default: NCA = size (A,2).

LDA — Leading dimension of A exactly as specified in the dimension statement of the calling program. (Input)
Default: LDA = size (A,1).

IPATH — Option parameter. (Input)
IPATH = 1 means the rows of A will be permuted. IPATH = 2 means the columns of A will be permuted.
Default: IPATH = 1.

LDAPER — Leading dimension of APER exactly as specified in the dimension statement of the calling program. (Input)
Default: LDAPER = size (APER,1).

FORTRAN 90 Interface

Generic: CALL PERMA (A, IPERMU, APER [, ...])

Specific: The specific interface names are S_PERMA and D_PERMA.

FORTRAN 77 Interface

Single: CALL PERMA (NRA, NCA, A, LDA, IPERMU, IPATH, APER, LDAPER)

Double: The double precision name is DPERMA.

Description

Routine PERMA interchanges the rows or columns of a matrix using a permutation vector such as the one obtained from routines SVRBP (see the Utilities chapter in Math Library manual) or SVRGP.

The routine PERMA permutes a column (row) at a time by calling PERMU. This process is continued until all the columns (rows) are permuted. On completion, let B = APER and p_i = IPERMU(I), then

$$B_{ij} = A_{p_i j}$$

for all i, j.

Comments

1. Workspace may be explicitly provided, if desired, by use of P2RMA/DP2RMA. The reference is:

   CALL P2RMA (NRA, NCA, A, LDA, IPERMU, IPATH, APER, LDAPER, WORK)

   The additional argument is:

   WORK — Real work vector of length NCA.

Example

This example permutes the columns of a matrix A.

```
      USE PERMA_INT
      USE UMACH_INT

      IMPLICIT   NONE
!                                Declare variables
      INTEGER    IPATH, LDA, LDAPER, NCA, NRA
      PARAMETER  (IPATH=2, LDA=3, LDAPER=3, NCA=5, NRA=3)
!
      INTEGER    I, IPERMU(5), J, NOUT
      REAL       A(LDA,NCA), APER(LDAPER,NCA)
!                                Set values for  A, IPERMU
!                                A = ( 3.0  5.0  1.0  2.0  4.0 )
!                                    ( 3.0  5.0  1.0  2.0  4.0 )
!                                    ( 3.0  5.0  1.0  2.0  4.0 )
!
!                                IPERMU = ( 3 4 1 5 2 )
!
```

```
      DATA A/3*3.0, 3*5.0, 3*1.0, 3*2.0, 3*4.0/, IPERMU/3, 4, 1, 5, 2/
!                                Perform column permutation on A,
!                                giving APER
      CALL PERMA (A, IPERMU, APER, IPATH=IPATH)
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Print results
      WRITE (NOUT,99999) ((APER(I,J),J=1,NCA),I=1,NRA)
!
99999 FORMAT ('  The output matrix is:', /, 3(5F8.1,/))
      END
```

Output

```
 The Output matrix is:
 1.0     2.0     3.0     4.0     5.0
 1.0     2.0     3.0     4.0     5.0
 1.0     2.0     3.0     4.0     5.0
```

RORDM

Reorders rows and columns of a symmetric matrix.

Required Arguments

AA — NAA by NAA symmetric matrix to be reordered. (Input)
Only elements in the upper triangle of AA are referenced.

INDAA — Index vector of length NA containing the indices of the rows/columns of AA that are being selected for inclusion into A. (Input)
INDAA(I) = J means the J-th row and column of AA will be the I-th row and column of A.

A — NAA by NAA matrix containing the reordered AA. (Output)
The first NA rows and columns of A are those specified by INDAA. The remaining elements of A contain the rows and columns not specified in INDAA

Optional Arguments

NAA — Order of the matrix AA. (Input)
Default: NAA = size (AA,2).

LDAA — Leading dimension of AA exactly as specified in the dimension statement in the calling program. (Input)
Default: LDAA = size (AA,1).

NA — Order of the reordered matrix A. (Input)
NA must be less than or equal to NAA.
Default: NA = size (INDAA,1).

LDA — Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

FORTRAN 90 Interface

Generic: CALL RORDM (AA, INDAA, A [, ...])
Specific: The specific interface names are S_RORDM and D_RORDM.

FORTRAN 77 Interface

Single: CALL RORDM (NAA, AA, LDAA, NA, INDAA, A, LDA)

Double: The double precision name is DRORDM.

Description

Routine RORDM reorders the rows and columns of a symmetric matrix. Frequently in practice a sum of squares and crossproducts matrix is first computed for all variables in a data set. Then, a sum of squares and crossproducts matrix is needed for some subset of the data set variables. Alternatively, a specific order for the selected variables may be required for input into an analysis routine. For example, in regression, IMSL routine RCOV requires the sum of squares and crossproducts matrix for the independent variables and the dependent variables. Sums of squares and crossproducts for the independent variables must appear first, followed by entries for the dependent variables. Variables not in the regression analysis, but in the data set, can appear last. RORDM can be used to reorder the sum of squares and crossproducts matrix for input to RCOV.

Comments

Workspace may be explicitly provided, if desired, by use of R2RDM/DR2RDM. The reference is:

CALL R2RDM (NAA, AA, LDAA, NA, INDAA, A, LDA, IWK)

The additional argument is

IWK — Work vector of length NAA indicating how the entire AA matrix has been reordered and returned in A. IWK(I) = J means the J-th row and column of AA are returned as the I-th row and column of A.

Example

A 4 x 4 symmetric matrix AA is reordered so that row/column 4, 3, and 1 of AA correspond to row/column 1, 2, and 3 of A, respectively.

```
      USE RORDM_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    LDA, LDAA, NA, NAA, J
      PARAMETER  (NA=3, NAA=4, LDA=NAA, LDAA=NAA)
!
      INTEGER    INDAA(NA)
      REAL       A(LDA,NAA), AA(LDAA,NAA)
!
      DATA (AA(1,J),J=1,NAA)/10., 20., 40., 70./
      DATA (AA(2,J),J=1,NAA)/20., 30., 50., 80./
      DATA (AA(3,J),J=1,NAA)/40., 50., 60., 90./
```

```
      DATA (AA(4,J),J=1,NAA)/70., 80., 90., 100./
      DATA INDAA/4, 3, 1/
!
      CALL RORDM (AA, INDAA, A)
      CALL WRRRN ('A', A)
      END
```

Output

```
                  A
          1       2       3       4
1   100.0    90.0    70.0    80.0
2    90.0    60.0    40.0    50.0
3    70.0    40.0    10.0    20.0
4    80.0    50.0    20.0    30.0
```

MVNAN

Moves any rows of a matrix with the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix.

Required Arguments

IIND — Index vector option. (Input)

IIND	Meaning
< 0	The first –IIND columns of X are checked for NaN.
> 0	The IIND columns of X given by IND are checked for NaN.

IND — Index vector of length IIND containing the column numbers of X that are to be checked for NaN. (Input if IIND is positive)
If IIND is negative, IND is not referenced and can be a vector of length one.

X — NROW by NCOL matrix whose rows are checked for NaN (not a number). (Input/Output)
On output, the rows of X containing NaN are the last NRMISS rows of X.

ISWP — Vector of length NROW specifying the rows that were exchanged (swapped). (Output)
The number of nonzero elements in ISWP is the number of swaps that took place. ISWP(I) = J (J greater than zero) means that rows I and J of X were swapped, i.e., row I of the input X is row J of the output X and row J of the input X is row I of the output X.

Optional Arguments

NROW — Number of rows. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

NRMISS — Number of rows that contained NaN in the specified columns of X. (Output)

FORTRAN 90 Interface

Generic: CALL MVNAN (IIND, IND, X, ISWP [, ...])

Specific: The specific interface names are S_MVNAN and D_MVNAN.

FORTRAN 77 Interface

Single: CALL MVNAN (NROW, NCOL, IIND, IND, X, LDX, ISWP, NRMISS)

Double: The double precision name is DMVNAN.

Examples

Example 1

In this example, MVNAN is used to move rows containing NaN in columns 1 and 2 of a 5 by 3 matrix X to the last rows.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    LDX, NCOL, NROW, J
          PARAMETER  (NCOL=3, NROW=5, LDX=NROW)
    !
          INTEGER    IIND, IND(1), ISWP(NROW), NOUT, NRMISS
          REAL       X(LDX,NCOL)
    !
          DATA (X(1,J),J=1,NCOL)/1.0, 10.0, 100.0/
          DATA (X(2,J),J=1,NCOL)/2.0, 20.0, 200.0/
          DATA (X(3,J),J=1,NCOL)/3.0, 30.0, 300.0/
          DATA (X(4,J),J=1,NCOL)/4.0, 40.0, 400.0/
          DATA (X(5,J),J=1,NCOL)/5.0, 50.0, 500.0/
    !
          X(2,2) = AMACH(6)
          X(4,1) = AMACH(6)
          IIND   = -2
          CALL WRRRN ('Input X', X)
          CALL MVNAN (IIND, IND, X, ISWP, NRMISS=NRMISS)
          CALL WRRRN ('Output X', X)
          CALL WRIRN ('ISWP', ISWP)
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) ' '
          WRITE (NOUT,*) 'NRMISS = ', NRMISS
          END
```

Output

```
            Input X
          1       2       3
    1     1.0    10.0   100.0
    2     2.0     NaN   200.0
    3     3.0    30.0   300.0
```

```
    4     NaN    40.0   400.0
    5     5.0    50.0   500.0

            Output X
            1       2       3
    1     1.0    10.0   100.0
    2     5.0    50.0   500.0
    3     3.0    30.0   300.0
    4     NaN    40.0   400.0
    5     2.0     NaN   200.0

    ISWAP
    1   0
    2   5
    3   0
    4   0
    5   0
    NRMISS =   2
```

Example 2

In this example, **MVNAN** is used to move rows containing NaN in column 1 and 3 of a 5 by 3 matrix **X** to the last rows.

```
          USE IMSL_LIBRARIES

          IMPLICIT   NONE
          INTEGER    LDX, NCOL, NROW, J
          PARAMETER  (NCOL=3, NROW=5, LDX=NROW)
    !
          INTEGER    IIND, IND(2), ISWP(NROW), NOUT, NRMISS
          REAL       X(LDX,NCOL)
    !
          DATA (X(1,J),J=1,NCOL)/1.0, 10.0, 100.0/
          DATA (X(2,J),J=1,NCOL)/2.0, 20.0, 200.0/
          DATA (X(3,J),J=1,NCOL)/3.0, 30.0, 300.0/
          DATA (X(4,J),J=1,NCOL)/4.0, 40.0, 400.0/
          DATA (X(5,J),J=1,NCOL)/5.0, 50.0, 500.0/
          DATA IND/1, 3/
    !
          X(2,2) = AMACH(6)
          X(4,1) = AMACH(6)
          IIND   = 2
          CALL WRRRN ('Input X', X)
          CALL MVNAN (IIND, IND, X, ISWP, NRMISS=NRMISS)
          CALL WRRRN ('Output X', X)
          CALL WRIRN ('ISWP', ISWP)
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) ' '
          WRITE (NOUT,*) 'NRMISS = ', NRMISS
          END
```

Output

```
             Input X
            1       2       3
    1     1.0    10.0   100.0
```

```
     2     2.0     NaN   200.0
     3     3.0    30.0   300.0
     4     NaN    40.0   400.0
     5     5.0    50.0   500.0

           Output X
           1       2       3
     1     1.0    10.0   100.0
     2     2.0     NaN   200.0
     3     3.0    30.0   300.0
     4     5.0    50.0   500.0
     5     NaN    40.0   400.0
```

```
     ISWP
     1   0
     2   0
     3   0
     4   5
     5   0
     NRMISS =   1
```

SVRGN

Sorts a real array by algebraically increasing value.

Required Arguments

RA — Vector of length **N** containing the array to be sorted. (Input)

RB — Vector of length **N** containing the sorted array. (Output)
If **RA** is not needed, **RA** and **RB** can share the same storage locations.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: **N** = size (**RA**,1).

FORTRAN 90 Interface

Generic: **CALL SVRGN** (**RA**, **RB** [, ...])
Specific: The specific interface names are **S_SVRGN** and **D_SVRGN**.

FORTRAN 77 Interface

Single: **CALL SVRGN** (**N**, **RA**, **RB**)
Double: The double precision name is **DSVRGN**.

Description

Routine **SVRGN** sorts the elements of an array, *A*, into ascending order by algebraic value. The array *A* is divided into two parts by picking a central element *T* of the array. The first and last elements of *A* are compared with *T* and exchanged until the three values appear in the array in ascending order. The elements of the array are rearranged until all elements greater than or equal to the central element appear in the second part of the array and all those less than or equal to the central element appear in the first part. The upper and lower subscripts of one of the segments are saved, and the process continues iteratively on the other segment. When one segment is finally sorted, the process begins again by retrieving the subscripts of another unsorted portion of the array. On completion, $A_j \leq A_i$ for $j < i$. For more details, see Singleton (1969), Griffin and Redish (1970), and Petro (1970).

Example

This example sorts the 10-element array **RA** algebraically.

```
      USE SVRGN_INT
      USE UMACH_INT

      IMPLICIT   NONE
!                                Declare variables
      INTEGER    J
      PARAMETER  (N=10)
      REAL       RA(N), RB(N)
!                                Set values for  RA
!     RA = ( -1.0  2.0  -3.0  4.0  -5.0  6.0  -7.0  8.0  -9.0  10.0 )
!
      DATA RA/-1.0, 2.0, -3.0, 4.0, -5.0, 6.0, -7.0, 8.0, -9.0, 10.0/
!                                Sort RA by algebraic value into RB
      CALL SVRGN (RA, RB)
!                                Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99999) (RB(J),J=1,N)
!
99999 FORMAT (’  The output vector is:’, /, 10(1X,F5.1))
      END
```

Output

```
 The Output vector is:
 -9.0  -7.0  -5.0  -3.0  -1.0   2.0   4.0   6.0   8.0  10.0
```

SVRGP

Sorts a real array by algebraically increasing value and return the permutation that rearranges the array.

Required Arguments

RA — Vector of length **N** containing the array to be sorted. (Input)

RB — Vector of length **N** containing the sorted array. (Output)
If **RA** is not needed, **RA** and **RB** can share the same storage locations.

IPERM — Vector of length **N**. (Input/Output)
On input, **IPERM** should be initialized to the values 1, 2, ..., **N**. On output, **IPERM** contains a record of permutations made on the vector **RA**.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: **N** = size (**IPERM**,1).

FORTRAN 90 Interface

Generic: **CALL SVRGP** (**RA**, **RB**, **IPERM** [, ...])
Specific: The specific interface names are **S_SVRGP** and **D_SVRGP**.

FORTRAN 77 Interface

Single: **CALL SVRGP** (**N**, **RA**, **RB**, **IPERM**)
Double: The double precision name is **DSVRGP**.

Description

Routine **SVRGP** sorts the elements of an array, *A*, into ascending order by algebraic value, keeping a record in *P* of the permutations to the array *A*. That is, the elements of *P* are moved in the same manner as are the elements in *A* as *A* is being sorted. The routine **SVRGP** uses the algorithm discussed in SVRGN. On completion, $A_j \le A_i$ for $j < i$.

Comments

For wider applicability, integers (1, 2, …, **N**) that are to be associated with **RA**(**I**) for **I** = 1, 2, …, **N** may be entered into **IPERM**(**I**) in any order. Note that these integers must be unique.

Example

This example sorts the 10-element array **RA** algebraically.

```
      USE SVRGP_INT
      USE UMACH_INT

      IMPLICIT   NONE
!                                Declare variables
      INTEGER    N, J, NOUT
      PARAMETER  (N=10)
      REAL       RA(N), RB(N)
      INTEGER    IPERM(N)
!                                Set values for  RA and IPERM
!     RA    = ( 10.0  -9.0  8.0  -7.0  6.0  5.0  4.0  -3.0  -2.0  -1.0 )
!
!     IPERM = ( 1  2  3  4  5  6  7  8  9  10)
!
      DATA RA/10.0, -9.0, 8.0, -7.0, 6.0, 5.0, 4.0, -3.0, -2.0, -1.0/
      DATA IPERM/1, 2, 3, 4, 5, 6, 7, 8, 9, 10/
!                                Sort RA by algebraic value into RB
      CALL SVRGP (RA, RB, IPERM)
!                                Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99998) (RB(J),J=1,N)
      WRITE (NOUT, 99999) (IPERM(J),J=1,N)
!
99998 FORMAT ('  The output vector is:', /, 10(1X,F5.1))
99999 FORMAT ('  The permutation vector is:', /, 10(1X,I5))
      END
```

Output

```
 The output vector is:
 -9.0  -7.0  -3.0  -2.0  -1.0   4.0   5.0   6.0   8.0  10.0

 The permutation vector is:
 2     4     8     9    10     7     6     5     3     1
```

SVIGN

Sorts an integer array by algebraically increasing value.

Required Arguments

IA — Integer vector of length **N** containing the array to be sorted. (Input)

IB — Integer vector of length **N** containing the sorted array. (Output)
If **IA** is not needed, **IA** and **IB** can share the same storage locations.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: **N** = size (**IA**,1).

FORTRAN 90 Interface

Generic: **CALL SVIGN** (**IA**, **IB** [, ...])
Specific: The specific interface name is **S_SVIGN** .

FORTRAN 77 Interface

Single: **CALL SVIGN** (**N**, **IA**, **IB**)

Description

Routine **SVIGN** sorts the elements of an integer array, A, into ascending order by algebraic value. The routine **SVIGN** uses the algorithm discussed in **SVRGN**. On completion, $A_j \leq A_i$ for $j < i$.

Example

This example sorts the 10-element array **IA** algebraically.

```
      USE SVIGN_INT
      USE UMACH_INT

      IMPLICIT   NONE
```

```
      !                                  Declare variables
            INTEGER    N, J, NOUT
            PARAMETER  (N=10)
            INTEGER    IA(N), IB(N)
      !                                  Set values for  IA
      !     IA = ( -1  2  -3  4  -5  6  -7  8  -9  10 )
      !
            DATA IA/-1, 2, -3, 4, -5, 6, -7, 8, -9, 10/
      !                                  Sort IA by algebraic value into IB
            CALL SVIGN (IA, IB)
      !                                  Print results
            CALL UMACH (2,NOUT)
            WRITE (NOUT, 99999) (IB(J),J=1,N)
      !
      99999 FORMAT ('  The output vector is:', /, 10(1X,I5))
            END
```

Output

```
      The Output vector is:
      -9    -7    -5    -3    -1     2     4     6     8    10
```

SVIGP

Sorts an integer array by algebraically increasing value and return the permutation that rearranges the array.

Required Arguments

IA — Integer vector of length **N** containing the array to be sorted. (Input)

IB — Integer vector of length **N** containing the sorted array. (Output)
If **IA** is not needed, **IA** and **IB** can share the same storage locations.

IPERM — Vector of length **N**. (Input/Output)
On input, **IPERM** should be initialized to the values 1, 2, ..., **N**. On output, **IPERM** contains a record of permutations made on the vector **IA**.

Optional Arguments

N — Number of elements in the array to be sorted. (Input)
Default: **N** = size (**IPERM**,1).

FORTRAN 90 Interface

Generic: **CALL SVIGP** (**IA**, **IB**, **IPERM** [, ...])
Specific: The specific interface name is **S_SVIGP**.

FORTRAN 77 Interface

Single: **CALL SVIGP** (**N**, **IA**, **IB**, **IPERM**)

Description

Routine **SVIGP** sorts the elements of an integer array, *A*, into ascending order by algebraic value, keeping a record in *P* of the permutations to the array *A*. That is, the elements of *P* are moved in the same manner as are the elements in *A* as *A* is being sorted. The routine **SVIGP** uses the algorithm discussed in **SVRGN**. On completion, $A_j \leq A_i$ for $j < i$.

Comments

For wider applicability, integers (1, 2, …, **N**) that are to be associated with **IA**(**I**) for **I** = 1, 2, …, **N** may be entered into **IPERM**(**I**) in any order. Note that these integers must be unique.

Example

This example sorts the 10-element array **IA** algebraically.

```
      USE SVIGP_INT
      USE UMACH_INT

      IMPLICIT   NONE
!                                Declare variables
      INTEGER    N, J, NOUT
      PARAMETER  (N=10)
      INTEGER    IA(N), IB(N), IPERM(N)
!                                Set values for  IA and IPERM
!     IA    = ( 10  -9  8  -7  6  5  4  -3  -2  -1 )
!
!     IPERM = ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 )
!
      DATA IA/10, -9, 8, -7, 6, 5, 4, -3, -2, -1/
      DATA IPERM/1, 2, 3, 4, 5, 6, 7, 8, 9, 10/
!                                Sort IA by algebraic value into IB
      CALL SVIGP (IA, IB, IPERM)
!                                Print results
      CALL UMACH (2,NOUT)
      WRITE (NOUT, 99998) (IB(J),J=1,N)
      WRITE (NOUT, 99999) (IPERM(J),J=1,N)
!
99998 FORMAT (' The output vector is:', /, 10(1X,I5))
99999 FORMAT (' The permutation vector is:', /, 10(1X,I5))
      END
```

Output

```
 The Output vector is:
 -9    -7    -3    -2    -1     4     5     6     8    10

 The permutation vector is:
 2     4     8     9    10     7     6     5     3     1
```

SCOLR

Sorts columns of a real rectangular matrix using keys in rows.

Required Arguments

X — NRX by NCX matrix. (Input, if IRET = 1; input/output if IRET = 0)
On input, X contains the matrix to be sorted. If IRET = 0, the output X contains the sorted matrix.

INDKEY — Vector of length NKEY giving the row numbers of X which are to be used in the sort. (Input)

IPERM — Permutation vector of length NCX specifying the rearrangement of the columns. (Output)
IPERM (I) = J means column I of the sorted X is column J of the unsorted X.

NGROUP — Number of groups. (Output)
The columns of the sorted X are partitioned into groups. A group contains columns that are equal with respect to the method of comparison. NGROUP is the number of groups of different columns.

NI — Vector of length NGROUP containing the number of columns in each group. (Output)
The first NI(1) columns of the sorted X are group number 1; the next NI(2) columns of the sorted X are group number 2; ... the last NI(NGROUP) columns of the sorted X are group number NGROUP. If NGROUP is not known prior to the invocation of this routine, NCX(an upper bound for NGROUP) can be used as the dimension of NI.

Optional Arguments

NRX — Number of rows of X. (Input)
Default: NRX = size (X,1).

NCX — Number of columns of X. (Input)
Default: NCX = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

ICOMP — Option giving the method of comparison of the column vectors. (Input)
Default: ICOMP = 0.

ICOMP	Action
0	Elementwise, by algebraic values
1	Elementwise, by absolute values

IORDR — Option giving the sorting order. (Input)
Default: IORDR = 0.

IORDR	Action
0	Ascending
1	Descending

IRET — Option for determining whether the columns of X are to be permuted. (Input)
Default: IRET = 0.

IRET	Action
0	The columns of X are sorted.
1	X is unchanged (detached key sort).

NKEY — Number of rows of X on which to sort. (Input)
Default: NKEY = size (INDKEY,1).

FORTRAN 90 Interface

Generic: CALL SCOLR (X, INDKEY, IPERM, NGROUP, NI [, ...])
Specific: The specific interface names are S_SCOLR and D_SCOLR.

FORTRAN 77 Interface

Single: CALL SCOLR (NRX, NCX, X, LDX, ICOMP, IORDR, IRET, NKEY, INDKEY, IPERM, NGROUP, NI)
Double: The double precision name is DSCOLR.

Description

Routine SCOLR sorts the columns of a real matrix *X* using particular rows in *X* as the keys. One of two methods for comparing the columns can be used for sorting.

1. Algebraic with the first key as the most significant, the second key next most significant and so forth.

2. Absolute values with the first key as the most significant, the second key next most significant and so forth.

The columns of *X* can be put in ascending or descending order.

The routine is useful for data containing classification variables. Routine **CSTAT** (see Chapter 1, "Basic Statistics") can be used to form the cells and frequency counts for a multi-way table from data. The columns of the output matrix contain the values of each combination of values of the classification variables along with the tallies. **SCOLR** can then be used to sort the columns of this output matrix using the classification variables as keys.

SCOLR is based on a quicksort method given by Singleton (1969). Modifications by Griffin and Redish (1970) and Petro (1970) are incorporated.

Comments

1. Workspace may be explicitly provided, if desired, by use of **S2OLR/DS2OLR**. The reference is:

   CALL S2OLR (**NRX**, **NCX**, **X**, **LDX**, **ICOMP**, **IORDR**, **IRET**, **NKEY**, **INDKEY**, **IPERM**, **NGROUP**, **NI**, **WK**, **IWK**)

   The additional arguments are as follows:

   WK — Work vector of length 2 * *m*.

   IWK — Work vector of length *m* + **INT**(2.8854 ln(*m*)) + 2.

2. When **X** is sorted by algebraic value (**ICOMP** = 0) in ascending order, the resulting array **X** is such that:
   For *i* = 1, 2, ..., **NCX** – 1, **X**(**INDKEY**(1), *i*) ≤ **X**(**INDKEY**(1), *i* + 1)
   For *k* = 2, ..., **NKEY**, if **X**(**INDKEY**(*j*), *i*) = **X**(**INDKEY**(*j*), *i* + 1) for *j* = 1, 2, ..., *k* - 1, then
   X(**INDKEY** (*k*), *i*) ≤ **X**(**INDKEY**(*k*), *i* + 1).
   When **ICOMP** = 1, the absolute values are compared instead.

Example

The columns of a 5 x 10 matrix *X* are sorted in descending order by absolute value using rows 1, 2, 3, and 5 as the keys. The permutations to put the columns of *X* in order are returned. The input matrix *X* is not changed.

```
          USE SCOLR_INT
          USE WRRRL_INT
          USE WRIRL_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    LDX, NCX, NKEY, NRX
          PARAMETER  (NCX=10, NKEY=4, NRX=5, LDX=NRX)
    !
          INTEGER    ICOMP, INDKEY(NKEY), IORDR, IPERM(NCX), IRET, NI(NCX), &
```

```
                  NGROUP, NOUT
      REAL         X(LDX,NCX)
      CHARACTER    CLABEL(1)*10, FMT*10, RLABEL(1)*23
!
      DATA CLABEL(1)/'NONE'/, RLABEL(1)/'NONE'/
      DATA X/-1.0, -10.0, -11.0, 10.0, -1.0, 2.0, 20.0, 22.0, -20.0, &
           -2.0, -3.0, -30.0, 33.0, 30.0, -3.0, 4.0, 40.0, 44.0, &
           -40.0, -4.0, -5.0, -50.0, 55.0, 50.0, -5.0, -1.0, 60.0, &
           -66.0, -60.0, 6.0, 2.0, -70.0, -77.0, 70.0, 7.0, -3.0, &
           -30.0, -88.0, 80.0, 8.0, 4.0, 40.0, -99.0, -90.0, 9.0, &
           -5.0, -50.0, -100.0, 100.0, 10.0/
      DATA INDKEY/1, 2, 3, 5/
!
      ICOMP = 1
      IORDR = 1
      IRET  = 1
      CALL SCOLR (X, INDKEY, IPERM, NGROUP, NI, ICOMP=ICOMP, &
                  IORDR=IORDR, IRET=IRET)
!
      FMT       = '(F6.1)'
      RLABEL(1) = 'NONE'
      CALL WRRRL ('X', X, RLABEL, CLABEL)
!
      FMT       = '(I4)'
      RLABEL(1) = 'IPERM = '
      CALL WRIRL ('%/', IPERM, RLABEL, CLABEL, 1, NCX, 1, FMT='(I4)')
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*)
      WRITE (NOUT,*) 'NGROUP = ', NGROUP
!
      RLABEL(1) = 'NI = '
      CALL WRIRL ('%/', NI, RLABEL, CLABEL, 1, NGROUP, 1, FMT='(I4)')
!
      END
```

Output

```
                                    X
 -1.0    2.0   -3.0     4.0    -5.0    -1.0     2.0    -3.0     4.0    -5.0
-10.0   20.0  -30.0    40.0   -50.0    60.0   -70.0   -30.0    40.0   -50.0
-11.0   22.0   33.0    44.0    55.0   -66.0   -77.0   -88.0   -99.0  -100.0
 10.0  -20.0   30.0   -40.0    50.0   -60.0    70.0    80.0   -90.0   100.0
 -1.0   -2.0   -3.0    -4.0    -5.0     6.0     7.0     8.0     9.0    10.0
IPERM =    10     5     9     4     8     3     7     2     6     1
NGROUP =   10
NI =     1     1     1     1     1     1     1     1     1     1
```

SROWR

Sorts rows of a real rectangular matrix using keys in columns.

Required Arguments

X — NROW by NCOL matrix. (Input, if IRET = 1; input/output if IRET = 0)
On input, X contains the matrix to be sorted. If IRET = 0, the output X contains the sorted matrix.

INDKEY — Vector of length NKEY giving the column numbers of X which are to be used in the sort. (Input)

IPERM — Permutation vector of length NROW specifying the rearrangement of the rows. (Output)
IPERM(I) = J means row I of the sorted X is row J of the unsorted X.

NGROUP — Number of groups. (Output, if IRET ≤ 1)
The rows of the sorted X are partitioned into groups. A group contains rows that are equal with respect to the method of comparison. NGROUP is the number of groups of different rows.

NI — Vector of length NGROUP containing the number of rows in each group. (Output, if IRET ≤ 1)
The first NI(1) rows of the sorted X are group number 1. The next NI(2) rows of the sorted X are group number 2. ... The last NI(NGROUP) rows of the sorted X are group number NGROUP. If NGROUP is not known prior to the invocation of this routine, NROW(an upper bound for NGROUP) can be used as the dimension of NI. If IRET ≥ 2, NI is not referenced and can be a vector of length one.

Optional Arguments

NROW — Number of rows of X. (Input)
Default: NROW = size (X,1).

NCOL — Number of columns of X. (Input)
Default: NCOL = size (X,2).

LDX — Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

ICOMP — Option giving the method of comparison of the row vectors. (Input)
Default: ICOMP = 0.

ICOMP	Action
0	Elementwise, by algebraic values
1	Elementwise, by absolute values

IORDR — Option giving the sorting order. (Input)
Default: IORDR = 0.

IORDR	Action
0	Ascending
1	Descending

IRET — Option to indicate information returned. (Input)
Default: IRET = 0.

IRET	Action
0	The sorted X is returned along with NGROUP and NI.
1	X is unchanged (detached key sort) and NGROUP and NI are returned.
2	The sorted X is returned, but NGROUP and NI are not returned.
3	X is unchanged (detached key sort) and NGROUP and NI are not returned.

NKEY — Number of columns of X on which to sort. (Input)
Default: NKEY = size (INDKEY,1).

NRMISS — Number of rows that contained NaN in the columns of X used in the sort. (Output)
These rows are considered as a separate group from the other NGROUP groups and are put as the last NRMISS rows of the sorted X.

FORTRAN 90 Interface

Generic: CALL SROWR (X, INDKEY, IPERM, NGROUP, NI [, ...])
Specific: The specific interface names are S_SROWR and D_SROWR.

FORTRAN 77 Interface

Single: CALL SROWR (NROW, NCOL, X, LDX, ICOMP, IORDR, IRET, NKEY, INDKEY, IPERM, NGROUP, NI, NRMISS)
Double: The double precision name is DSROWR.

Description

Routine SROWR sorts the rows of a real matrix X using particular rows in X as the keys. One of two methods for comparing the rows can be used for sorting.

1. Algebraic with the first key as the most significant, the second key next most significant and so forth.
2. Absolute values with the first key as the most significant, the second key next most significant and so forth.

The rows of X can be put in ascending or descending order.

The routine is useful for grouping data based on values of specified variables. The rows of X containing the IMSL missing value code NaN (not a number) in at least one of the specified columns are considered as an additional group of NRMISS rows. These rows are moved to the end of the sorted X. SROWR is based on a quicksort method given by Singleton (1969). Modifications by Griffin and Redish (1970) and Petro (1970) are incorporated.

Comments

1. Workspace may be explicitly provided, if desired, by use of S2OWR/DS2OWR. The reference is:

   CALL S2OWR (NROW, NCOL, X, LDX, ICOMP, IORDR, IRET, NKEY, INDKEY, IPERM, NGROUP, NI, NRMISS, WK, IWK)

   The additional arguments are as follows:

   WK — Work vector of length 2 * *m*.

   IWK — Work vector of length *m* + INT(2.8854 * ln(*m*)) + 2.

2. When X is sorted by algebraic values (ICOMP = 0), in ascending order, the resulting array X is such that:
   For *i* = 1, 2, …, NROW – 1, X(*i*, INDKEY(1)) ≤ X(*i* + 1, INDKEY(1)).
   For k = 2, …, NKEY, if X(*i*, INDKEY(*j*)) = X(*i* + 1, INDKEY(*j*)) for *j* = 1, 2, …, *k* – 1; then
   X(*i*, INDKEY(*k*)) ≤ X(*i* + 1, INDKEY(*k*)).
   When ICOMP = 1, the absolute values are compared instead.

Example

The rows of a 10 x 3 matrix X are sorted in ascending order by algebraic value using columns 2 and 3 as the keys. The permutations to put the rows of the input X into sorted order are returned along with the sorted X.

```
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDX, NCOL, NKEY, NROW, J
```

```
      PARAMETER  (NCOL=3, NKEY=2, NROW=10, LDX=NROW)
!
      INTEGER    ICOMP, INDKEY(NKEY), IORDR, IPERM(NROW), IRET, &
                 NGROUP, NI(NROW), NOUT, NRMISS
      REAL       X(LDX,NCOL)
!
      DATA (X(1,J),J=1,3)/1.0, 1., 1./
      DATA (X(2,J),J=1,3)/2.0, 2., 1./
      DATA (X(3,J),J=1,3)/3.0, 1., 1./
      DATA (X(4,J),J=1,3)/4.0, 1., 1./
      DATA (X(5,J),J=1,3)/5.0, 2., 2./
      DATA (X(6,J),J=1,3)/6.0, 1., 2./
      DATA (X(7,J),J=1,3)/7.0, 1., 2./
      DATA (X(8,J),J=1,3)/8.0, 1., 1./
      DATA (X(9,J),J=1,3)/9.0, 2., 2./
      DATA (X(10,J),J=1,3)/9.0, 1., 1./
      DATA INDKEY/2, 3/
!
      X(5,3) = AMACH(6)
      X(7,2) = AMACH(6)
      CALL SROWR (X, INDKEY, IPERM, NGROUP, NI, NRMISS=NRMISS)
      CALL WRRRN ('X', X)
      CALL WRIRN ('IPERM', IPERM)
      CALL WRIRN ('NI', NI, NGROUP, 1, NGROUP)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) ' '
      WRITE (NOUT,*) 'NRMISS = ', NRMISS
      END
```

Output

```
                X
          1       2       3
 1    1.000   1.000   1.000
 2    9.000   1.000   1.000
 3    3.000   1.000   1.000
 4    4.000   1.000   1.000
 5    8.000   1.000   1.000
 6    6.000   1.000   2.000
 7    2.000   2.000   1.000
 8    9.000   2.000   2.000
 9    7.000     NaN   2.000
10    5.000   2.000     NaN

 IPERM
 1    1
 2   10
 3    3
 4    4
 5    8
 6    6
 7    2
 8    9
 9    7
10    5

  NI
 1   5
```

```
      2   1
      3   1
      4   1
     NRMISS =   2
```

SRCH

Searches a sorted vector for a given scalar and return its index.

Required Arguments

VALUE — Scalar to be searched for in Y. (Input)

X — Vector of length N * INCX. (Input)
Y is obtained from X for I = 1, 2, ..., N by Y(I) = X(1 + (I - 1) * INCX). Y(1), Y(2), ..., Y(N) must be in ascending order.

INDEX — Index of Y pointing to VALUE. (Output)
If INDEX is positive, VALUE is found in Y. If INDEX is negative, VALUE is not found in Y.

INDEX	Location of VALUE
1 thru N	VALUE = Y(INDEX)
-1	VALUE < Y(1) or N = 0
-N thru -2	Y(-INDEX - 1) < VALUE < Y(INDEX)
-(N + 1)	VALUE > YN

Optional Arguments

N — Length of vector Y. (Input)
Default: N = (size (X,1)) / INCX.

INCX — Displacement between elements of X. (Input)
INCX must be greater than zero.
Default: INCX = 1.

FORTRAN 90 Interface

Generic: CALL SRCH (VALUE, X, INDEX [, ...])
Specific: The specific interface names are S_SRCH and D_SRCH.

FORTRAN 77 Interface

Single: CALL SRCH (N, VALUE, X, INCX, INDEX)

Double: The double precision name is **DSRCH**.

Description

Routine **SRCH** searches a real vector *x* (stored in **X**), whose *n* elements are sorted in ascending order for a real number *c* (stored in **VALUE**). If *c* is found in *x*, its index *i* (stored in **INDEX**) is returned so that $x_i = c$. Otherwise, a negative number *i* is returned for the index. Specifically,

if $1 \leq i \leq n$	then $x_i = c$
if $i = -1$	then $c < x_1$ or $n = 0$
if $-n \leq i \leq -2$	then $x_{-i-1} < c < x_{-i}$
if $i = -(n + 1)$	then $c > x_n$

The argument **INCX** is useful if a row of a matrix, for example, row number **I** of a matrix **X**, must be searched. The elements of row **I** are assumed to be in ascending order. In this case, set **INCX** equal to the leading dimension of **X** exactly as specified in the dimension statement in the calling program. With **X** declared

```
REAL  X(LDX,N)
```

the invocation

```
CALL SRCH (VALUE, X(I, 1), INDEX, N=N, INCX=LDX)
```

returns an index that will reference a column number of **X**.

Routine **SRCH** performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407-411).

Example

This example searches a real vector sorted in ascending order for the value 653.0. The problem is discussed by Knuth (1973, pages 407-409).

```
      USE SRCH_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    N
      PARAMETER  (N=16)
!
      INTEGER    INDEX, NOUT
      REAL       VALUE, X(N)
!
      DATA X/61.0, 87.0, 154.0, 170.0, 275.0, 426.0, 503.0, 509.0, &
```

```
                512.0, 612.0, 653.0, 677.0, 703.0, 765.0, 897.0, 908.0/
      !
            VALUE = 653.0
            CALL SRCH (VALUE, X, INDEX)
      !
            CALL UMACH (2, NOUT)
            WRITE (NOUT,*) 'INDEX = ', INDEX
            END
```

Output

```

      INDEX =   11
```

ISRCH

Searches a sorted integer vector for a given integer and return its index.

Required Arguments

IVALUE — Scalar to be searched for in `IY`. (Input)

IX — Vector of length `N * INCX`. (Input)
`IY` is obtained from `IX` for `I` = 1, 2, ..., `N` by `IY(I)` = `IX`(1 + (`I - 1`) * `INCX`). `IY`(1), `IY`(2), ..., `IY`(`N`) must be in ascending order.

INDEX — Index of `IY` pointing to `IVALUE`. (Output)
If `INDEX` is positive, `IVALUE` is found in `IY`. If `INDEX` is negative, `IVALUE` is not found in `IY`.

`INDEX`	Location of `VALUE`
1 thru `N`	`IVALUE` = `IY(INDEX)`
`-1`	`IVALUE` < `IY`(1) or `N` = 0
`-N` thru `-2`	`IY(-INDEX - 1)` < `IVALUE` < `IY (- INDEX)`
`-(N + 1)`	`IVALUE`> `Y(N)`

Optional Arguments

N — Length of vector `IY`. (Input)
Default: `N` = size (`IX`,1) / `INCX`.

INCX — Displacement between elements of `IX`. (Input)
`INCX` must be greater than zero.
Default: `INCX` = 1.

FORTRAN 90 Interface

Generic: `CALL ISRCH` (`IVALUE`, `IX`, `INDEX` [, ...])
Specific: The specific interface name is `S_ISRCH`.

FORTRAN 77 Interface

Single: `CALL ISRCH` (`N`, `IVALUE`, `IX`, `INCX`, `INDEX`)

Description

Routine **ISRCH** searches an integer vector *x* (stored in **IX**), whose *n* elements are sorted in ascending order for an integer *c* (stored in **IVALUE**). If *c* is found in *x*, its index *i* (stored in **INDEX**) is returned so that $x_i = c$. Otherwise, a negative number *i* is returned for the index. Specifically,

if $1 \le i \le n$	Then $x_i = c$
if $i = -1$	Then $c < x_1$ or $n = 0$
if $-n \le i \le -2$	Then $x-i-1 < c < x-i$
if $i = -(n + 1)$	Then $c > x_n$

The argument **INCX** is useful if a row of a matrix, for example, row number **I** of a matrix **IX**, must be searched. The elements of row **I** are assumed to be in ascending order. Here, set **INCX** equal to the leading dimension of **IX** exactly as specified in the dimension statement in the calling program. With **IX** declared

```
INTEGER IX(LDIX,N)
```

the invocation

```
CALL ISRCH (VALUE, X(I, 1), INDEX, N=N, INCX=LDIX)
```

returns an index that will reference a column number of **IX**.

The routine **ISRCH** performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407-411).

Example

This example searches an integer vector sorted in ascending order for the value 653. The problem is discussed by Knuth (1973, pages 407-409).

```
      USE ISRCH_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    N
      PARAMETER  (N=16)
!
      INTEGER    INDEX, NOUT
      INTEGER    IVALUE, IX(N)
!
      DATA IX/61, 87, 154, 170, 275, 426, 503, 509, 512, 612, 653, 677, &
             703, 765, 897, 908/
!
      IVALUE = 653
      CALL ISRCH (IVALUE, IX, INDEX)
```

```
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'INDEX = ', INDEX
      END
```

Output

```
 INDEX =   11
```

SSRCH

Searches a character vector, sorted in ascending ASCII order, for a given string and return its index.

Required Arguments

N — Length of vector **CHY**. (Input)
Default: **N** = size (**CHX**,1) / **INCX**.

STRING — Character string to be searched for in **CHY**. (Input)

CHX — Vector of length **N** * **INCX** containing character strings. (Input)
CHY is obtained from **CHX** for **I** = 1, 2, ..., **N** by **CHY**(**I**) = **CHX**(1 + (**I** - 1) * **INCX**). **CHY**(1), **CHY**(2), ..., **CHY**(**N**) must be in ascending ASCII order.

INCX — Displacement between elements of **CHX**. (Input)
INCX must be greater than zero.
Default: **INCX** = 1.

INDEX — Index of **CHY** pointing to **STRING**. (Output)
If **INDEX** is positive, **STRING** is found in **CHY**. If **INDEX** is negative, **STRING** is not found in **CHY**.

INDEX	**Location of STRING**
1 thru N	STRING = CHY(INDEX)
-1	STRING < CHY(1) or N = 0
-N thru -2	CHY(-INDEX - 1) < STRING < CHY(-INDEX)
-(N + 1)	STRING > CHY(N)

FORTRAN 90 Interface

Generic: CALL SSRCH (N, STRING, CHX, INCX, INDEX)
Specific: The specific interface name is SSRCH.

FORTRAN 77 Interface

Single: CALL SSRCH (N, STRING, CHX, INCX, INDEX)

Description

Routine **SSRCH** searches a vector of character strings *x* (stored in **CHX**), whose *n* elements are sorted in ascending ASCII order, for a character string *c* (stored in **STRING**). If *c* is found in *x*, its index *i* (stored in **INDEX**) is returned so that $x_i = c$. Otherwise, a negative number *i* is returned for the index. Specifically,

if $1 \le i \le n$	Then $x_i = c$
if $i = -1$	Then $c < x_1$ or $n = 0$
if $-n \le i \le -2$	Then $x_{-i-1} < c < x_{-i}$
if $i = -(n + 1)$	Then $c > x_n$

Here, "<" and ">" are in reference to the ASCII collating sequence. For comparisons made between character strings *c* and x_i with different lengths, the shorter string is considered as if it were extended on the right with blanks to the length of the longer string. (**SSRCH** uses FORTRAN intrinsic functions **LLT** and **LGT**.)

The argument **INCX** is useful if a row of a matrix, for example, row number **I** of a matrix **CHX**, must be searched. The elements of row **I** are assumed to be in ascending ASCII order. In this case, set **INCX** equal to the leading dimension of **CHX** exactly as specified in the dimension statement in the calling program. With **CHX** declared

```
CHARACTER * 7 CHX(LDCHX,N)
```

the invocation

```
CALL SSRCH (STRING, CHX(I:,1), INDEX, N=N, INCX=LDCHX)
```

returns an index that will reference a column number of **CHX**.

Routine **SSRCH** performs a binary search. The routine is an implementation of algorithm *B* discussed by Knuth (1973, pages 407-411).

Example

This example searches a **CHARACTER *** 2 vector containing 9 character strings, sorted in ascending ASCII order, for the value **'CC'**.

```
          USE SSRCH_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    N, INCX
          PARAMETER  (N=9)

    !
          INTEGER    INDEX, NOUT
```

```
      CHARACTER  CHX(N)*2, STRING*2
!
      DATA CHX/'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'GG', 'HH', &
          'II'/
!
      INCX   = 1
      STRING = 'CC'
      CALL SSRCH (N, STRING, CHX, INCX, INDEX)
!
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'INDEX = ', INDEX
      END
```

Output

```
 INDEX =   3
```

ACHAR

This function returns a character given its ASCII value.

Function Return Value

ACHAR — CHARACTER * 1 string containing the character in the **I**-th position of the ASCII collating sequence. (Output)

Required Arguments

I — Integer ASCII value of the character desired. (Input)
I must be greater than or equal to zero and less than or equal to 127.

FORTRAN 90 Interface

Generic: **ACHAR** (**I**)

Specific: The specific interface name is **ACHAR**.

FORTRAN 77 Interface

Single: **ACHAR** (**I**)

Description

Routine **ACHAR** returns the character of the input ASCII value. The input value should be between 0 and 127. If the input value is out of range, the value returned in **ACHAR** is machine dependent.

Example

This example returns the character of the ASCII value 65.

```
      USE ACHAR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    I, NOUT
!
```

```
      CALL UMACH (2, NOUT)
!                                Get character for ASCII value
!                                of 65 ('A')
      I = 65
      WRITE (NOUT,99999) I, ACHAR(I)
!
99999 FORMAT (' For the ASCII value of ', I2, ', the character is : ', &
            A1)
      END
```

Output

```
 For the ASCII value of 65, the character is : A
```

IACHAR

This function returns the integer ASCII value of a character argument.

Function Return Value

IACHAR — Integer ASCII value for **CH**. (Output)
The character **CH** is in the **IACHAR**-th position of the ASCII collating sequence.

Required Arguments

CH — Character argument for which the integer ASCII value is desired. (Input)

FORTRAN 90 Interface

Generic: **IACHAR** (**CH**)
Specific: The specific interface name is **IACHAR**.

FORTRAN 77 Interface

Single: **IACHAR** (**CH**)

Description

Routine **IACHAR** returns the ASCII value of the input character.

Example

This example gives the ASCII value of character A.

```
      USE IACHAR_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      CHARACTER  CH
!
      CALL UMACH (2, NOUT)
!                                Get ASCII value for the character
!                                'A'.
      CH = 'A'
```

```
      WRITE (NOUT,99999) CH, IACHAR(CH)
!
99999 FORMAT (' For the character  ', A1, '  the ASCII value is : ', &
             I3)
      END
```

Output

```
 For the character  A  the ASCII value is :  65
```

ICASE

This function returns the ASCII value of a character converted to uppercase.

Function Return Value

ICASE — Integer ASCII value for **CH** without regard to the case of **CH**. (Output)
Routine **ICASE** returns the same value as **IACHAR** for all but lowercase letters. For these, it returns the **IACHAR** value for the corresponding uppercase letter.

Required Arguments

CH — Character to be converted. (Input)

FORTRAN 90 Interface

Generic: **ICASE** (**CH**)
Specific: The specific interface name is **ICASE**.

FORTRAN 77 Interface

Single: **ICASE** (**CH**)

Description

Routine **ICASE** converts a character to its integer ASCII value. The conversion is case insensitive; that is, it returns the ASCII value of the corresponding uppercase letter for a lowercase letter.

Example

This example shows the case insensitive conversion.

```
      USE ICASE_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      CHARACTER  CHR
!                                Get output unit number
```

```
      CALL UMACH (2, NOUT)
!                                Get ASCII value for the character
!                                'a'.
      CHR = 'a'
      WRITE (NOUT,99999) CHR, ICASE(CHR)
!
99999 FORMAT (' For the character  ', A1, '  the ICASE value is : ', &
             I3)
      END
```

Output

```
 For the character  a  the ICASE value is :  65
```

IICSR

This function compares two character strings using the ASCII collating sequence but without regard to case.

Function Return Value

IICSR — Comparison indicator. (Output)

Let USTR1 and USTR2 be the uppercase versions of STR1 and STR2, respectively. The following table indicates the relationship between USTR1 and USTR2 as determined by the ASCII collating sequence.

IICSR	Meaning
-1	USTR1 precedes USTR2
0	USTR1 equals USTR2
1	USTR1 follows USTR2

Required Arguments

STR1 — First character string. (Input)

STR2 — Second character string. (Input)

FORTRAN 90 Interface

Generic: IICSR (STR1, STR2)

Specific: The specific interface name is IICSR.

FORTRAN 77 Interface

Single: IICSR (STR1, STR2)

Description

Routine IICSR compares two character strings. It returns -1 if the first string is less than the second string, 0 if they are equal, and 1 if the first string is greater than the second string. The comparison is case insensitive.

Comments

If the two strings, STR1 and STR2, are of unequal length, the shorter string is considered as if it were extended with blanks to the length of the longer string.

Example

This example shows different cases on comparing two strings.

```
      USE IICSR_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      CHARACTER  STR1*6, STR2*6
!                                Get output unit number
      CALL UMACH (2, NOUT)
!                                Compare String1 and String2
!                                String1 is 'bigger' than String2
      STR1 = 'ABc 1'
      STR2 = ' '
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
!                                String1 is 'equal' to String2
      STR1 = 'AbC'
      STR2 = 'ABc'
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
!                                String1 is 'smaller' than String2
      STR1 = 'ABc'
      STR2 = 'aBC 1'
      WRITE (NOUT,99999) STR1, STR2, IICSR(STR1,STR2)
!
99999 FORMAT (' For String1 = ', A6, 'and String2 = ', A6, &
             ' IICSR = ', I2, /)
      END
```

Output

```
 For String1 = ABc 1 and String2 =        IICSR =  1

 For String1 = AbC   and String2 = ABc    IICSR =  0

 For String1 = ABc   and String2 = aBC 1  IICSR = -1
```

IIDEX

This function determines the position in a string at which a given character sequence begins without regard to case.

Function Return Value

IIDEX — Position in **CHRSTR** where **KEY** begins. (Output)
If **KEY** occurs more than once in **CHRSTR**, the starting position of the first occurrence is returned. If **KEY** does not occur in **CHRSTR**, then **IIDEX** returns a zero.

Required Arguments

CHRSTR — Character string to be searched. (Input)

KEY — Character string that contains the key sequence. (Input)

FORTRAN 90 Interface

Generic: **IIDEX** (**CHRSTR**, **KEY**)
Specific: The specific interface name is **S_IIDEX**.

FORTRAN 77 Interface

Single: **IIDEX** (**CHRSTR**, **KEY**)

Description

Routine **IIDEX** searches for a key string in a given string and returns the index of the starting element at which the key character string begins. It returns 0 if there is no match. The comparison is case insensitive. For a case-sensitive version, use the FORTRAN 77 intrinsic function **INDEX**.

Comments

If the length of **KEY** is greater than the length **CHRSTR**, **IIDEX** returns a zero.

Example

This example locates a key string.

```
      USE IIDEX_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      CHARACTER  KEY*5, STRING*10
!                                 Get output unit number
      CALL UMACH (2, NOUT)
!                                 Locate KEY in STRING
      STRING = 'a1b2c3d4e5'
      KEY    = 'C3d4E'
      WRITE (NOUT,99999) STRING, KEY, IIDEX(STRING,KEY)
!
      KEY = 'F'
      WRITE (NOUT,99999) STRING, KEY, IIDEX(STRING,KEY)
!
99999 FORMAT (' For STRING = ', A10, ' and KEY = ', A5, ' IIDEX = ', I2, &
             /)
      END
```

Output

```
 For STRING = a1b2c3d4e5 and KEY = C3d4E IIDEX =  5

 For STRING = a1b2c3d4e5 and KEY = F     IIDEX =  0
```

CVTSI

Converts a character string containing an integer number into the corresponding integer form.

Required Arguments

STRING — Character string containing an integer number. (Input)

NUMBER — The integer equivalent of `STRING`. (Output)

FORTRAN 90 Interface

Generic: `CALL CVTSI (STRING, NUMBER)`
Specific: The specific interface name is `CVTSI`.

FORTRAN 77 Interface

Single: `CALL CVTSI (STRING, NUMBER)`

Description

Routine `CVTSI` converts a character string containing an integer to an `INTEGER` variable. Leading and trailing blanks in the string are ignored. If the string contains something other than an integer, a terminal error is issued. If the string contains an integer larger than can be represented by an `INTEGER` variable as determined from routine `IMACH` (see the Reference Material), a terminal error is issued.

Example

The string "12345" is converted to an `INTEGER` variable.

```
      USE CVTSI_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    NOUT, NUMBER
      CHARACTER  STRING*10
!
      DATA STRING/'12345'/
!
      CALL CVTSI (STRING, NUMBER)
!
```

```
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'NUMBER = ', NUMBER
      END
```

Output

```
 NUMBER =   12345
```

CPSEC

This fuction returns CPU time used in seconds.

Function Return Value

CPSEC — CPU time used (in seconds) since first call to **CPSEC**. (Output)

Required Arguments

None

FORTRAN 90 Interface

Generic: **CPSEC** ()

Specific: The specific interface name is **CPSEC**.

FORTRAN 77 Interface

Single: **CPSEC** ()

Comments

1. The first call to **CPSEC** returns 0.0.
2. The accuracy of this routine depends on the hardware and the operating system. On some systems, identical runs can produce timings differing by more than 10 percent.

TIMDY

Gets time of day.

Required Arguments

IHOUR — Hour of the day. (Output)
`IHOUR` is between 0 and 23 inclusive.

MINUTE — Minute within the hour. (Output)
`MINUTE` is between 0 and 59 inclusive.

ISEC — Second within the minute. (Output)
`ISEC` is between 0 and 59 inclusive.

FORTRAN 90 Interface

Generic: `CALL TIMDY (IHOUR, MINUTE, ISEC)`
Specific: The specific interface name is `TIMDY`.

FORTRAN 77 Interface

Single: `CALL TIMDY (IHOUR, MINUTE, ISEC)`

Description

Routine `TIMDY` is used to retrieve the time of day.

Example

The following example uses `TIMDY` to return the current time. Obviously, the output is dependent upon the time at which the program is run.

```
      USE TIMDY_INT
      USE UMACH_INT

      IMPLICIT   NONE
      INTEGER    IHOUR, IMIN, ISEC, NOUT
!
      CALL TIMDY (IHOUR, IMIN, ISEC)
```

```
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Hour:Minute:Second = ', IHOUR, ':', IMIN, &
                     ':', ISEC
      IF (IHOUR .EQ. 0) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past midnight.'
      ELSE IF (IHOUR .LT. 12) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past ', IHOUR, ' am.'
      ELSE IF (IHOUR .EQ. 12) THEN
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past noon.'
      ELSE
         WRITE (NOUT,*) 'The time is ', IMIN, ' minute(s), ', ISEC, &
                        ' second(s) past ', IHOUR-12, ' pm.'
      END IF
      END
```

Output

```
 Hour:Minute:Second =   16:  52:  29
 The time is   52 minute(s),   29 second(s) past   4 pm.
```

TDATE

Gets today's date.

Required Arguments

IDAY — Day of the month. (Output)
IDAY is between 1 and 31 inclusive.

MONTH — Month of the year. (Output)
MONTH is between 1 and 12 inclusive.

IYEAR — Year. (Output)
For example, IYEAR = 1985.

FORTRAN 90 Interface

Generic: CALL TDATE (IDAY, MONTH, IYEAR)
Specific: The specific interface name is TDATE.

FORTRAN 77 Interface

Single: CALL TDATE (IDAY, MONTH, IYEAR)

Description

Routine TDATE is used to retrieve today's date. Obviously, the output is dependent upon the date the program is run.

Example

The following example uses TDATE to return today's date.

```
          USE TDATE_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    IDAY, IYEAR, MONTH, NOUT
    !
          CALL TDATE (IDAY, MONTH, IYEAR)
```

```
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'Day-Month-Year = ', IDAY, '-', MONTH, &
                     '-', IYEAR
      END
```

Output

```

 Day-Month-Year =   2-  4-  1991
```

NDAYS

This function computes the number of days from January 1, 1900, to the given date.

Function Return Value

NDAYS — Function value. (Output)
If NDAYS is negative, it indicates the number of days prior to January 1, 1900.

Required Arguments

IDAY — Day of the input date. (Input)

MONTH — Month of the input date. (Input)

IYEAR — Year of the input date. (Input)
1950 would correspond to the year 1950 A.D. and 50 would correspond to year 50 A.D.

FORTRAN 90 Interface

Generic: NDAYS (IDAY, MONTH, IYEAR)
Specific: The specific interface name is NDAYS.

FORTRAN 77 Interface

Single: NDAYS (IDAY, MONTH, IYEAR)

Description

Function NDAYS returns the number of days from January 1, 1900, to the given date. The function NDAYS returns negative values for days prior to January 1, 1900. A negative IYEAR can be used to specify B.C. Input dates in year 0 and for October 5, 1582, through October 14, 1582, inclusive, do not exist; consequently, in these cases, NDAYS issues a terminal error.

Comments

1. Informational error

Type	Code	Description
1	1	The Julian calendar, the first modern calendar, went into use in 45 B.C. No calendar prior to 45 B.C. was as universally used nor as accurate as the Julian. Therefore, it is assumed that the Julian calendar was in use prior to 45 B.C.

2. The number of days from one date to a second date can be computed by two references to NDAYS and then calculating the difference.

3. The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. NDAYS makes the proper adjustment for the change in calendars.

Example

The following example uses NDAYS to compute the number of days from January 15, 1986, to February 28, 1986:

```
          USE NDAYS_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    IDAY, IYEAR, MONTH, NDAY0, NDAY1, NOUT
    !
          IDAY  = 15
          MONTH = 1
          IYEAR = 1986
          NDAY0 = NDAYS(IDAY,MONTH,IYEAR)
          IDAY  = 28
          MONTH = 2
          IYEAR = 1986
          NDAY1 = NDAYS(IDAY,MONTH,IYEAR)
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) 'Number of days = ', NDAY1 - NDAY0
          END
```

Output

```
    Number of days =   44
```

NDYIN

Gives the date corresponding to the number of days since January 1, 1900.

Required Arguments

NDAYS — Number of days since January 1, 1900. (Input)

IDAY — Day of the input date. (Output)

MONTH — Month of the input date. (Output)

IYEAR — Year of the input date. (Output)
1950 would correspond to the year 195 A.D. and `-`50 would correspond to year 50 B.C.

FORTRAN 90 Interface

Generic: `CALL NDYIN (NDAYS, IDAY, MONTH, IYEAR)`
Specific: The specific interface name is `NDYIN`.

FORTRAN 77 Interface

Single: `CALL NDYIN (NDAYS, IDAY, MONTH, IYEAR)`

Description

Routine `NDYIN` computes the date corresponding to the number of days since January 1, 1900. For an input value of `NDAYS` that is negative, the date computed is prior to January 1, 1900. The routine `NDYIN` is the inverse of `NDAYS`.

Comments

The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. Routine `NDYIN` makes the proper adjustment for the change in calendars.

Example

The following example uses **NDYIN** to compute the date for the 100th day of 1986. This is accomplished by first using NDAYS to get the "day number" for December 31, 1985.

```
          USE NDYIN_INT
          USE NDAYS_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    IDAY, IYEAR, MONTH, NDAYO, NOUT, NDAY0
    !
          NDAY0 = NDAYS(31,12,1985)
          CALL NDYIN (NDAY0+100, IDAY, MONTH, IYEAR)
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) 'Day 100 of 1986 is (day-month-year) ', IDAY, &
                         '-', MONTH, '-', IYEAR
          END
```

Output

```
    Day 100 of 1986 is (day-month-year)   10-  4-  1986
```

IDYWK

This function computes the day of the week for a given date.

Function Return Value

IDYWK — Function value. (Output)
The value of **IDYWK** ranges from 1 to 7, where 1 corresponds to Sunday and 7 corresponds to Saturday.

Required Arguments

IDAY — Day of the input date. (Input)

MONTH — Month of the input date. (Input)

IYEAR — Year of the input date. (Input)
1950 would correspond to the year 1950 A.D. and 50 would correspond to year 50 A.D.

FORTRAN 90 Interface

Generic: **IDYWK** (**IDAY**, **MONTH**, **IYEAR**)
Specific: The specific interface name is **IDYWK**.

FORTRAN 77 Interface

Single: **IDYWK** (**IDAY**, **MONTH**, **IYEAR**)

Description

Function **IDYWK** returns an integer code that specifies the day of week for a given date. Sunday corresponds to 1, Monday corresponds to 2, and so forth.

A negative **IYEAR** can be used to specify B.C. Input dates in year 0 and for October 5, 1582, through October 14, 1582, inclusive, do not exist; consequently, in these cases, **IDYWK** issues a terminal error.

Comments

1. Informational error

Type	Code	Description
1	1	The Julian calendar, the first modern calendar, went into use in 45 B.C. No calendar prior to 45 B.C. was as universally used nor as accurate as the Julian. Therefore, it is assumed that the Julian calendar was in use prior to 45 B.C.

2. The beginning of the Gregorian calendar was the first day after October 4, 1582, which became October 15, 1582. Prior to that, the Julian calendar was in use. Function **IDYWK** makes the proper adjustment for the change in calendars.

Example

The following example uses **IDYWK** to return the day of the week for February 24, 1963.

```
          USE IDYWK_INT
          USE UMACH_INT

          IMPLICIT   NONE
          INTEGER    IDAY, IYEAR, MONTH, NOUT
    !
          IDAY  = 24
          MONTH = 2
          IYEAR = 1963
          CALL UMACH (2, NOUT)
          WRITE (NOUT,*) 'IDYWK (index for day of week) = ', &
                         IDYWK(IDAY,MONTH,IYEAR)
          END
```

Output

```
    IDYWK (index for day of week) =   1
```

VERSL

This function obtains STAT/LIBRARY-related version and system information.

Function Return Value

VERSL — CHARACTER string containing information. (Output)

Required Arguments

ISELCT — Option for the information to retrieve. (Input)

ISELCT	VERSL
1	IMSL STAT/LIBRARY version number
2	Operating system (and version number) for which the library was produced.
3	Fortran compiler (and version number) for which the library was produced.

FORTRAN 90 Interface

Generic: VERSL (ISELCT)

Specific: The specific interface name is S_VERSL.

FORTRAN 77 Interface

Single: VERSL (ISELCT)

Example

In this example, we print all of the information returned by VERSL on a particular machine. The output is omitted because the results are system dependent.

```
          USE UMACH_INT
          USE VERSL_INT

          IMPLICIT   NONE
          INTEGER    ISELCT, NOUT
          CHARACTER  STRING(3)*50, TEMP*32
    !
          STRING(1) = '('' IMSL STAT/LIBRARY Version Number:  '', A)'
```

```
      STRING(2) = '('' Operating System ID Number:  '', A)'
      STRING(3) = '('' Fortran Compiler Version Number:  '', A)'
!                                Print the versions and numbers.
      CALL UMACH (2, NOUT)
      DO 10  ISELCT=1, 3
         TEMP = VERSL(ISELCT)
         WRITE (NOUT,STRING(ISELCT)) TEMP
   10 CONTINUE
      END
```

GDATA

Retrieves a commonly analyzed data set.

Required Arguments

IDATA — Data set indicator. (Input)

IDATA	NOBS	NVAR	Description of Data Set
1	16	7	Longley
2	176	2	Wolfer sunspot
3	150	5	Fisher iris
4	144	1	Box and Jenkins Series G
5	13	5	Draper and Smith Appendix B
6	197	1	Box and Jenkins Series A
7	296	2	Box and Jenkins Series J
8	100	4	Robinson Multichannel Time Series
9	113	34	Afifi and Azen Data Set A

Set `IDATA` = 0 to print a description of all the data sets above. In this case, the remaining arguments are not referenced.

X — `NOBS` by `NVAR` matrix containing the data set. (Output)

NOBS — Number of observations or rows in the output matrix. (Output)

NVAR — Number of variables or columns in the output matrix. (Output)

Optional Arguments

IPRINT — Printing option. (Input)
Default: `IPRINT` = 0.

IPRINT	Action
0	No printing is performed.
1	Rows 1 through 10 of X are printed.
2	All rows of X are printed.

When printing is performed, a header listing the data set name and a reference is printed.

LDX — Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

NDX — Second dimension of the matrix **X** exactly as specified in the dimension statement of the calling program. (Input)
Default: **NDX** = size (**X**,2).

FORTRAN 90 Interface

Generic: CALL GDATA (IDATA, X, NOBS, NVAR, [, ...])
Specific: The specific interface names are S_GDATA and D_GDATA.

FORTRAN 77 Interface

Single: CALL GDATA (IDATA, IPRINT, NOBS, NVAR, X, LDX, NDX)
Double: The double precision name is DGDATA.

Description

Routine **GDATA** retrieves a standard data set frequently cited in statistics textbooks or in this manual. The following table gives the references for each data set:

IDATA	Reference
1	Longley (1967)
2	Anderson (1971, page 660)
3	Fisher (1936); Mardia, Kent, and Bibby (1979, Table 1.2.2)
4	Box and Jenkins (1976, page 531)
5	Draper and Smith (1981, pages 629-630)
6	Box and Jenkins (1976, page 525)
7	Box and Jenkins (1976, page 532-533)
8	Robinson (1967, page 204)
9	Afifi and Azen (1979, pages 16-22)

Example

GDATA is used to copy the Longley data set into the matrix X.

```
      USE GDATA_INT

      IMPLICIT   NONE
      INTEGER    LDX, NDX
      PARAMETER  (LDX=200, NDX=10)
!
      INTEGER    IDATA, IPRINT, NOBS, NVAR
      REAL       X(LDX,NDX)
!
      IDATA  = 1
      IPRINT = 2
      CALL GDATA (IDATA, X, NOBS, NVAR, IPRINT=IPRINT)
!
      END
```

Output

```
 The Longley data.
 Longley, James W. (1967), An appraisal of least squares programs for the electronic
 computer from the point of view of the user, Journal of the American Statistical
 Association, 62, 819-841.
 This data set consists of 16 observations on 7 variables.
                                  X
           1          2          3          4          5          6
 1      83.0   234289.0     2356.0     1590.0   107608.0     1947.0
 2      88.5   259426.0     2325.0     1456.0   108632.0     1948.0
 3      88.2   258054.0     3682.0     1616.0   109773.0     1949.0
 4      89.5   284599.0     3351.0     1650.0   110929.0     1950.0
 5      96.2   328975.0     2099.0     3099.0   112075.0     1951.0
 6      98.1   346999.0     1932.0     3594.0   113270.0     1952.0
```

7	99.0	365385.0	1870.0	3547.0	115094.0	1953.0
8	100.0	363112.0	3578.0	3350.0	116219.0	1954.0
9	101.2	397469.0	2904.0	3048.0	117388.0	1955.0
10	104.6	419180.0	2822.0	2857.0	118734.0	1956.0
11	108.4	442769.0	2936.0	2798.0	120445.0	1957.0
12	110.8	444546.0	4681.0	2637.0	121950.0	1958.0
13	112.6	482704.0	3813.0	2552.0	123366.0	1959.0
14	114.2	502601.0	3931.0	2514.0	125368.0	1960.0
15	115.7	518173.0	4806.0	2572.0	127852.0	1961.0
16	116.9	554894.0	4007.0	2827.0	130081.0	1962.0

	7
1	60323.0
2	61122.0
3	60171.0
4	61187.0
5	63221.0
6	63639.0
7	64989.0
8	63761.0
9	66019.0
10	67857.0
11	68169.0
12	66513.0
13	68655.0
14	69564.0
15	69331.0
16	70551.0

Mathematical Support

Routines

GIRTS

Solves a triangular (possibly singular) set of linear systems and/or compute a generalized inverse of an upper triangular matrix.

Required Arguments

R - N by N upper triangular matrix. (Input)
If R contains a zero along the diagonal, the remaining elements of the row must also be zero. Only the upper triangle of R is referenced.

B - N by NB matrix containing the right hand sides of the linear system. (Input, if NB > 0)
If NB = 0, B is not referenced and can be a vector length one.

IRANK - Rank of R. (Output)

X - N by NB matrix containing the solution matrix corresponding to the right hand side B. (Output, if NB > 0)
If B is not needed, then X and B can share the same storage locations. If NB = 0, x is not referenced and can be a vector of length one.

Optional Arguments

N - Order of the upper triangular matrix R. (Input)
Default: N = size (R,2).

LDR - Leading dimension of R exactly as specified in the dimension statement of the calling program. (Input)
Default: LDR = size (R,1).

NB - Number of columns in B. (Input)
NB must be nonnegative. If NB is zero, no linear systems are solved.
Default: NB = size (B,2).

LDB - Leading dimension of B exactly as specified in the dimension statement of the calling program. (Input)
Default: LDB = size (B,1).

IPATH - Path option. (Input)
Default: IPATH = 1.

IPATH	Action
1	Solve R * X = B.
2	Solve R^T * X = B.
3	Solve R * X = B and compute RINV.
4	Solve R^T * X = B and compute RINV.

LDX - Leading dimension of X exactly as specified in the dimension statement of the calling program. (Input)
Default: LDX = size (X,1).

RINV - N by N upper triangular matrix that is the inverse of R when R is nonsingular. (Output, if IPATH equals 3 or 4)
(When R is singular, RINV is g_3 inverse. See the Algorithm section for an explanation of g_3 inverses.) If IPATH = 1 or 2, RINV is not referenced and can be a 1x1 array. If IPATH = 3 or 4 and R is not needed, then R and RINV can share the same storage locations.

LDRINV - Leading dimension of RINV exactly as specified in the dimension statement of the calling program. (Input)

FORTRAN 90 Interface

Generic: CALL GIRTS (R, B, IRANK, X [, ...])
Specific: The specific interface names are S_GIRTS and D_GIRTS.

FORTRAN 77 Interface

Single: CALL GIRTS (N, R, LDR, NB, B, LDB, IPATH, IRANK, X, LDX, RINV, LDRINV)
Double: The double precision name is DGIRTS.

Description

Routine GIRTS solves systems of linear algebraic equations with a triangular coefficient matrix. Inversion of the coefficient matrix is an option. The coefficient matrix can contain a zero diagonal element, but if so, the remaining elements in the row must be zero also. (A terminal error message is issued if a nonzero element appears in the row of the coefficient matrix where a zero diagonal element appears.)

If solution of a linear system is requested (i.e., NB > 0) and row *i* of the coefficient matrix contains elements all equal to zero, the following action is taken:

The *i*-th row of the solution X is set to zero.

If **IPATH** is 1 or 3, a warning error is issued when the *i*-th row of the right-hand side **B** is not zero.

If **IPATH** is 2 or 4, a warning error is issued when the *i*-th row of the reduced right-hand side (obtained after the first *i* – 1 variables are eliminated from row *i*) is not zero within a computed tolerance.

If an inverse of the coefficient matrix is requested and row *i* contains elements all equal to zero, row *i* and column *i* elements of **RINV** are set to zero. The resulting inverse is a g_3 inverse of *R*. For a matrix *G* to be g_3 inverse of a matrix *A*, *G* must satisfy Conditions 1, 2, and 3 for the Moore-Penrose inverse but generally fail Condition 4. The four conditions for *G* to be a Moore-Penrose inverse of *A* are as follows:

1. *AGA* = *A*
2. *GAG* = *G*
3. *AG* is symmetric
4. *GA* is symmetric

For a detailed description of the algorithm, see Section 2 in Sallas and Lionti (1988).

Comments

1. Informational error

Type	Code	Description
3	1	The linear system of equations is inconsistent.

2. Routine **GIRTS** assumes that a singular **R** is represented by zero rows in **R**. No other forms of singularity in **R** are allowed.

Example

The following example is taken from Maindonald (1984, pp. 102-105). A linear system *Rx* = *B* is solved, and a g_3 inverse of *R* is computed.

```
          USE GIRTS_INT
          USE WRRRN_INT

          IMPLICIT   NONE
          INTEGER    LDB, LDR, LDRINV, LDX, N, NB, J
          PARAMETER  (N=4, NB=1, LDB=N, LDR=N, LDRINV=N, LDX=N)
     !
          INTEGER    IPATH, IRANK
          REAL       B(LDB,NB), R(LDR,N), RINV(LDRINV,N), X(LDX,NB)
     !
```

```
      DATA (R(1,J),J=1,N)/6.0, 2.0, 5.0, 1.0/, B(1,1)/3.0/
      DATA (R(2,J),J=1,N)/0.0, 4.0,-2.0, 2.0/, B(2,1)/4.0/
      DATA (R(3,J),J=1,N)/0.0, 0.0, 0.0, 0.0/, B(3,1)/0.0/
      DATA (R(4,J),J=1,N)/0.0, 0.0, 0.0, 3.0/, B(4,1)/3.0/
!
      IPATH = 3
      CALL GIRTS (R, B, IRANK, X, IPATH=IPATH, RINV=RINV)
!
      CALL WRRRN ('RINV', RINV)
      CALL WRRRN ('X', X)
      END
```

Output

```
                RINV
            1       2       3       4
1   0.1667  -0.0833   0.0000   0.0000
2   0.0000   0.2500   0.0000  -0.1667
3   0.0000   0.0000   0.0000   0.0000
4   0.0000   0.0000   0.0000   0.3333

    X
1   0.167
2   0.500
3   0.000
4   1.000
```

CHFAC

Computes an upper triangular factorization of a real symmetric nonnegative definite matrix.

Required Arguments

A - N by N symmetric nonnegative definite matrix for which an upper triangular factorization is desired. (Input)
Only elements in the upper triangle of A are referenced.

IRANK - Rank of A. (Output)
N - IRANK is the number of effective zero pivots.

R - N by N upper triangular matrix containing the R matrix from a Cholesky decomposition R^TR of A. (Output)
The elements of the appropriate rows of R are set to 0.0 if linear dependence of the columns of A is declared. (There are N - IRANK rows of R whose elements are set to 0.0.) If A is not needed, then R and A can share the same storage locations.

Optional Arguments

N - Order of the matrix. (Input)
Default: N = size (A,2).

LDA - Leading dimension of A exactly as specified in the dimension statement in the calling program. (Input)
Default: LDA = size (A,1).

TOL - Tolerance used in determining linear dependence. (Input)
TOL = 100 * AMACH(4) is a common choice. See documentation for routine AMACH.
Default: TOL = 1.19e –5 for single precision and 2.2d –14 for double precision.

LDR - Leading dimension of R exactly as specified in the dimension statement in the calling program. (Input)
Default: LDR = size (R,1).

FORTRAN 90 Interface

Generic: CALL CHFAC (A, IRANK, R [, ...])

Specific: The specific interface names are **S_CHFAC** and **D_CHFAC**.

FORTRAN 77 Interface

Single: **CALL CHFAC** (**N**, **A**, **LDA**, **TOL**, **IRANK**, **R**, **LDR**)

Double: The double precision name is **DCHFAC**.

Description

Routine **CHFAC** computes a Cholesky factorization $R^T R = A$ of an $n \times n$ symmetric nonnegative definite matrix A. The matrix R is taken to be an upper triangular matrix. The diagonal elements of R are taken to be nonnegative. If A is singular and has rank r, $n - r$ rows of R have all their elements taken to be zero.

The algorithm is based on the work of Healy (1968). The algorithm proceeds sequentially by columns. The i-th column is declared to be linearly dependent on the first $i - 1$ columns if

$$\left| a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 \right| \leq \varepsilon |a_{ii}|$$

where **ε** (stored in **TOL**) is the input tolerance. When a linear dependence is declared, all elements in the i-th row of R are set to zero.

Modifications due to Farebrother and Berry (1974) and Barrett and Healy (1978) for checking for matrices that are not nonnegative definite are also incorporated. Routine **CHFAC** declares A to not be nonnegative definite and issues an error message with an error code of 1 if either of the
following conditions is satisfied:

1. $a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 < -\varepsilon |a_{ii}|$
2. $r_{ii} = 0 \quad \text{and} \quad |a_{ik} - \sum_{j=1}^{i-1} r_{ji} r_{jk}| > \varepsilon \sqrt{a_{ii} a_{kk}}, \quad k > i$

Healy's (1968) algorithm and **CHFAC** permit the matrices A and R to occupy the same storage locations. Barrett and Healy (1978) in their remark neglect this fact. Routine **CHFAC** uses

$$\sum_{j=1}^{i-1} r_{ji}^2$$

for a_{ii} in the Condition 2 above to remedy this problem.

Comments

1. Informational error

Type	Code	Description
3	1	The input matrix is not nonnegative definite within the tolerance defined by TOL.

2. Elements of row *i* of *R* are set to 0.0 if a linear dependence is declared. Linear dependence is declared if

$$\left| a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2 \right| \leq TOL * \left| a_{ii} \right|$$

Example

A Cholesky factorization of a 5 × 5 symmetric nonnegative definite matrix is computed. Maindonald (1984, pages 85-86) discusses in detail the computations for this problem.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE IMSL_LIBRARIES

      IMPLICIT   NONE
      INTEGER    LDA, LDR, N, J
      PARAMETER  (N=5, LDA=N, LDR=N)
!
      INTEGER    IRANK, NOUT
      REAL       A(LDA,N), R(LDR,N), TOL
!
      DATA (A(1,J),J=1,N)/36.0, 12.0, 30.0,  6.0, 18.0/
      DATA (A(2,J),J=1,N)/12.0, 20.0,  2.0, 10.0, 22.0/
      DATA (A(3,J),J=1,N)/30.0,  2.0, 29.0,  1.0,  7.0/
      DATA (A(4,J),J=1,N)/ 6.0, 10.0,  1.0, 14.0, 20.0/
      DATA (A(5,J),J=1,N)/ 8.0, 22.0,  7.0, 20.0, 40.0/
!
      CALL CHFAC (A, IRANK, R)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,*) 'IRANK = ', IRANK
      CALL WRRRN ('R', R)
      END
```

Output

```
 IRANK =   4

                     R
         1       2       3       4       5
1    6.000   2.000   5.000   1.000   3.000
```

```
     2   0.000   4.000  -2.000   2.000   4.000
     3   0.000   0.000   0.000   0.000   0.000
     4   0.000   0.000   0.000   3.000   3.000
     5   0.000   0.000   0.000   0.000   2.449
```

MCHOL

Computes an upper triangular factorization of a real symmetric matrix *A* plus a diagonal matrix *D*, where *D* is determined sequentially during the Cholesky factorization in order to make *A* + *D* nonnegative definite.

Required Arguments

A - **N** by **N** symmetric matrix for which a Cholesky factorization is attempted. (Input)
Only elements in the upper triangle and diagonal of **A** are referenced.

IRANK - Rank of *A* + *D*. (Output)

R - **N** by **N** upper triangular matrix containing the *R* matrix from a Cholesky decomposition R^TR of *A* + *D*. (Output)
The lower triangle of **R** is not referenced. If **A** is not needed, then **R** and **A** can share the same storage locations.

DMAX - Largest diagonal element of *D*. (Output)
If **DMAX** equals 0.0, then *A* is nonnegative definite, and *R* is a Cholesky factorization of *A*. If **DMAX** is positive, then *A* is indefinite, i.e., *A* has at least one negative eigenvalue. In this case, **DMAX** is an upper bound on the absolute value of the minimum eigenvalue.

IND - Index for subsequent computation of a descent direction in the case of a saddle point. (Output)
If **IND** = 0, then *A* is nonnegative definite. For positive **IND**, let *e* be a unit vector with **IND**-th element 1 and remaining elements 0. The solution *s* of *Rs* = *e* is a direction of negative curvature, i.e., $s^T As$ is negative.

Optional Arguments

N - Order of the matrix. (Input)
Default: **N** = size (**A**,2).

LDA - Leading dimension of **A** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDA** = size (**A**,1).

TOL - Tolerance used in determining linear dependence. (Input)
TOL = 100 * **AMACH**(4) is a common choice. See documentation for routine **AMACH**.
Default: **TOL** = 1.e-5 for single precision and 2.d –14 for double precision.

LDR - Leading dimension of **R** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDR** = size (**R**,1).

FORTRAN 90 Interface

Generic: **CALL MCHOL** (**A**, **IRANK**, **R**, **DMAX**, **IND** [, ...])
Specific: The specific interface names are **S_MCHOL** and **D_MCHOL**.

FORTRAN 77 Interface

Single: **CALL MCHOL** (**N**, **A**, **LDA**, **TOL**, **IRANK**, **R**, **LDR**, **DMAX**, **IND**)
Double: The double precision name is **DMCHOL**.

Description

Routine **MCHOL** computes a Cholesky factorization, R^TR, of $A + D$ where A is symmetric and D is a diagonal matrix with sufficiently large diagonal elements such that $A + D$ is nonnegative definite. The routine is similar to one described by Gill, Murray, and Wright (1981, pages 108-111). Here, though, we allow $A + D$ to be singular.

The algorithm proceeds sequentially by rows. If $A + D$ is singular, the Cholesky factor R is taken to have some rows that are entirely zero. The i-th row of $A + D$ is declared to be linearly dependent on the first $i - 1$ rows if the following two conditions are satisfied:

$$1. \quad |a_{ii} - \sum_{j=1}^{i-1} r_{ji}^2| \leq \varepsilon |a_{ii}|$$
$$2. \quad |a_{ik} - \sum_{j=1}^{i-1} r_{ji} r_{jk}| \leq \varepsilon \sqrt{|a_{ii} a_{kk}|}, \quad k > i$$

where **ε** is the input argument **TOL**.

The routine **MCHOL** is often used to find a descent direction in a minimization problem. Let A and g be the current Hessian and gradient, respectively, associated with the minimization problem. The solution s of $As = -g$ may not give a descent direction if A is not nonnegative definite. Instead, in order to guarantee a descent direction, a solution s of $(A + D)s = -g$ can be found where $A + D$ is nonnegative definite. Routine **MCHOL** is useful for computing the upper triangular Cholesky factor R of $A + D$ so that routine **GIRTS** can be invoked to compute the descent direction s by solving successively the two triangular linear systems $R^Tx = -g$ and $Rs = x$ for x and then s. Also if $g = 0$ and A is not nonnegative definite, i.e., the current solution is a saddle point, **GIRTS** can be used to compute a descent direction s from the linear system $Rs = e$ where e is a unit vector with

$$\varepsilon_i = \begin{cases} 1 & \text{if } i = \text{IND} \\ 0 & \text{otherwise} \end{cases}$$

Examples

Example 1

A Cholesky factorization of a 5 × 5 symmetric nonnegative definite matrix is computed. Maindonald (1984, pages 85-86) discusses the example.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE MCHOL_INT
      USE UMACH_INT
      USE WRRRN_INT

      IMPLICIT   NONE

      INTEGER    LDA, LDR, N, J
      PARAMETER  (N=5, LDA=N, LDR=N)
!
      INTEGER    IND, IRANK, NOUT
      REAL       A(LDA,N), DMAX, R(LDR,N), TOL
!
      DATA (A(1,J),J=1,N)/36.0, 12.0, 30.0, 6.0, 18.0/
      DATA (A(2,J),J=1,N)/12.0, 20.0, 2.0, 10.0, 22.0/
      DATA (A(3,J),J=1,N)/30.0, 2.0, 29.0, 1.0, 7.0/
      DATA (A(4,J),J=1,N)/6.0, 10.0, 1.0, 14.0, 20.0/
      DATA (A(5,J),J=1,N)/8.0, 22.0, 7.0, 20.0, 40.0/
!
      TOL = 0.00001
      CALL MCHOL (A, IRANK, R, DMAX, IND, TOL=TOL)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998) ' IRANK = ', IRANK
      WRITE (NOUT,99999) ' DMAX =  ', DMAX
      WRITE (NOUT,99998) ' IND =   ', IND
99998 FORMAT (A, I3)
99999 FORMAT (A, 1PE10.3)
      CALL WRRRN ('R', R)
      END
```

Output

```
 IRANK =   4
 DMAX =   0.000E+00
 IND =     0

                    R
        1       2       3       4       5
1   6.000   2.000   5.000   1.000   3.000
2   0.000   4.000  -2.000   2.000   4.000
3   0.000   0.000   0.000   0.000   0.000
4   0.000   0.000   0.000   3.000   3.000
5   0.000   0.000   0.000   0.000   2.449
```

Example 2

A modified Cholesky factorization of a 3×3 symmetric indefinite matrix A is computed. A solution of $Rs = e_3$ is also obtained using routine **GIRTS**. Note that $s^T As$ is negative as verified by using routine **BLINF** (IMSL MATH/LIBRARY). Gill, Murray, and Wright (1981, page 111) discuss the example.

```
!                                  SPECIFICATIONS FOR PARAMETERS
      USE IMSL_LIBRARIES

      IMPLICIT   NONE

      INTEGER    LDA, LDR, N, J
      PARAMETER  (N=3, LDA=N, LDR=N)
!
      INTEGER    IND, IRANK, NOUT
      REAL       A(LDA,N), DMAX, E(N,1), R(LDR,N), S(N, 1), SPAS, TOL
!
      DATA (A(1,J),J=1,N)/1, 1, 2/
      DATA (A(2,J),J=1,N)/1, 1, 3/
      DATA (A(3,J),J=1,N)/2, 3, 1/
!
      TOL = 0.00001
      CALL MCHOL (A, IRANK, R, DMAX, IND, TOL=TOL)
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99998) ' IRANK = ', IRANK
      WRITE (NOUT,99999) ' DMAX  = ', DMAX
      WRITE (NOUT,99998) ' IND   = ', IND
      CALL WRRRN ('R', R)
      IF (IND .GT. 0) THEN
         E= 0.0
         E(IND, 1) = 1.0
         CALL GIRTS (R, E, IRANK, S)
         SPAS = BLINF(A, S(:,1), S(:,1))
         WRITE (NOUT,*) ' '
         WRITE (NOUT,99999) ' trans(s)*A*s = ', SPAS
      END IF
99998 FORMAT (A, I3)
99999 FORMAT (A, F10.3)
      END
```

Output

```
IRANK =   3
DMAX  =      5.016
IND   =   3

            R
          1       2       3
1   1.942   0.515   1.030
2   0.000   2.398   1.030
3   0.000   0.000   1.059

trans(s)*A*s =     -2.254
```

ENOS

This function evaluates the expected value of a normal order statistic.

Function Return Value

ENOS - Function value, the expected value of the `I`-th order statistic in a sample of size `N` from the standard normal distribution. (Output)
See Comment 1.

Required Arguments

I - Rank of the order statistic. (Input)

N - Sample size. (Input)

FORTRAN 90 Interface

Generic: `ENOS` (`I`, `N`)
Specific: The specific interface names are `S_ENOS` and `D_ENOS`.

FORTRAN 77 Interface

Single: `ENOS` (`I`, `N`)
Double: The double precision name is `DENOS`.

Description

Let $X_1 \le X_2 \le \ldots \le X_n$ be the order statistics of a random sample of size n from a standard normal distribution. The expected value of X_i is given by

$$\frac{n!}{(n-i)!(i-1)!}\int_{-\infty}^{\infty} x\left[\Phi(x)\right]^{i-1}\left[1-\Phi(x)\right]^{n-i}\phi(x)\,dx$$

where $\phi(x)$ and $\Phi(x)$ are the standard normal density and cumulative distribution functions respectively (David 1981).

Function **ENOS** evaluates the integral using a trapezoidal rule after first making a logarithmic transformation. This is the method used by Harter (1961). Although the method permits computations for any value of *n*, extremely large values of *n* cannot be guaranteed to be as accurate as smaller values of *n*. For *n* > 2500, the method is inappropriate.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = ENOS(I, N)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(ENOS(I, N))
   ```

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Informational errors

Type	Code	Description
3	1	The rank of the order statistic is less than 1. A rank of 1 is assumed.
3	2	The rank of the order statistic is greater than sample size (N). A rank of N is assumed.

Example

In this example, we compute the expected value of the first order statistic in a sample of size 5 from a standard normal distribution.

```
      USE UMACH_INT
      USE ENOS_INT

      IMPLICIT   NONE
      INTEGER    I, N, NOUT
      REAL       EX
!
      CALL UMACH (2, NOUT)
      I  = 1
      N  = 5
      EX = ENOS(I,N)
      WRITE (NOUT,99999) EX
99999 FORMAT (' The expected value of the smallest order statistic', &
             /, ' in a normal sample of size 5 is ', F9.5)
      END
```

Output

```
    The expected value of the smallest order statistic
    in a normal sample of size 5 is  -1.16296
```

AMILLR

This functon evaluates Mill's ratio (the ratio of the ordinate to the upper tail area of the standardized normal distribution).

Function Return Value

AMILLR - Function value, Mill's ratio. (Output)

Required Arguments

X - Value at which Mill's ratio is evaluated. (Input)
In order to avoid overflow, **X** must be less than a bound that is machine dependent. On most machines, the bound is greater than –13. The function underflows (and is set to 0.0) for small values of **X**. On most machines, the underflow does not occur unless **X** is less than -13.

FORTRAN 90 Interface

Generic: **AMILLR** (**X**)
Specific: The specific interface names are **S_AMILLR** and **D_AMILLR**.

FORTRAN 77 Interface

Single: **AMILLR** (**X**)
Double: The double precision name is **DMILLR**.

Description

Function **AMILLR** evaluates Mill's ratio, the *hazard rate* for the standard normal distribution. It is computed as the ratio of the ordinate to the upper tail area of the standard normal distribution, that is, $\phi(x)/(1 - \Phi(x))$, where $\phi(x)$ and $\Phi(x)$ are the standard normal density and cumulative distribution functions, respectively. The reciprocal of Mill's ratio is called the *failure rate* in reliability and life testing applications. As x becomes small, the ratio goes to zero. For large x (how large is machine dependent), the ratio cannot be computed. Function **AMILLR** computes $1 - \Phi(x)$ using the complementary error function (IMSL 1991) rather than as one minus the normal distribution function, which would underflow sooner as x gets small.

Comments

Informational Error

Type	Code	Description
2	1	The function underflows because X is too small.

Example

In this example, we compute Mill's ratio at $x = -1.0$.

```
      USE UMACH_INT
      USE AMILLR_INT

      IMPLICIT   NONE
      INTEGER    NOUT
      REAL       R, X
!
      CALL UMACH (2, NOUT)
      X = -1.0
      R = AMILLR(X)
      WRITE (NOUT,99999) R
99999 FORMAT (' Mill''s ratio at -1.0 is ', F8.5)
      END
```

Output

```
Mill's ratio at -1.0 is  0.28760
```

QUADT

Forms a *k-d* tree.

Required Arguments

X - NROW by NCOL matrix containing the data to be used on this call. (Input/Output)
On output the rows of X have been rearranged to form the *k-d* tree. X must not contain missing values (NaN).

IND - Vector of length NVAR containing the column numbers in X to be used in the forming the *k-d* tree. (Input)

NBUCK - Bucket size. (Input)
NBUCK gives the maximum number of observations in a leaf of the *k-d* tree. NBUCK = 3 is a common choice. NBUCK should be small when compared to NROW.

IDISCR - Vector of length NROW containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree. (Output)
IDISCR(I) = 0 if the observation is a terminal node. IND(IDISCR(I)) is the column number in X to be used as the discriminator.

PART - Vector of length NROW containing the value to be used in the partition for this observation. (Output)

Optional Arguments

NROW - Number of rows of X to be used in forming the *k-d* tree. (Input)
Default: NROW = size (X,1).

NVAR - Number of variables to be used in forming the tree. (Input)
Default: NVAR = size (IND,1).

NCOL - Number of columns in X. (Input)
Default: NCOL = size (X,2).

LDX - Leading dimension of X exactly as specified in the dimension statement in the calling program. (Input)
Default: LDX = size (X,1).

FORTRAN 90 Interface

Generic: **CALL QUADT** (**X**, **IND**, **NBUCK**, **IDISCR**, **PART** [, ...])
Specific: The specific interface names are **S_QUADT** and **D_QUADT**.

FORTRAN 77 Interface

Single: **CALL QUADT** (**NROW**, **NVAR**, **NCOL**, **X**, **LDX**, **IND**, **NBUCK**, **IDISCR**, **PART**)
Double: The double precision name is **DQUADT**.

Description

Routine **QUADT** creates the data structure required for a *k-d* tree. A *k-d* tree is a multivariate form of B-tree that is especially useful for finding nearest neighbors but may be of use in other situations. Once the *k-d* tree has been formed, routine **NGHBR** may be used to find the nearest neighbors for any point in logarithmic time.

The basic algorithm is given by Friedman, Bentley, and Finkel (1977) and can be summarized as follows:

1. Let *l* = 1 and *h* = **NROW**.
2. Let *k* = (*l* + *h*)/2.
3. Each column in **X** to be used in forming the *k-d* tree is examined over the range [*l*, *h*] in order to find the column with the maximum spread. Let *j* equal this column number.
4. The *k*-th element of **PART** is set to the median value in the range [*l*, *h*] of the *j*-th column in **X** while **IDISCR**(**k**) is set to the element in **IND** that points to this column.
5. The rows of **X** are interchanged so that all rows of **X** with values in column *j* less than or equal to the median value computed in Step 4 occur before (or at) the *k*-th element.
6. Go to Step 2 repeatedly with zero, one, or two submatrices in **X**. Go to Step 2 with the submatrix formed from rows *l* to *k* of **X** if *k* **-** *l* is greater than **NBUCK**. Go to Step 2 with the submatrix formed from rows *k* + 1 to *h* of **X** if *h* **-** *k* **-** 1 is greater than **NBUCK**.

The bucket size, **NBUCK**, is the maximum number of observations allowed in the lowest level of the *k-d* tree, i.e., in the leaves of the tree. The choice of **NBUCK** can affect the speed with which nearest neighbors are found. A value of 3 or 5 is a common choice, but if the number of nearest neighbors to be obtained is large, a larger value for **NBUCK** should probably be used.

Comments

Workspace may be explicitly provided, if desired, by use of **Q2ADT**/**DQ2ADT**. The reference is:

CALL Q2ADT (NROW, NVAR, NCOL, X, LDX, IND, NBUCK, IDISCR, PART, ILOW, IHIGH, WK, IWK)

The additional arguments are as follows:

ILOW - Work vector of length $\log_2$ (NROW) + 3.

IHIGH - Work vector of length $\log_2$ (NROW) + 3.

WK - Work vector of length NROW.

IWK - Work vector of length NROW.

Example

The following example creates a *k-d* tree from financial data collected for firms approximately 2 years prior to bankruptcy and for financially sound firms at about the same point in time. The data on five variables, X_1 = (population), X_2 = (cash flow)/(total debt), X_3 = (net income)/(total assets), X_4 = (current assets)/(current liabilities), and X_5 = (current assets)/(net sales) are taken from Johnson and Wichern (1988, page 536).

```
          USE QUADT_INT
          USE WRRRN_INT
          USE WRIRN_INT

          IMPLICIT   NONE
          INTEGER    LDX, NBUCK, NCOL, NROW, NVAR, I
          PARAMETER  (LDX=47, NBUCK=3, NCOL=5, NROW=47, NVAR=4)
     !
          INTEGER    IDISCR(NROW), IND(NVAR)
          REAL       PART(NROW), X(LDX,NCOL)
     !
          DATA IND/2, 3, 4, 5/
          DATA (X(I,1),I=1,47)/1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., &
              1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2., 2., &
              2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., &
              2., 2., 2., 2., 2., 2./
          DATA (X(I,2),I=1,47)/-0.4485, -0.5633, 0.0643, -0.0721, -0.1002, &
              -0.1421, 0.0351, -0.0653, 0.0724, -0.1353, -0.2298, 0.0713, &
              0.0109, -0.2777, 0.1454, 0.3703, -0.0757, 0.0451, 0.0115, &
              0.1227, -0.2843, 0.5135, 0.0769, 0.3776, 0.1933, 0.3248, &
              0.3132, 0.1184, -0.0173, 0.2169, 0.1703, 0.1460, -0.0985, &
              0.1398, 0.1379, 0.1486, 0.1633, 0.2907, 0.5383, -0.3330, &
              0.4785, 0.5603, 0.2029, 0.2029, 0.4746, 0.1661, 0.5808/
          DATA (X(I,3),I=1,47)/-0.4106, -0.3114, -0.3114, -0.0930, &
              -0.0917, -0.0651, 0.0147, -0.0566, -0.0076, -0.1433, &
              -0.2961, 0.0205, 0.0011, -0.2316, 0.0500, 0.1098, -0.0821, &
              0.0263, -0.0032, 0.1055, -0.2703, 0.1001, 0.0195, 0.1075, &
              0.0473, 0.0718, 0.0511, 0.0499, 0.0233, 0.0779, 0.0695, &
              0.0518, -0.0123, -0.0312, 0.0728, 0.0564, 0.0486, 0.0597, &
              0.1064, -0.0854, 0.0910, 0.1112, 0.0792, 0.0792, 0.1380, &
              0.0351, 0.0371/
          DATA (X(I,4),I=1,47)/1.0865, 1.5134, 1.0077, 1.4544, 1.5644, &
              0.7066, 1.5046, 1.3737, 1.3723, 1.4196, 0.3310, 1.3124, &
              2.1495, 1.1918, 1.8762, 1.9941, 1.5077, 1.6756, 1.2602, &
              1.1434, 1.2722, 2.4871, 2.0069, 3.2651, 2.2506, 4.2401, &
              4.4500, 2.5210, 2.0538, 2.3489, 1.7973, 2.1692, 2.5029, &
```

```
            0.4611, 2.6123, 2.2347, 2.3080, 1.8381, 2.3293, 3.0124, &
            1.2444, 4.2918, 1.9936, 1.9936, 2.9166, 2.4527, 5.0594/
      DATA (X(I,5),I=1,47)/0.4526, 0.1642, 0.3978, 0.2589, 0.6683, &
            0.2794, 0.7080, 0.4032, 0.3361, 0.4347, 0.1824, 0.2497, &
            0.6969, 0.6601, 0.2723, 0.3828, 0.4215, 0.9494, 0.6038, &
            0.1655, 0.5128, 0.5368, 0.5304, 0.3548, 0.3309, 0.6279, &
            0.6852, 0.6925, 0.3483, 0.3970, 0.5174, 0.5500, 0.5778, &
            0.2643, 0.5151, 0.5563, 0.1978, 0.3786, 0.4835, 0.4730, &
            0.1847, 0.4443, 0.3018, 0.3018, 0.4487, 0.1370, 0.1268/
!
      CALL QUADT (X, IND, NBUCK, IDISCR, PART)
      CALL WRRRN ('first 10 rows of X after QUADT', X, 10, NCOL, LDX)
      CALL WRRRN ('PART', PART, 1, NROW, 1)
      CALL WRIRN ('IDISCR', IDISCR, 1, NROW, 1)
!
      END
```

Output

```
      first 10 rows of X after QUADT
          1       2       3       4       5
 1    1.000  -0.230  -0.296   0.331   0.182
 2    2.000   0.140  -0.031   0.461   0.264
 3    1.000  -0.142  -0.065   0.707   0.279
 4    1.000  -0.449  -0.411   1.087   0.453
 5    1.000   0.064  -0.311   1.008   0.398
 6    1.000   0.123   0.105   1.143   0.166
 7    1.000  -0.284  -0.270   1.272   0.513
 8    1.000  -0.278  -0.232   1.192   0.660
 9    1.000   0.012  -0.003   1.260   0.604
10    1.000   0.071   0.021   1.312   0.250

                                  PART
    1      2      3      4      5      6      7      8      9     10
0.000  0.461  0.857  0.000  0.064  1.168  0.000 -0.278  0.041  0.000

   11     12     13     14     15     16     17     18     19     20
0.072  1.373  0.000 -0.072  0.412  0.000  0.435 -0.015  0.000  1.876

   21     22     23     24     25     26     27     28     29     30
0.448  0.000   .708  1.994  0.000  0.203  2.152  0.000  2.308  0.390

   31     32     33     34     35     36     37     38     39     40
0.000   .550  0.147  0.000  0.217  2.453  0.000  2.521  0.128  0.000

   41     42     43     44     45     46     47
2.612  3.012  0.000  4.240  4.292  4.755  0.000

                                IDISCR
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 0  3  3  0  1  3  0  1  1  0  1  3  0  1  4  0  4  1  0  3

21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 4  0  4  3  0  1  3  0  3  4  0  4  1  0  1  3  0  3  1  0

41 42 43 44 45 46 47
 3  3  0  3  3  3  0
```

NGHBR

Search's a *k-d* tree for the *k* nearest neighbors of a key.

Required Arguments

XKEY - Vector of length NVAR containing the key for which nearest neighbors are desired. (Input)
Note that the elements in XKEY are not arranged in the same manner as the columns in X.

K - Number of nearest neighbors to find. (Input)

X - NROW by NCOL matrix containing the data used to form the *k-d* tree as output from routine QUADT. (Input)
X must not contain missing values (NaN).

IND - Vector of length NVAR containing the column numbers in X used in forming the *k-d* tree. (Input)

NBUCK - Bucket size. (Input)
NBUCK is the maximum number of observations in a leaf of the *k-d* tree. The value of NBUCK should be the same as the value used in forming the *k-d* tree (i.e. as input to the routine QUADT).

IDISCR - Vector of length NROW containing the element number in IND that points to the column of X to be used as the discriminator in the *k-d* tree, as output from routine QUADT. (Input)
IDISCR(I) = 0 if the observation is a terminal node. IND(IDISCR(I)) is the column number in X to be used as the discriminator.

PART - Vector of length NROW containing the median value to be used for the partition, as output from routine QUADT. (Input)

IPQR - Vector of length K containing the indices of the nearest neighbors. (Output)

PQD - Vector of length K containing the nearest neighbor distances. (Output)

Optional Arguments

NVAR - Number of variables used to form the *k-d* tree. (Input)
Default: NVAR = size (XKEY,1).

NROW - Number of rows of X used to form the *k-d* tree. (Input)
Default: NROW = size (X,1).

NCOL - Number of columns in **X**. (Input)
Default: **NCOL** = size (**X**,2).

LDX - Leading dimension of **X** exactly as specified in the dimension statement in the calling program. (Input)
Default: **LDX** = size (**X**,1).

METRIC - Metric to use in computing the *k* nearest neighbors. (Input)
Default: **METRIC** = 0.

METRIC	Metric used
0	Euclidean distance
1	L_1 norm
2	L_∞ norm

FORTRAN 90 Interface

Generic: **CALL NGHBR** (**XKEY**, **K**, **X**, **IND**, **NBUCK**, **IDISCR**, **PART**, **IPQR**, **PQD** [, ...])
Specific: The specific interface names are **S_NGHBR** and **D_NGHBR**.

FORTRAN 77 Interface

Single: **CALL NGHBR** (**NVAR**, **XKEY**, **K**, **NROW**, **NCOL**, **X**, **LDX**, **IND**, **NBUCK**, **IDISCR**, **PART**, **METRIC**, **IPQR**)
Double: The double precision name is **DNGHBR**.

Description

Routine **NGHBR** finds the *k* nearest neighbors in an input *k-d* tree for an arbitrary key, **XKEY** in logarithmic time. A *k-d* tree is a form of B-tree that is especially useful for finding nearest neighbors. The *k-d* tree input into routine **NGHBR** should be produced by routine **QUADT**. Three metrics, Euclidean, L_1, and L_∞, are available for defining the nearest neighbors. The user should note that if the input key is a row of the *k-d* tree, then the row will be returned as one of the nearest neighbors. In this case, only *k* - 1 nearest neighbors will be found.

The algorithm is given by Friedman, Bentley, and Finkel (1977) and is summarized in the following. The basic idea is to traverse the *k-d* tree in order to determine which leaves of the tree need to be examined for the nearest neighbor. The algorithm is efficient because most leaves are not examined.

1. Let *l* = 1 and *h* = **NROW**.
2. Let *k* = (*l* + *h*)/2, and *j* and *p* be the *k*-th elements of **IDISCR** and **PART**, respectively.

3. If (*h* - *l*) is less than **NBUCK**, then go to Step 4. Otherwise, let *m* be the *j*-th element of **IND**. If the (*k*, *m*)-th element of **X** is greater than *p*, then let *l* = *k* + 1 and go to Step 2. Otherwise, set *h* = *k* and go to Step 2.

4. Examine each row in **X** from row *l* to row *h* to determine if it is a nearest neighbor. Check to see if rows in **X** (leaves of the tree) adjacent to these rows need to be examined (see Friedman, Bentley, and Finkel (1977)). If necessary, examine the adjacent rows for nearest neighbors.

The value used for the bucket size, **NBUCK**, must be the same value as was used in routine **QUADT** when the *k-d* tree was created. A common choice for **NBUCK** is three.

Comments

1. Workspace may be explicitly provided, if desired, by use of **N2HBR/DN2HBR**. The reference is:

   ```
   CALL N2HBR (NVAR, XKEY, K, NROW, NCOL, X, LDX, IND, IDISCR, PART, METRIC, IPQR,
       PQD, ILOW, IHIGH, ISIDE, BNDL, BNDH)
   ```

   The additional arguments are as follows:

   ILOW - Work vector of length $\log_2$ (**NROW**) + 3.

   IHIGH - Work vector of length $\log_2$ (**NROW**) + 3.

   ISIDE - Work vector of length $\log_2$ (**NROW**) + 3.

   BNDL - Work vector of length **NVAR** * ($\log_2$ (**NROW**) + 3).

   BNDH - Work vector of length **NVAR** * ($\log_2$ (**NROW**) + 3).

2. Informational error

Type	Code	Description
4	1	The data structure input is not a *k-d* tree. Use routine QUADT to create the *k-d* tree.

Example

The following example creates a *k-d* tree from financial data collected for firms approximately 2 years prior to bankruptcy and for financially sound firms at about the same point in time. The data on five variables, X_1 = (population), X_2 = (cash flow)/(total dept), X_3 = (net income)/(total assets), X_4 = (current assets)/(current liabilities), and X_5 = (current assets)/(net sales) are taken from Johnson and Wichern, page 536. Routine **NGHBR** is then used to determine the 5 nearest neighbors of the first row in **X**. As expected, one of the nearest neighbors found is the key (the first row in **X**).

```
      USE QUADT_INT
      USE NGHBR_INT
```

```
      USE WRIRN_INT
      USE WRRRN_INT

      IMPLICIT   NONE
      INTEGER    K, LDX, METRIC, NBUCK, NCOL, NROW, NVAR
      PARAMETER  (K=5, LDX=47, METRIC=1, NBUCK=3, NCOL=5, NROW=47, &
                 NVAR=4)
!
      INTEGER    I, IDISCR(NROW), IND(NVAR), IPQR(K)
      REAL       PART(NROW), PQD(K), X(LDX,NCOL), XKEY(NVAR)
!
      DATA IND/2, 3, 4, 5/
      DATA (X(I,1),I=1,47)/1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., &
          1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., 2., &
          2., 2., 2., 2., 2., 2./
      DATA (X(I,2),I=1,47)/-0.4485, -0.5633, 0.0643, -0.0721, -0.1002, &
          -0.1421, 0.0351, -0.0653, 0.0724, -0.1353, -0.2298, 0.0713, &
          0.0109, -0.2777, 0.1454, 0.3703, -0.0757, 0.0451, 0.0115, &
          0.1227, -0.2843, 0.5135, 0.0769, 0.3776, 0.1933, 0.3248, &
          0.3132, 0.1184, -0.0173, 0.2169, 0.1703, 0.1460, -0.0985, &
          0.1398, 0.1379, 0.1486, 0.1633, 0.2907, 0.5383, -0.3330, &
          0.4785, 0.5603, 0.2029, 0.2029, 0.4746, 0.1661, 0.5808/
      DATA (X(I,3),I=1,47)/-0.4106, -0.3114, -0.3114, -0.0930, &
          -0.0917, -0.0651, 0.0147, -0.0566, -0.0076, -0.1433, &
          -0.2961, 0.0205, 0.0011, -0.2316, 0.0500, 0.1098, -0.0821, &
          0.0263, -0.0032, 0.1055, -0.2703, 0.1001, 0.0195, 0.1075, &
          0.0473, 0.0718, 0.0511, 0.0499, 0.0233, 0.0779, 0.0695, &
          0.0518, -0.0123, -0.0312, 0.0728, 0.0564, 0.0486, 0.0597, &
          0.1064, -0.0854, 0.0910, 0.1112, 0.0792, 0.0792, 0.1380, &
          0.0351, 0.0371/
      DATA (X(I,4),I=1,47)/1.0865, 1.5134, 1.0077, 1.4544, 1.5644, &
          0.7066, 1.5046, 1.3737, 1.3723, 1.4196, 0.3310, 1.3124, &
          2.1495, 1.1918, 1.8762, 1.9941, 1.5077, 1.6756, 1.2602, &
          1.1434, 1.2722, 2.4871, 2.0069, 3.2651, 2.2506, 4.2401, &
          4.4500, 2.5210, 2.0538, 2.3489, 1.7973, 2.1692, 2.5029, &
          0.4611, 2.6123, 2.2347, 2.3080, 1.8381, 2.3293, 3.0124, &
          1.2444, 4.2918, 1.9936, 1.9936, 2.9166, 2.4527, 5.0594/
      DATA (X(I,5),I=1,47)/0.4526, 0.1642, 0.3978, 0.2589, 0.6683,&
          0.2794, 0.7080, 0.4032, 0.3361, 0.4347, 0.1824, 0.2497, &
          0.6969, 0.6601, 0.2723, 0.3828, 0.4215, 0.9494, 0.6038, &
          0.1655, 0.5128, 0.5368, 0.5304, 0.3548, 0.3309, 0.6279, &
          0.6852, 0.6925, 0.3483, 0.3970, 0.5174, 0.5500, 0.5778, &
          0.2643, 0.5151, 0.5563, 0.1978, 0.3786, 0.4835, 0.4730, &
          0.1847, 0.4443, 0.3018, 0.3018, 0.4487, 0.1370, 0.1268/
!
!                                Create the k-d tree
!
      CALL QUADT (X, IND, NBUCK, IDISCR, PART)
!
      DO 10  I=1, NVAR
         XKEY(I) = X(1,IND(I))
   10 CONTINUE
!
      CALL NGHBR (XKEY, K, X, IND, NBUCK, IDISCR, PART, IPQR, PQD, &
                  METRIC=METRIC)
!
      CALL WRIRN ('Indices of the nearest neighbors, IPQR.', IPQR, 1, K, 1)
      CALL WRRRN ('Nearest neighbor distances, PQD.', PQD, 1, K, 1)
!
```

```
          END
```

Output

```

    Indices of the nearest neighbors, IPQR.
              1   2   3   4   5
              1   3   2   5   7

      Nearest neighbor distances, PQD.
        1       2       3       4       5
    0.000   0.791   0.847   1.201   1.352
```

Reference Material

Contents

User Errors

IMSL routines attempt to detect user errors and handle them in a way that provides as much information to the user as possible. To do this, we recognize various levels of severity of errors, and we also consider the extent of the error in the context of the purpose of the routine; a trivial error in one situation may be serious in another. IMSL routines attempt to report as many errors as they can reasonably detect. Multiple errors present a difficult problem in error detection because input is interpreted in an uncertain context after the first error is detected.

What Determines Error Severity

In some cases, the user's input may be mathematically correct, but because of limitations of the computer arithmetic and of the algorithm used, it is not possible to compute an answer accurately. In this case, the assessed degree of accuracy determines the severity of the error. In cases where the routine computes several output quantities, if some are not computable but most are, an error condition exists. The severity depends on an assessment of the overall impact of the error.

Terminal errors

If the user's input is regarded as meaningless, such as **N** = **-** 1 when "**N**" is the number of equations, the routine prints a message giving the value of the erroneous input argument(s) and the reason for the erroneous input. The routine will then cause the user's program to stop. An error in which the user's input is meaningless is the most severe error and is called a *terminal error*. Multiple terminal error messages may be printed from a single routine.

Informational errors

In many cases, the best way to respond to an error condition is simply to correct the input and rerun the program. In other cases, the user may want to take actions in the program itself based on errors that occur. An error that may be used as the basis for corrective action within the program is called an *informational error*. If an informational error occurs, a user-retrievable code is set. A routine can return at most one informational error for a single reference to the routine. The codes for the informational error codes are printed in the error messages.

Other errors

In addition to informational errors, IMSL routines issue error messages for which no user-retrievable code is set. Multiple error messages for this kind of error may be printed. These errors, which generally are not described in the documentation, include terminal errors as well as less serious errors. Corrective action within the calling program is not possible for these errors.

Kinds of Errors and Default Actions

Five levels of severity of errors are defined in the STAT/LIBRARY. Each level has an associated PRINT attribute and a STOP attribute. These attributes have default settings (YES or NO), but they may also be set by the user. The purpose of having multiple error severity levels is to provide independent control of actions to be taken for errors of different severity. Upon return from an IMSL routine, exactly one error state exists. (A code 0 "error" is no informational error.) Even if more than one informational error occurs, only one message is printed (if the PRINT attribute is YES). Multiple errors for which no corrective action within the calling program is reasonable or necessary result in the printing of multiple messages (if the PRINT attribute for their severity level is YES). Errors of any of the severity levels except level 5 may be informational errors.

Level	Type
1	**Note**. A *note* is issued to indicate the possibility of a trivial error or simply to provide information about the computations. Default attributes: PRINT = NO, STOP = NO
2	**Alert**. An *alert* indicates that the user should be advised about events occurring in the software. Default attributes: PRINT = NO, STOP = NO
3	**Warning**. A *warning* indicates the existence of a condition that may require corrective action by the user or calling routine. A warning error may be issued because the results are accurate to only a few decimal places, because some of the output may be erroneous but most of the output is correct, or because some assumptions underlying the analysis technique are violated. Often no corrective action is necessary and the condition can be ignored. Default attributes: PRINT = YES, STOP = NO
4	**Fatal**. A *fatal* error indicates the existence of a condition that may be serious. In most cases, the user or calling routine must take corrective action to recover. Default attributes: PRINT = YES, STOP = YES
5	**Terminal**. A *terminal* error is serious. It usually is the result of an incorrect specification, such as specifying a negative number as the number of equations. These errors may also be caused by various programming errors impossible to diagnose correctly in FORTRAN. The resulting error message may be perplexing to the user. In such cases, the user is advised to compare carefully the actual arguments passed to the routine with the dummy argument descriptions given in the documentation. Special attention should be given to checking argument order and data types. A terminal error is not an informational error because corrective action within the program is generally not reasonable. In normal usage, execution is terminated immediately when a terminal error occurs. Messages relating to more than one terminal error are printed if they occur. Default attributes: PRINT = YES, STOP = YES

The user can set PRINT and STOP attributes by calling **ERSET** as described in "Routines for Error Handling."

Errors in Lower-Level Routines

It is possible that a user's program may call an IMSL routine that in turn calls a nested sequence of lower-level IMSL routines. If an error occurs at a lower level in such a nest of routines and if the lower-level routine cannot pass the information up to the original user- called routine, then a traceback of the routines is produced. The only common situation in which this can occur is when an IMSL routine calls a user-supplied routine that in turn calls another IMSL routine.

Routines for Error Handling

There are three ways in which the user may interact with the IMSL error handling system: (1) to change the default actions, (2) to retrieve the integer code of an informational error so as to take corrective action, and (3) to determine the severity level of an error. The routines to use are `ERSET`, `IERCD`, and `N1RTY`, respectively.

ERSET

Change the default printing or stopping actions when errors of a particular error severity level occur.

Required Arguments

IERSVR — Error severity level indicator. (Input)
If IERSVR = 0, actions are set for levels 1 to 5. If IERSVR is 1 to 5, actions are set for errors of the specified severity level.

IPACT — Printing action. (Input)

IPACT	Action
-1	Do not change current setting(s).
0	Do not print.
1	Print.
2	Restore the default setting(s).

ISACT — Stopping action. (Input)

ISACT	Action
-1	Do not change current setting(s).
0	Do not stop.
1	Stop.
2	Restore the default setting(s).

FORTRAN 90 Interface

Generic: CALL ERSET (IERSVR, IPACT, ISACT)
Specific: The specific interface name is ERSET.

FORTRAN 77 Interface

Single: CALL ERSET (IERSVR, IPACT, ISACT)

IERCD and N1RTY

The last two routines for interacting with the error handling system, **IERCD** and **N1RTY**, are **INTEGER** functions and are described in the following material.

IERCD retrieves the integer code for an informational error. Since it has no arguments, it may be used in the following way:

```
ICODE = IERCD( )
```

The function retrieves the code set by the most recently called IMSL routine.

N1RTY retrieves the error type set by the most recently called IMSL routine. It is used in the following way:

```
ITYPE = N1RTY(1)
```

ITYPE = 1, 2, 4, and 5 correspond to error severity levels 1, 2, 4, and 5, respectively. **ITYPE** = 3 and **ITYPE** = 6 are both warning errors, error severity level 3. While **ITYPE** = 3 errors are informational errors (**IERCD**( ) ≠ 0), **ITYPE** = 6 errors are not informational errors (**IERCD**( ) = 0).

For software developers requiring additional interaction with the IMSL error handling system, see Aird and Howell (1991).

Examples

Changes to default actions

Some possible changes to the default actions are illustrated below. The default actions remain in effect for the kinds of errors not included in the call to **ERSET**.

To turn off printing of warning error messages:

```
CALL ERSET (3, 0, -1)
```

To stop if warning errors occur:

```
CALL ERSET (3, -1, 1)
```

To print all error messages:

```
CALL ERSET (0, 1, -1)
```

To restore all default settings:

```
CALL ERSET (0, 2, 2)
```

Use of informational error to determine program action

In the program segment below, the Cholesky factorization of a matrix is to be performed. If it is determined that the matrix is not nonnegative definite (and often this is not immediately obvious), the program is to take a different branch.

```
                                        .
                                        .
                                        .
      CALL CHFAC (A, IRANK, R)
      IF (IERCD() .EQ. 1) THEN
!           Handle matrix that is not nonnegative definite
                                        .
                                        .
                                        .
      END IF
```

Examples of All Types of Errors

The program below illustrates each of the different types of errors detected by the STAT/LIBRARY routines. If the call to **ERSET** was not made, messages for errors of levels 1 and 2 would not be printed.

The error messages refer to the argument names that are used in the documentation for the routine, rather than the user's name of the variable used for the argument. In the messages generated by IMSL routine **CHFAC** in this example, references are made to **LDA** and **LDR**, whereas in the program literals were used for these arguments. Note that error codes are printed as part of the messages for informational errors.

```
      USE IMSL_LIBRARIES
!                                   Specifications for local variables
      INTEGER   IDO, IOPT, IRANK, N, NMISS, NOBS, NPOP, NROW, NUM
      REAL      A(2,2), CHSQ, CONPER, DF, PR, R(2,2), RCOEF, STAT(20), &
                SUMRY(11), TOL, X(10), XMEAN, Y(10)
!
      DATA X/-5.0, -4.0, -3.0, -2.0, -1.0, 1.0, 2.0, 3.0, 4.0, 5.0/
      DATA Y/3.0, 5.0, 4.0, 5.0, 6.0, 7.0, 6.0, 8.0, 7.0, 9.0/
      DATA A/2.0, 0.0, 0.0, -3.0/
!                                    Turn on printing and turn off
!                                    stopping for all error types.
      CALL ERSET (0, 1, 0)
!                                    Generate level 1 informational error.
      DF = 1000.0
      CHSQ = -1.0
      PR = CHIDF(CHSQ,DF)
!                                    Generate level 2 informational error.
      DF = 1000.0
      CHSQ = 10.0
      PR = CHIDF(CHSQ,DF)
!                                    Generate level 3 informational error.
      NUM = 11
      CALL LETTR (X, SUMRY, NMISS, NUM=NUM)
!                                    Generate level 4 informational error.
      N = 2
      TOL = 0.0001
      CALL CHFAC (A, IRANK, R, TOL=TOL)
!                                    Generate several level 5 errors.
```

```
      CALL CHFAC (A, IRANK, R, TOL=TOL, LDR = -2)
!                                     Generate several warning errors that
!                                     do not allow corrective action
!                                     (because no codes are listed for
!                                     these errors in the document for the
!                                     routine).
      NROW = 10
      NPOP = 100
      IOPT = 1
      CONPER = 0.95
      CALL SMPRR (NROW, X, Y, NPOP, XMEAN, STAT, IOPT=IOPT, CONPER=CONPER)
      END
```

Output

```
*** NOTE      ERROR 1 from CHIDF. Since CHSQ = -1.000000E+00 is less than
***           zero, the distribution function is zero at CHSQ.
*** ALERT     ERROR 3 from CHIDF. The normal distribution is used for large
***           degrees of freedom. However, it has produced underflow.
***           Therefore, the probability is set to 0.
*** WARNING   ERROR 3 from LETTR. NUM = 11 and the number of observations =
***           10. Since NUM is greater than the number of observations, it
***           is likely that the results are not useful.
*** WARNING   ERROR 1 from CHFAC. The leading 2 by 2 submatrix of the input
***           matrix is not nonnegative definite within the tolerance
***           definedby TOL = 1.000000E-04.
*** TERMINAL  ERROR 3 from CHFAC. N = 2 and LDA = 1. N must be less than or
***           equal to LDA.
*** TERMINAL  ERROR 5 from CHFAC. LDR = -2. LDR must be greater than or
***           equal to 1.
*** WARNING   ERROR 1 from SMPRR. CONPER = 9.500000E-01. The confidence
***           percentage is less than 50.0. Commonly used confidence
***           percentages are: 90.0, 95.0 or 99.0.
*** WARNING   ERROR 3 from SMPRR. The sample size, STAT(19) = 10. This is
***           less than 30. The confidence limits, which are computed using
***           a normal approximation, may not be very accurate.
*** WARNING   ERROR 7 from SMPRR. The coefficient of variation of one or
***           both of the variables exceeds 10%. The confidence limits,
***           which are computed using a normal approximation, may not be
***           very accurate.
```

Example of Traceback

The next program illustrates a situation in which a traceback is produced. Although the traceback shows an error code associated with a terminal error, this code has no meaning to the user; the printed message contains all relevant information and it is not assumed that the user would take corrective action based on knowledge of the code.

```

      USE IMSL_LIBRARIES
!                                 Specifications for local variables
      REAL       A, B, ERRABS, ERRREL, RESULT, ERREST
!                                 Specifications for common variables
      REAL       PIN, QIN, SAMP
      COMMON     PIN, QIN, SAMP
!                                 Specifications for functions
```

```
      EXTERNAL F
      REAL F
!                                Compute the expected value of the
!                                maximum order statistic in a sample
!                                of size SAMP from a beta distribution.
      A = 0.0
      B = 1.0
      ERRABS = 0.0
      ERRREL = 0.001
!                                Initialize parameters for the beta
!                                order statistic of interest.
      SAMP = 10.0
      PIN = 2.0
      QIN = -3.0
!                                The parameters for the beta must be
!                                nonnegative -- hence, the preceeding
!                                assignment causes an error.
      CALL QDAGS (F, A, B, RESULT, ERRABS=ERRABS, ERREST=ERREST)
!
      WRITE (*, *) RESULT, ERREST
      END
!
      REAL FUNCTION F (X)
      USE BETDF_INT
      REAL       X, PIN, QIN, SAMP
      COMMON     PIN, QIN, SAMP
!
      F = X*BETDF(X,PIN,QIN)**(SAMP-1.0)
      RETURN
      END
```

Output

```
 *** TERMINAL ERROR 4 from BETDF.   QIN = -3.000000E+00 must be greater than
 ***          0.0.
     Here is a traceback of subprogram calls in reverse order:
     Routine name                     Error type Error code
     ------------                     ---------- ----------
     BETDF                                 5          4
     Q2AGS                                 0          0 (Called internally)
     QDAGS                                 0          0
     USER                                  0          0
```

Machine-Dependent Constants

The function subprograms in this section return machine-dependent information and can be used to enhance portability of programs between different computers. The routines `IMACH`, and `AMACH` describe the computer's arithmetic. The routine `UMACH` describes the input, ouput, and error output unit numbers.

IMACH

This function retrieves machine integer constants that define the arithmetic used by the computer.

Function Return Value

IMACH(1) = Number of bits per integer storage unit.

IMACH(2) = Number of characters per integer storage unit:

Integers are represented in *M*-digit, base *A* form as

$$\sigma \sum_{k=0}^{M} x_k A^k$$

where **σ** is the sign and $0 \le x_k < A$, $k = 0, \ldots, M$.

Then,

IMACH(3) = *A*, the base.

IMACH(4) = *M*, the number of base-*A* digits.

IMACH(5) = $A^M - 1$, the largest integer.

The machine model assumes that floating-point numbers are represented in normalized *N*-digit base *B* form as

$$\sigma B^E \sum_{k=1}^{N} x_k B^{-k}$$

where **σ** is the sign, $0 < x_1 < B$, $0 \le x_k < B$, k = 2, …, *N* and $E\exists \le E \le E\forall$.

Then,

IMACH(6) = B, the base.

IMACH(7) = N_s, the number base-B-digits in single precision.

IMACH(8) = $E_{\min_s}$, the smallest single precision exponent.

IMACH(9) = $E_{\max_s}$, the largest single precision exponent.

IMACH(10) = N_d, the number base-*B*-digits in double precision.

IMACH(11) = $E_{\min_d}$, the smallest double precision exponent.

IMACH(12) = $E_{\max_d}$, largest double precision exponent.

Required Arguments

I — Index of the desired constant. (Input)

FORTRAN 90 Interface

Generic: **IMACH** (**I**)

Specific: The specific interface name is **IMACH**.

FORTRAN 77 Interface

Single: **IMACH** (**I**)

AMACH

The function subprogram **AMACH** retrieves machine constants that define the computer's single-precision or double precision arithmetic. Such floating-point numbers are represented in normalized *N*-digit, base *B* form as

$$\sigma B^{E} \sum_{k=1}^{N} x_k B^{-k}$$

where **σ** is the sign, $0 < x_1 < B$, $0 \le x_k < B$, $k = 2, \ldots, N$ and

$$E_{\min} \le E \le E_{\max}$$

Function Return Value

AMACH(1) = $B^{E_{\min}-1}$, the smallest normalized positive number.

AMACH(2) = $B^{E_{\max}}(1 - B^{-N})$, the largest number.

AMACH(3) = B^{-N}, the smallest relative spacing.

AMACH(4) = B^{1-N}, the largest relative spacing.

AMACH(5) = $\log_{10}(B)$.

AMACH(6) = NaN (*quiet* not a number).

AMACH(7) = positive machine infinity.

AMACH(8) = negative machine infinity.

See Comment 1 for a description of the use of the generic version of this function.

See Comment 2 for a description of `min`, `max`, and **N**.

Required Arguments

I — Index of the desired constant. (Input)

FORTRAN 90 Interface

Generic: **AMACH** (`I`)

Specific: The specific interface names are **S_AMACH** and **D_AMACH**.

FORTRAN 77 Interface

Single: **AMACH** (**I**)

Double: The double precision name is **DMACH**.

Comments

1. If the generic version of this function is used, the immediate result must be stored in a variable before use in an expression. For example:

   ```
   X = AMACH(I)
   Y = SQRT(X)
   ```

   must be used rather than

   ```
   Y = SQRT(AMACH(I))
   ```

   .

   If this is too much of a restriction on the programmer, then the specific name can be used without this restriction.

2. Note that for single precision B = **IMACH**(6), N = **IMACH**(7).
   E_{min} = **IMACH**(8), and E_{max} = **IMACH**(9).
   For double precision B = **IMACH**(6), N = **IMACH**(10).
   E_{min} = **IMACH**(11), and E_{max} = **IMACH**(12).

3. The IEEE standard for binary arithmetic (see IEEE 1985) specifies *quiet* NaN (not a number) as the result of various invalid or ambiguous operations, such as 0/0. The intent is that **AMACH**(6) return a quiet NaN. On computers that do not support a *quiet* NaN, a special value near **AMACH**(2) is returned for **AMACH**(6). On computers that do not have a special representation for infinity, **AMACH**(7) returns the same value as **AMACH**(2).

DMACH

See AMACH.

IFNAN(X)

This logical function checks if the argument **X** is NaN (not a number).

Function Return Value

IFNAN - Logical function value. True is returned if the input argument is a **NAN**. Otherwise, False is returned. (Output)

Required Arguments

X – Argument for which the test for **NAN** is desired. (Input)

FORTRAN 90 Interface

Generic: `IFNAN(X)`

Specific: The specific interface names are `S_IFNAN` and `D_IFNAN`.

FORTRAN 77 Interface

Single: `IFNAN (X)`

Double: The double precision name is `DIFNAN`.

Description

The logical function `IFNAN` checks if the single or double precision argument **X** is **NAN** (not a number). The function `IFNAN` is provided to facilitate the transfer of programs across computer systems. This is because the check for NaN can be tricky and not portable across computer systems that do not adhere to the IEEE standard. For example, on computers that support the IEEE standard for binary arithmetic (see IEEE 1985), NaN is specified as a bit format not equal to itself. Thus, the check is performed as

```
IFNAN = X .NE. X
```

On other computers that do not use IEEE floating-point format, the check can be performed as:

```
IFNAN = X .EQ. AMACH(6)
```

The function **IFNAN** is equivalent to the specification of the function Isnan listed in the Appendix, (IEEE 1985). The above example illustrates the use of **IFNAN**. If **X** is NaN, a message is printed instead of **X**. (Routine **UMACH**, which is described in the following section, is used to retrieve the output unit number for printing the message.)

Example

```
      USE IFNAN_INT
      USE AMACH_INT
      USE UMACH_INT
      INTEGER      NOUT
      REAL         X
!
      CALL UMACH (2, NOUT)
!
      X = AMACH(6)
      IF (IFNAN(X)) THEN
         WRITE (NOUT,*) ' X is NaN (not a number).'
      ELSE
         WRITE (NOUT,*) ' X = ', X
      END IF
!
      END
```

Output

```
 X is NaN (not a number).
```

UMACH

Routine **UMACH** sets or retrieves the input, output, or error output device unit numbers.

Required Arguments

N — Integer value indicating the action desired. If the value of **N** is negative, the input, output, or error output unit number is reset to **NUNIT**. If the value of **N** is positive, the input, output, or error output unit number is returned in **NUNIT**. See the table in argument **NUNIT** for legal values of **N**. (Input)

NUNIT — The unit number that is either retrieved or set, depending on the value of input argument **N**. (Input/Output)

The arguments are summarized by the following table:

N	Effect
1	Retrieves input unit number in NUNIT.
2	Retrieves output unit number in NUNIT.
3	Retrieves error output unit number in NUNIT.
–1	Sets the input unit number to NUNIT.
–2	Sets the output unit number to NUNIT.
–3	Sets the error output unit number to NUNIT.

FORTRAN 90 Interface

Generic: CALL UMACH (N, NUNIT)

Specific: The specific interface name is **UMACH**.

FORTRAN 77 Interface

Single: CALL UMACH (N, NUNIT)

Description

Routine **UMACH** sets or retrieves the input, output, or error output device unit numbers. **UMACH** is set automatically so that the default FORTRAN unit numbers for standard input, standard output, and standard error are used. These unit numbers can be changed by inserting a call to **UMACH** at the beginning of the main program that calls MATH/LIBRARY routines. If these unit numbers are changed from the standard values, the user should insert an appropriate **OPEN** statement in the calling program.

Example

In the following example, a terminal error is issued from the MATH/LIBRARY **AMACH** function since the argument is invalid. With a call to **UMACH**, the error message will be written to a local file named "**CHECKERR**".

```
      USE AMACH_INT
      USE UMACH_INT
      INTEGER     N, NUNIT
      REAL        X
!                                     Set Parameter
      N = 0
!
      NUNIT = 9
      CALL UMACH (-3, NUNIT)
      OPEN (UNIT=9,FILE='CHECKERR')
      X = AMACH(N)
      END
```

Output

```
The output from this example, written to "CHECKERR" is:
*** TERMINAL ERROR 5 from AMACH.  The argument must be between 1 and 8
***           inclusive. N = 0
```

Matrix Storage Modes

In this section, the word *matrix* will be used to refer to a mathematical object, and the word *array* will be used to refer to its representation as a FORTRAN data structure.

General Mode

A *general* matrix is an $N \times N$ matrix A. It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of **A**. It must be at least as large as **N**. IMSL general matrix subprograms only refer to values A_{ij} for $i = 1, \ldots, N$ and $j = 1, \ldots, N$. The data type of a general array can be one of **REAL**, **DOUBLE PRECISION**, or **COMPLEX**. If your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX** can also be declared.

Rectangular Mode

A *rectangular* matrix is an $M \times N$ matrix A. It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of **A**. It must be at least as large as **M**. IMSL rectangular matrix subprograms only refer to values A_{ij} for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. The data type of a rectangular array can be **REAL**, **DOUBLE PRECISION**, or **COMPLEX**. If your FORTRAN compiler allows, you can declare the nonstandard data type **DOUBLE COMPLEX**.

Symmetric Mode

A symmetric matrix is a square $N \times N$ matrix A, such that $A^T = A$. (A^T is the transpose of A.) It is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of **A**. It must be at least as large as **N**. IMSL symmetric matrix subprograms only refer to the upper or to the lower half of **A** (i.e., to values A_{ij} for $i = 1, \ldots, N$ and $j = i, \ldots, N$, or A_{ij} for $j = 1, \ldots, N$ and $i = j, \ldots, N$). The data type of a symmetric array can be one of **REAL** or **DOUBLE PRECISION**.

Use of the upper half of the array is denoted in the BLAS that compute with symmetric matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library using the **CHARACTER*1** flag **UPLO = 'U'**. Otherwise, **UPLO = 'L'** denotes that the lower half of the array is used.

Hermitian Mode

A *Hermitian* matrix is a square $N \times N$ matrix A, such that

$$\overline{A}^T = A$$

The matrix

$$\overline{A}$$

is the complex conjugate of A and

$$A^H \equiv \overline{A}^T$$

is the conjugate transpose of A. For Hermitian matrices, $A^H = A$. The matrix is stored in a FORTRAN array that is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of **A**. It must be at least as large as N. IMSL Hermitian matrix subprograms only refer to the upper or to the lower half of **A** (i.e., to values A_{ij} for $i = 1, \ldots, N$ and $j = i, \ldots, N$, or A_{ij} for $j = 1, \ldots, N$ and $i = j, \ldots, N$). Use of the upper half of the array is denoted in the BLAS that compute with Hermitian matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, using the **CHARACTER*1** flag **UPLO** = **'U'**. Otherwise, **UPLO** = **'L'** denotes that the lower half of the array is used. The data type of a Hermitian array can be **COMPLEX** or, if your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX**.

Triangular Mode

A *triangular* matrix is a square $N \times N$ matrix A such that values $A_{ij} = 0$ for $i < j$ or $A_{ij} = 0$ for $i > j$. The first condition defines a *lower* triangular matrix while the second condition defines an *upper* triangular matrix. A lower triangular matrix A is stored in the lower triangular part of a FORTRAN array **A**. An upper triangular matrix is stored in the upper triangular part of a FORTRAN array. Triangular matrices are called *unit* triangular whenever $A_{jj} = 1, j = 1, \ldots, N$. For unit triangular matrices, only the strictly lower or upper parts of the array are referenced. This is denoted in the BLAS that compute with triangular matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, using the **CHARACTER*1** flag **DIAG = 'U'**. Otherwise, **DIAG = 'N'** denotes that the diagonal array terms should be used. For unit triangular matrices, the diagonal terms are each used with the mathematical value

1. The array diagonal term does not need to be 1.0 in this usage. Use of the upper half of the array is denoted in the BLAS that compute with triangular matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, sing the **CHARACTER*1** flag **UPLO = 'U'**. Otherwise, **UPLO = 'L'** denotes that the lower half of the array is used. The data type of an array that contains a triangular matrix can be one of **REAL**, **DOUBLE PRECISION**, or **COMPLEX**. If your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX** can also be declared.

Band Storage Mode

A *band matrix* is an $M \times N$ matrix A with all of its nonzero elements "close" to the main diagonal. Specifically, values $A_{ij} = 0$ if $i - j >$ **NLCA** or $j - i >$ **NUCA**. The integers **NLCA** and **NUCA** are the *lower* and *upper* band widths. The integer m = **NLCA** + **NUCA** + 1 is the total band width. The diagonals, other than the main diagonal, are called *codiagonals*. While any $M \times N$ matrix is a band matrix, the band matrix mode is most useful only when the number of nonzero codiagonals is much less than m.

In the band storage mode, the **NLCA** lower codiagonals and **NUCA** upper codiagonals are stored in the rows of a FORTRAN array of dimension $m \times N$. The elements are stored in the same column of the array as they are in the matrix. The values A_{ij} inside the band width are stored in array positions ($i - j$ + **NUCA** + 1, j). This array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of A. It must be at least as large as m. The data type of a band matrix array can be one of **REAL**, **DOUBLE PRECISION**, **COMPLEX** or, if your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX**. Use of the **CHARACTER*1** flag **TRANS='N'** in the BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, specifies that the matrix A is used. The flag value

$$\text{TRANS} = \text{'}T\text{'} \text{ uses } A^T$$

while

$$\text{TRANS} = \text{'C'} \text{ uses } \overline{A}^T$$

For example, consider a real 5 × 5 band matrix with 1 lower and 2 upper codiagonals, stored in the FORTRAN array declared by the following statements:

```
      PARAMETER (N=5, NLCA=1, NUCA=2)
      REAL A(NLCA+NUCA+1, N)
```

The matrix A has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{21} & A_{22} & A_{23} & A_{24} & 0 \\ 0 & A_{32} & A_{33} & A_{34} & A_{35} \\ 0 & 0 & A_{43} & A_{44} & A_{45} \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix}$$

As a FORTRAN array, it is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{21} & A_{32} & A_{43} & A_{54} & \times \end{bmatrix}$$

The entries marked with an **x** in the above array are not referenced by the IMSL band subprograms.

Band Symmetric Storage Mode

A *band symmetric* matrix is a band matrix that is also symmetric. The band symmetric storage mode is similar to the band mode except only the lower or upper codiagonals are stored.

In the band symmetric storage mode, the **NCODA** upper codiagonals are stored in the rows of a FORTRAN array of dimension (**NCODA** + **1**) × *N*. The elements are stored in the same column of the array as they are in the matrix. Specifically, values A_{ij}, $j \le i$ inside the band are stored in array positions i **-** j + **NCODA** + 1, j). This is the storage mode designated by using the **CHARACTER*1** flag **UPLO = 'U'** in Level 2 BLAS that compute with band symmetric matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. Alternatively, A_{ij}, $j \le i$, inside the band, are stored in array positions (i **-** j + 1, j). This is the storage mode designated by using the **CHARACTER*1** flag **UPLO = 'L'** in these Level 2 BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. The array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of *A*. It must be at least as large as **NCODA** + 1. The data type of a band symmetric array can be **REAL** or **DOUBLE PRECISION**.

For example, consider a real 5 × 5 band matrix with 2 codiagonals. Its FORTRAN declaration is

```
      PARAMETER (N=5, NCODA=2)
      REAL A(NCODA+1, N)
```

The matrix *A* has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{12} & A_{22} & A_{23} & A_{24} & 0 \\ A_{13} & A_{23} & A_{33} & A_{34} & A_{35} \\ 0 & A_{24} & A_{34} & A_{44} & A_{45} \\ 0 & 0 & A_{35} & A_{45} & A_{55} \end{bmatrix}$$

Since A is symmetric, the values $A_{ij} = A_{ji}$. In the FORTRAN array, it is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

The entries marked with an × in the above array are not referenced by the IMSL band symmetric subprograms.

An alternate storage mode for band symmetric matrices is designated using the **CHARACTER*1** flag **UPLO = 'L'** in Level 2 BLAS that compute with band symmetric matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. In that case, the example matrix is represented as

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{12} & A_{23} & A_{34} & A_{45} & \times \\ A_{13} & A_{24} & A_{35} & \times & \times \end{bmatrix}$$

Band Hermitian Storage Mode

A *band Hermitian* matrix is a band matrix that is also Hermitian. The band Hermitian mode is a complex analogue of the band symmetric mode.

In the band Hermitian storage mode, the **NCODA** upper codiagonals are stored in the rows of a FORTRAN array of dimension (**NCODA** + 1) × *N*. The elements are stored in the same column of the array as they are in the matrix. In the Level 2 BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, this is denoted by using the **CHARACTER*1** flag **UPLO ='U'**. The array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of *A*. It must be at least as large as (**NCODA** + 1). The data type of a band Hermitian array can be **COMPLEX** or, if your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX**.

For example, consider a complex 5 × 5 band matrix with 2 codiagonals. Its FORTRAN declaration is

```
      PARAMETER (N=5, NCODA = 2)
```

```
      COMPLEX A(NCODA + 1, N)
```

The matrix *A* has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ \overline{A}_{12} & A_{22} & A_{23} & A_{24} & 0 \\ \overline{A}_{13} & \overline{A}_{23} & A_{33} & A_{34} & A_{35} \\ 0 & \overline{A}_{24} & \overline{A}_{34} & A_{44} & A_{45} \\ 0 & 0 & \overline{A}_{35} & \overline{A}_{45} & A_{55} \end{bmatrix}$$

where the value

$$\overline{A}_{ij}$$

is the complex conjugate of A_{ij}. This matrix represented as a FORTRAN array is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

The entries marked with an × in the above array are not referenced by the IMSL band Hermitian subprograms.

An alternate storage mode for band Hermitian matrices is designated using the **CHARACTER*1** flag **UPLO = 'L'** in Level 2 BLAS that compute with band Hermitian matrices, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. In that case, the example matrix is represented as

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ \overline{A}_{12} & \overline{A}_{23} & \overline{A}_{34} & \overline{A}_{45} & \times \\ \overline{A}_{13} & \overline{A}_{24} & \overline{A}_{35} & \times & \times \end{bmatrix}$$

Band Triangular Storage Mode

A *band triangular* matrix is a band matrix that is also triangular. In the band triangular storage mode, the **NCODA** codiagonals are stored in the rows of a FORTRAN array of dimension (**NCODA** + 1) × *N*. The elements are stored in the same column of the array as they are in the matrix. For usage in the Level 2 BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library, the **CHARACTER*1** flag **DIAG** has the same meaning as used in section "Triangular Storage Mode". The flag **UPLO** has the meaning analogous with its usage in the section "Banded Symmetric Storage Mode". This array is declared by the following statement:

```
DIMENSION A(LDA,N)
```

The parameter **LDA** is called the *leading dimension* of *A*. It must be at least as large as (**NCODA** + 1).

For example, consider a 5 × 5 band upper triangular matrix with 2 codiagonals. Its FORTRAN declaration is

```
      PARAMETER (N = 5, NCODA = 2)
      COMPLEX A(NCODA + 1, N)
```

The matrix *A* has the form

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ 0 & A_{22} & A_{23} & A_{24} & 0 \\ 0 & 0 & A_{33} & A_{34} & A_{35} \\ 0 & 0 & 0 & A_{44} & A_{45} \\ 0 & 0 & 0 & 0 & A_{55} \end{bmatrix}$$

This matrix represented as a FORTRAN array is

$$A = \begin{bmatrix} \times & \times & A_{13} & A_{24} & A_{35} \\ \times & A_{12} & A_{23} & A_{34} & A_{45} \\ A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \end{bmatrix}$$

This corresponds to the **CHARACTER*1** flags **DIAG = 'N'** and **UPLO = 'U'**. The matrix A^T is represented as the FORTRAN array

$$A = \begin{bmatrix} A_{11} & A_{22} & A_{33} & A_{44} & A_{55} \\ A_{12} & A_{23} & A_{34} & A_{45} & \times \\ A_{13} & A_{24} & A_{35} & \times & \times \end{bmatrix}$$

This corresponds to the **CHARACTER*1** flags **DIAG = 'N'** and **UPLO = 'L'**. In both examples, the entries indicated with an × are not referenced by IMSL subprograms.

Codiagonal Band Symmetric Storage Mode

This is an alternate storage mode for band symmetric matrices. It is not used by any of the BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. Storing data in a form transposed from the **Band Symmetric Storage Mode** maintains unit spacing between consecutive referenced array elements. This data structure is used to get good performance in the Cholesky decomposition algorithm that solves positive definite symmetric systems of linear equations *Ax* = *b*. The data type can be **REAL** or **DOUBLE PRECISION**. In the codiagonal band symmetric storage mode, the **NCODA** upper codiagonals and right-hand-side are stored in columns of this FORTRAN array. This array is declared by the following statement:

```
DIMENSION A(LDA, NCODA + 2)
```

The parameter **LDA** is the *leading positive dimension* of *A*. It must be at least as large as **N** + **NCODA**.

Consider a real symmetric 5 × 5 matrix with 2 codiagonals

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & 0 \\ A_{12} & A_{22} & A_{23} & A_{24} & 0 \\ A_{13} & A_{23} & A_{33} & A_{34} & A_{35} \\ 0 & A_{24} & A_{34} & A_{44} & A_{45} \\ 0 & 0 & A_{35} & A_{45} & A_{55} \end{bmatrix}$$

and a right-hand-side vector

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{bmatrix}$$

A FORTRAN declaration for the array to hold this matrix and right-hand-side vector is

```
      PARAMETER (N = 5, NCODA = 2, LDA = N + NCODA)
      REAL A(LDA, NCODA + 2)
```

The matrix and right-hand-side entries are placed in the FORTRAN array *A* as follows:

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ A_{11} & \times & \times & b_1 \\ A_{22} & A_{12} & \times & b_2 \\ A_{33} & A_{23} & A_{13} & b_3 \\ A_{44} & A_{34} & A_{24} & b_4 \\ A_{55} & A_{45} & A_{35} & b_5 \end{bmatrix}$$

Entries marked with an × do not need to be defined. Certain of the IMSL band symmetric subprograms will initialize and use these values during the solution process. When a solution is computed, the b_i, $i = 1, \ldots, 5$, are replaced by x_i, $i = 1, \ldots, 5$.

The nonzero A_{ij}, $j \geq i$, are stored in array locations *A*(*j* + **NCODA**, (*j* - *i*) + 1) . The right-hand-side entries b_j are stored in locations *A*(*j* + **NCODA**, **NCODA** + 2). The solution entries x_j are returned in *A*(*j* + **NCODA**, **NCODA** + 2).

Codiagonal Band Hermitian Storage Mode

This is an alternate storage mode for band Hermitian matrices. It is not used by any of the BLAS, see *Chapter 9, "Programming Notes for BLAS"* in the Math Library. In the codiagonal band Hermitian storage mode, the real and imaginary parts of the 2 * NCODA + 1 upper codiagonals and right-hand-side are stored in columns of a FORTRAN array. Note that there is no explicit use of the COMPLEX or the nonstandard data type DOUBLE COMPLEX data type in this storage mode.

For *Hermitian* complex matrices,

$$A = U + \sqrt{-1}\,V$$

where *U* and *V* are real matrices. They satisfy the conditions $U = U^T$ and $V = -V^T$. The right-hand-side

$$b = c + \sqrt{-1}\,d$$

where *c* and *d* are real vectors. The solution vector is denoted as

$$x = u + \sqrt{-1}\,v$$

where *u* and *v* are real. The storage is declared with the following statement

```
DIMENSION A(LDA, 2*NCODA + 3)
```

The parameter LDA is the *leading positive dimension* of *A*. It must be at least as large as N + NCODA.

The diagonal terms U_{jj} are stored in array locations *A* (*j* + NCODA, 1). The diagonal V_{jj} are zero and are not stored. The nonzero U_{ij}, $j > i$, are stored in locations *A*(*j* + NCODA, 2 * (*j* - *i*)).

The nonzero V_{ij} are stored in locations *A*(*j* + NCODA, 2*(*j* - *i*) + 1). The right side vector *b* is stored with c_j and d_j in locations *A*(*j* + NCODA, 2*NCODA + 2) and *A*(*j* + NCODA, 2*NCODA + 3) respectively. The real and imaginary parts of the solution, u_j and v_j, respectively overwrite c_j and d_j.

Consider a complex hermitian 5 × 5 matrix with 2 codiagonals

$$A = \begin{bmatrix} U_{11} & U_{12} & U_{13} & 0 & 0 \\ U_{12} & U_{22} & U_{23} & U_{24} & 0 \\ U_{13} & U_{23} & U_{33} & U_{34} & U_{35} \\ 0 & U_{24} & U_{34} & U_{44} & U_{45} \\ 0 & 0 & U_{35} & U_{45} & U_{55} \end{bmatrix} + \sqrt{-1} \begin{bmatrix} 0 & V_{12} & V_{13} & 0 & 0 \\ -V_{12} & 0 & V_{23} & V_{24} & 0 \\ -V_{13} & -V_{23} & 0 & V_{34} & V_{35} \\ 0 & -V_{24} & -V_{34} & 0 & V_{45} \\ 0 & 0 & -V_{35} & -V_{45} & 0 \end{bmatrix}$$

and a right-hand-side vector

$$b = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} + \sqrt{-1} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix}$$

A FORTRAN declaration for the array to hold this matrix and right-hand-side vector is

```
      PARAMETER (N = 5, NCODA = 2, LDA = N + NCODA)
      REAL A(LDA,2*NCODA + 3)
```

The matrix and right-hand-side entries are placed in the FORTRAN array *A* as follows:

$$A = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \\ U_{11} & \times & \times & \times & \times & c_1 & d_1 \\ U_{22} & U_{12} & V_{12} & \times & \times & c_2 & d_2 \\ U_{33} & U_{23} & V_{23} & U_{13} & V_{13} & c_3 & d_3 \\ U_{44} & U_{34} & V_{34} & U_{24} & V_{24} & c_4 & d_4 \\ U_{55} & U_{45} & V_{45} & U_{35} & V_{35} & c_5 & d_5 \end{bmatrix}$$

Entries marked with an × do not need to be defined.

Sparse Matrix Storage Mode

The sparse linear algebraic equation solvers in Chapter 1, "Basic Statistics" accept the input matrix in *sparse storage mode*. This structure consists of **INTEGER** values **N** and **NZ**, the matrix dimension and the total number of nonzero entries in the matrix. In addition, there are two **INTEGER** arrays **IROW**(*) and **JCOL**(*) that contain unique matrix row and column coordinates where values are given. There is also an array *A*(*) of values. All other entries of the matrix are zero. Each of the arrays **IROW**(*), **JCOL**(*), *A*(*) must be of size **NZ**. The correspondence between matrix and array entries is given by

$$A_{\mathrm{IROW(i)},\mathrm{JCOL(i)}} = A(i),\ i = 1, \ldots, \mathrm{NZ}$$

The data type for *A*(*) can be one of **REAL**, **DOUBLE PRECISION**, or **COMPLEX**. If your FORTRAN compiler allows, the nonstandard data type **DOUBLE COMPLEX** can also be declared.

For example, consider a real 5 × 5 sparse matrix with 11 nonzero entries. The matrix *A* has the form

$$A = \begin{bmatrix} A_{11} & 0 & A_{13} & A_{14} & 0 \\ A_{21} & A_{22} & 0 & 0 & 0 \\ 0 & A_{32} & A_{33} & A_{34} & 0 \\ 0 & 0 & A_{43} & 0 & 0 \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix}$$

Declarations of arrays and definitions of the values for this sparse matrix are

```
      PARAMETER (NZ = 11, N = 5)
      DIMENSION IROW(NZ), JCOL(NZ), A(NZ)
      DATA IROW /1,1,1,2,2,3,3,3,4,5,5/
      DATA JCOL /1,3,4,1,2,2,3,4,3,4,5/
      DATA A    /A11,A13,A14,A21,A22,A32,A33,A34, &
                 A43,A54,A55/
```

Reserved Names

When writing programs accessing the STAT LIBRARY, the user should choose FORTRAN names that do not conflict with names of IMSL subroutines, functions, or named common blocks, such as the workspace common block **WORKSP**. The user needs to be aware of two types of name conflicts that can arise. The first type of name conflict occurs when a name (technically a *symbolic name*) is not uniquely defined within a program unit (either a main program or a subprogram). For example, such a name conflict exists when the name **RCURV** is used to refer both to a type **REAL** variable and to the IMSL subroutine **RCURV** in a single program unit. Such errors are detected during compilation and are easy to correct. The second type of name conflict, which can be more serious, occurs when names of program units and named common blocks are not unique. For example, such a name conflict would be caused by the user defining a subroutine named **WORKSP** and also referencing an STAT/LIBRARY subroutine that uses the named common block **WORKSP**. Likewise, the user must not define a subprogram with the same name as a subprogram in the STAT/LIBRARY, that is referenced directly by the user's program or is referenced indirectly by other STAT/LIBRARY subprograms.

The STAT/LIBRARY consists of many routines, some that are described in the *User's Manual* and others that are not intended to be called by the user and, hence, that are not documented. If the choice of names were completely random over the set of valid FORTRAN names, and if a program uses only a small subset of the STAT/LIBRARY, the probability of name conflicts is very small. Since names are usually chosen to be mnemonic, however, the user may wish to take some precautions in choosing FORTRAN names.

Many IMSL names consist of a root name that may have a prefix to indicate the type of the routine. For example, the IMSL single precision subroutine for fitting a polynomial by least squares has the name **RCURV**, which is the root name, and the corresponding IMSL double precision routine has the name **DRCURV**. Associated with these two routines are **R2URV** and **DR2URV**. **RCURV** and **DRCURV** are listed in the Alphabetical Index of Routines, but **R2URV** and **DR2URV** are not. The user of **RCURV** must consider both names **RCURV** and **R2URV** to be reserved; likewise, the user of **DRCURV** must consider both names **DRCURV** and **DR2URV** to be reserved. The names of *all* routines and named common blocks that are used by the STAT/LIBRARY and that do not have a numeral in the second position of the root name are listed in the Alphabetical Summary of Routines.

The careful user can avoid any conflicts with IMSL names if the following rules are observed:

- Do not choose a name that appears in the Alphabetical Summary of Routines in the *User's Manual*, nor one of these names preceded by a **D**, **S_**, **D_**, **C_**, or **Z_**.
- Do not choose a name of three or more characters with a numeral in the second or third position.

These simplified rules include many combinations that are, in fact, allowable. However, if the user selects names that conform to these rules, no conflict will be encountered.

Deprecated Features and Renamed Routines

Automatic Workspace Allocation

FORTRAN subroutines that work with arrays as input and output often require extra arrays for use as workspace while doing computations or moving around data. IMSL routines generally do not require the user explicitly to allocate such arrays for use as workspace. On most systems the workspace allocation is handled transparently. The only limitation is the actual amount of memory available on the system.

On some systems the workspace is allocated out of a stack that is passed as a FORTRAN array in a named common block **WORKSP**. A very similar use of a workspace stack is described by Fox et al. (1978, pages 116-121). (For compatiblity with older versions of the IMSL Libraries, space is allocated from the **COMMON** block, if possible.)

The arrays for workspace appear as arguments in lower-level routines. For example, the IMSL routine **FREQ** (see Chapter 1, "Basic Statistics"), which computes frequency tabulations, needs arrays for workspace. **FREQ** allocates arrays from the common area and passes them to the lower-level routine **F2EQ**, which does the computations. In the "Comments" section of the documentation for **FREQ**, the amount of workspace is noted, and the call to **F2EQ** is described. This scheme for using lower-level routines is followed throughout the IMSL Libraries. The names of these routines have a "2" in the second position (or in the third position in double precision routines having a "**D**" prefix). The user can provide workspace explicitly and call directly the "2-level" routine, which is documented along with the main routine. In a very few cases, the 2-level routine allows additional options that the main routine does not allow.

Prior to returning to the calling program, a routine that allocates workspace generally deallocates that space, so that it becomes available for use in other routines. There are some exceptions to this, as noted in the section "**IDO** Routines" which follows later in this chapter.

Changing the Amount of Space Allocated

This section is relevant only to those systems on which the transparent workspace allocator is not available.

By default, the total amount of space allocated in the common area for storage of numeric data is 5000 numeric storage units. (A numeric storage unit is the amount of space required to store an integer or a real number. By comparison, a double precision unit is twice this amount. Therefore the total amount of space allocated in the common area for storage of numeric data is 2500 double precision units.) This space is allocated as needed for **INTEGER**, **REAL**, or other numeric data. For larger problems in which the default amount of workspace is insufficient, the user can change the allocation by supplying the FORTRAN statements to define the array in the named common block and by informing the IMSL workspace allocation system of the new size of the common array. To request 7000 units, the statements are

```
      COMMON /WORKSP/ RWKSP
      REAL RWKSP(7000)
      CALL IWKIN(7000)
```

If an IMSL routine attempts to allocate workspace in excess of the amount available in the common stack, the routine issues a fatal error message that indicates how much space is needed and prints statements like those above to guide the user in allocating the necessary amount. The program below uses IMSL routine PERMA (Chapter 19, "Utilities") to permute rows or columns of a matrix. This routine requires workspace equal to the number of columns, which in this example is too large. (Note that the work vector RWKSP must also provide extra space for bookkeeping.)

```
!                                  Specifications for local variables
      INTEGER    NRA, NCA, LDA, IPERMU(6000), IPATH
      REAL A(2,6000)
!                                  Specifications for subroutines
      EXTERNAL PERMA
!
      NRA = 2
      NCA = 6000
      LDA = 2
!                                  Initialize permutation index
      DO 10 I = 1, NCA
         IPERMU(I) = NCA + 1 - I
   10 CONTINUE
      IPATH = 2
      CALL PERMA (NRA, NCA, A, LDA, IPERMU, IPATH, A, LDA)
      END
```

Output

```
*** TERMINAL ERROR 10 from PERMA.  Insufficient workspace for current
***          allocation(s). Correct by calling IWKIN from main program with
***          the three following statements: (REGARDLESS OF PRECISION)
***                COMMON /WORKSP/ RWKSP
***                REAL RWKSP(6018)
***                CALL IWKIN(6018)
*** TERMINAL ERROR 10 from PERMA.  Workspace allocation was based on NCA =
***          6000.
```

In most cases, the amount of workspace is dependent on the parameters of the problem so the amount needed is known exactly. In a few cases, however, the amount of workspace is dependent on the data (for example, if it is necessary to count all of the unique values in a vector), so the IMSL routine cannot tell in advance exactly how much workspace is needed. In such cases the error message printed is an estimate of the amount of space required.

IDO Routines

Some routines with an argument named "**IDO**" allocate workspace automatically and store intermediate results in elements of workspace that are referenced in subsequent calls. Typically, these routines are called in a loop. With each call, some rows of the data set are input to the routine and statistics stored in workspace are updated. In this case, the workspace must be preserved between calls.

For these routines, when **IDO** indicates this is the first call, the routine allocates workspace; when **IDO** indicates this is the last call, the routine deallocates the workspace. Because of the way this workspace is allocated and deallocated, no IMSL routine requiring additional automatic workspace can be used between these two calls. If it is necessary to call additional routines requiring workspace, use the 2-level routines and explicitly allocate the work arrays.

Not all **IDO** routines require workspace to be preserved between their first and last call. Some may not even use workspace. Others may allocate and deallocate workspace with each call. The statement "workspace should not be changed between calls" will be in the description of the "**IDO**" routine that requires that workspace be preserved. (This statement will occur in the description of one or more of the workspace arguments for the 2-level routine.)

Character Workspace

Since character arrays cannot be equivalenced with numeric arrays, a separate named common block **WKSPCH** is provided for character workspace. In most respects this stack is managed in the same way as the numeric stack. The default size of the character workspace is 2000 character units. (A character unit is the amount of space required to store one character.) The routine analogous to **IWKIN** used to change the default allocation is **IWKCIN**.

The routines in the following list are being deprecated in Version 2.0 of STAT/LIBRARY. A deprecated routine is one that is no longer used by anything in the library but is being included in the product for those users who may be currently referencing it in their application. However, any future versions of STAT/LIBRARY will not include these routines. If any of these routines are being called within an application, it is recommended that you change your code or retain the deprecated routine before replacing this library with the next version. Most of these routines were called by users only when they needed to set up their own workspace. Thus, the impact of these changes should be limited.

DHOUAP

DHOUTR

DG2DF

DG2IN

DG3DF

G2DF

G2IN

G3DF

SHOUAP

SHOUTR

The following routines have been renamed due to naming conflicts with other software manufacturers.

CTIME — replaced with CPSEC

DTIME — replaced with TIMDY

PAGE — replaced with PGOPT

Appendix A Alphabetical Summary of Routines

Links to Sections

[A][B][C][D][E][F][G][H][I][K][L][M][N][O][P]
[Q][R][S][T][U][V][W]

A

Function	Purpose Statement
ABALD	Analyzes a balanced complete experimental design for a fixed, random, or mixed model.
ABIBD	Analyzes a balanced incomplete block design or a balanced lattice design.
ACF	Computes the sample autocorrelation function of a stationary time series.
ACHAR	Returns a character given its ASCII value
ACTBL	Produces population and cohort life tables.
ADNRM	Performs an Anderson-Darling test for normality.
AKS1DF	Evaluates the cumulative distribution function of the one-sided Kolmogorov-Smirnov goodness-of-fit D^+ or D^- test statistic based on continuous data for one sample.
AKS2DF	Evaluates the cumulative distribution function of the Kolmogorov-Smirnov goodness-of-fit D test statistic based on continuous data for two samples
ALATN	Analyzes a Latin square design.
ALNDF	Evaluates the lognormal cumulative probability distribution function.

Function	Purpose Statement
ALNIN	This function evaluates the inverse of the lognormal cumulative probability distribution function.
ALNPR	Evaluates the lognormal probability density function.
AMACH	Retrieves machine constants.
AMILLR	Evaluates Mill's ratio (the ratio of the ordinate to the upper tail area of the standardized normal distribution).
ANEST	Analyzes a completely nested random model with possibly unequal numbers in the subgroups.
ANORPR	Evaluates the normal probability density function.
ANORDF	Evaluates the standard normal (Gaussian) cumulative distribution function.
ANORIN	Evaluates the inverse of the standard normal (Gaussian) cumulative distribution function.
ANWAY	Analyzes a balanced *n*-way classification model with fixed effects.
AONEC	Analyzes a one-way classification model with covariates.
AONEW	Analyzes a one-way classification model.
ARMA_SPEC	Calculates the rational power spectrum for an ARMA model.
ARMME	Computes method of moments estimates of the autoregressive parameters of an ARMA model.
ATWOB	Analyzes a randomized block design or a two-way balanced design.
AUTO_ARIMA	Automatically identifies time series outliers, determines parameters of a multiplicative seasonal ARIMA $(p,0,q) \times (0,d,0)_s$ model, and produces forecasts that incorporate the effects of outliers whose effects persist beyond the end of the series.
AUTO_FPE_MUL_AR	Automatic selection and fitting of a multivariate autoregressive time series model using Akaike's Multivariate Final Prediction Error (MFPE) criteria.
AUTO_FPE_UNI_AR	Automatic selection and fitting of a univariate autoregressive time series model using Akaike's Final Prediction Error (FPE) criteria.
AUTO_MUL_AR	Automatic selection and fitting of a multivariate autoregressive time series model.
AUTO_PARM	Estimates structural breaks in non-stationary univariate time series.
AUTO_UNI_AR	Automatic selection and fitting of a multivariate autoregressive time series model.

B

Function	Purpose Statement
BAY_SEA	Allows for a decomposition of a time series into trend, seasonal, and an error component.
BCTR	Performs a forward or an inverse Box-Cox (power) transformation.
BETDF	Evaluates the beta cumulative distribution function.
BETIN	Evaluates the inverse of the beta cumulative distribution function.
BETNDF	This function evaluates the noncentral beta cumulative distribution function (CDF) .
BETNIN	This function evaluates the inverse of the noncentral beta cumulative distribution function (CDF).
BETNPR	This function evaluates the noncentral beta probability density function.
BETPR	Evaluates the beta probability density function.
BHAKV	Performs a Bhapkar *V* test.
BINDF	Evaluates the binomial cumulative distribution function.
BINES	Estimates the parameter *p* of the binomial distribution.
BINPR	Evaluates the binomial probability density function.
BNRDF	Evaluates the bivariate normal cumulative distribution function.
BOXP	Prints boxplots for one or more samples.
BSCAT	Computes the biserial correlation coefficient for a dichotomous variable and a classification variable.
BSPBS	Computes the biserial and point-biserial correlation coefficients for a dichotomous variable and a numerically measurable classification variable.

C

Function	Purpose Statement
CANCOR	Given an input array of deviate values, generates a canonical correlation array.
CANCR	Performs canonical correlation analysis from a data matrix.
CANVC	Performs canonical correlation analysis from a variance-covariance matrix or a correlation matrix.
CCF	Computes the sample cross-correlation function of two stationary time series.
CDF2P	Prints a plot of two sample cumulative distribution functions.
CDFP	Prints a sample cumulative distribution function (CDF), a theoretical CDF, and confidence band information.
CDIST	Computes a matrix of dissimilarities (or similarities) between the columns (or rows) of a matrix.
CESTI	Constructs an equivalent completely testable multivariate general linear hypothesis $HBU = G$ from a partially testable hypothesis $H_pBU = G_p$.
CHFAC	Computes an upper triangular factorization of a real symmetric nonnegative definite matrix.
CHIDF	Evaluates the chi-squared cumulative distribution function.
CHIGF	Performs a chi-squared goodness-of-fit test.
CHIIN	Evaluates the inverse of the chi-squared cumulative distribution function.
CHIPR	Evaluates the chi-squared probability density function.
CIDMS	Computes a confidence interval on a variance component estimated as proportional to the difference in two mean squares in a balanced complete experimental design.
CLINK	Performs a hierarchical cluster analysis given a distance matrix.
CNCRD	Calculates and tests the significance of the Kendall coefficient of concordance.
CNUMB	Computes cluster membership for a hierarchical cluster tree.
CORVC	Computes the variance-covariance or correlation matrix.
COVPL	Computes a pooled variance-covariance matrix from the observations.
CPFFT	Computes the cross periodogram of two stationary time series using a fast Fourier transform.
CPSEC	Returns CPU time used in seconds.

Function	Purpose Statement
CSNDF	Evaluates the noncentral chi-squared cumulative distribution function.
CSNIN	Evaluates the inverse of the noncentral chi-squared cumulative function.
CSNPR	This function evaluates the noncentral chi-squared probability density function.
CSSWD	Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the time series data.
CSSWP	Estimates the nonnormalized cross-spectral density of two stationary time series using a spectral window given the spectral densities and cross periodogram.
CSTAT	Computes cell frequencies, cell means, and cell sums of squares for multivariate data.
CSWED	Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the time series data.
CSWEP	Estimates the nonnormalized cross-spectral density of two stationary time series using a weighted cross periodogram given the spectral densities and cross periodogram.
CTASC	Computes partial association statistics for log-linear models in a multidimensional contingency table.
CTCHI	Performs a chi-squared analysis of a two-way contingency table.
CTEPR	Computes Fisher's exact test probability and a hybrid approximation to the Fisher exact test probability for a contingency table using the network algorithm.
CTGLM	Analyzes categorical data using logistic, Probit, Poisson, and other generalized linear models.
CTLLN	Computes model estimates and associated statistics for a hierarchical log-linear model.
CTPAR	Computes model estimates and covariances in a fitted log-linear model.
CTPRB	Computes exact probabilities in a two-way contingency table.
CTRAN	Performs generalized Mantel-Haenszel tests in a stratified contingency table.
CTRHO	Estimates the bivariate normal correlation coefficient using a contingency table.
CTRST	Computes contrast estimates and sums of squares.
CTSTP	Builds hierarchical log-linear models using forward selection, backward selection, or stepwise selection.
CTTWO	Performs a chi-squared analysis of a 2 by 2 contingency table.

Function	Purpose Statement
CTWLS	Performs a generalized linear least squares analysis of transformed probabilities in a two-dimensional contingency table.
CVMNRM	Performs a Cramer-von Mises test for normality.
CVTSI	Converts a character string containing an integer number into the corresponding integer form.

D

Function	Purpose Statement
DCUBE	Performs a triplets test.
DESKN	Performs nonparametric probability density function estimation by the kernel method.
DESPL	Performs nonparametric probability density function estimation by the penalized likelihood method.
DESPT	Estimates a probability density function at specified points using linear or cubic interpolation.
DIFF	Difference a time series.
DIRIC	Computes the Dirichlet kernel.
DMACH	See AMACH.
DMSCR	Uses Fisher's linear discriminant analysis method to reduce the number of variables.
DNFFT	Computes Gaussian kernel estimates of a univariate density via the fast Fourier transform over a fixed interval.
DSCRM	Performs a linear or a quadratic discriminant function analysis among several known groups.
DSQAR	Performs a D-square test.

E

Function	Purpose Statement
ENOS	Evaluates the expected value of a normal order statistic.
EQTIL	Computes empirical quantiles.
ERSET	Sets error handler default print and stop actions.
ESTIMATE_MISSING	Estimates missing values in a time series.
EXPDF	Evaluates the exponential cumulative distribution function.
EXPIN	Evaluates the inverse of the exponential cumulative distribution function.
EXPPR	This function evaluates the exponential probability density function.
EXVDF	Evaluates the extreme value cumulative distribution function.
EXVIN	Evaluates the inverse of the extreme value cumulative distribution function.
EXVPR	Evaluates the extreme value probability density function.

F

Function	Purpose Statement
FACTR	Extracts initial factor-loading estimates in factor analysis.
FAURE_FREE	Frees the structure containing information about the Faure sequence.
FAURE_INIT	Computes a shuffled Faure sequence.
FAURE_NEXT	Shuffled Faure sequence initialization.
FCOEF	Computes a matrix of factor score coefficients for input to the following IMSL routine (FSCOR).
FDF	Evaluates the *F* cumulative distribution function.
FDOBL	Computes a direct oblimin rotation of a factor-loading matrix.
FEJER	Computes the Fejér kernel.
FGCRF	Computes direct oblique rotation according to a generalized fourth-degree polynomial criterion.

Function	Purpose Statement
FHARR	Computes an oblique rotation of an unrotated factor-loading matrix using the Harris-Kaiser method.
FIMAG	Computes the image transformation matrix.
FIN	Evaluates the inverse of the *F* cumulative distribution function.
FNDF	Noncentral *F* cumulative distribution function.
FNIN	This function evaluates the inverse of the noncentral *F* cumulative distribution function (CDF).
FNPR	This function evaluates the noncentral F cumulative distribution function (CDF).
FOPCS	Computes an orthogonal Procrustes rotation of a factor-loading matrix using a target matrix.
FPR	Evaluates the *F* probability density function.
FPRMX	Computes an oblique Promax or Procrustes rotation of a factor-loading matrix using a target matrix, including pivot and power vector options.
FRDMN	Performs Friedman's test for a randomized complete block design.
FREQ	Tallies multivariate observations into a multi-way frequency table.
FRESI	Computes commonalities and the standardized factor residual correlation matrix
FROTA	Computes an orthogonal rotation of a factor-loading matrix using a generalized orthomax criterion, including quartimax, varimax, and equamax rotations.
FRVAR	Computes the factor structures and the variance explained by each factor.
FSCOR	Computes a set of factor scores given the factor score coefficient matrix.

G

Function	Purpose Statement
GAMDF	Evaluates the gamma cumulative distribution function.
GAMIN	Evaluates the inverse of the gamma cumulative distribution function.

Function	Purpose Statement
GAMPR	Evaluates the gamma probability density function.
GARCH	Computes estimates of the parameters of a GARCH (*p*,*q*) model.
GCDF	Evaluates a general continuous cumulative distribution function given ordinates of the density.
GCIN	Evaluates the inverse of a general continuous cumulative distribution function given ordinates of the density.
GCLAS	Gets the unique values of each classification variable.
GCSCP	Generates centered variables, squares, and crossproducts.
GDATA	Retrieves a commonly analyzed data set.
GEODF	Evaluates the geometric cumulative probability distribution function.
GEOIN	Evaluates the inverse of the geometric cumulative probability distribution function.
GEOPR	Evaluates the geometric probability density function.
GFNIN	Evaluates the inverse of a general continuous cumulative distribution function given in a subprogram.
GIRTS	Solves a triangular (possibly singular) set of linear systems and/or compute a generalized inverse of an upper triangular matrix.
GRGLM	Generates regressors for a general linear model.
GRPES	Computes basic statistics from grouped data.
GSWEP	Performs a generalized sweep of a row of a nonnegative definite matrix.

H

Function	Purpose Statement
HAZEZ	Performs nonparametric hazard rate estimation using kernel functions. Easy-to-use version of the previous IMSL subroutine (HAZRD).
HAZRD	Performs nonparametric hazard rate estimation using kernel functions and quasi-likelihoods.
HAZST	Performs hazard rate estimation over a grid of points using a kernel function.

Function	Purpose Statement
HHSTP	Prints a horizontal histogram
HYPDF	Evaluates the hypergeometric cumulative distribution function.
HYPPR	Evaluates the hypergeometric probability function.

Function	Purpose Statement
IACHAR	Returns the integer ASCII value of a character argument.
ICASE	Returns the ASCII value of a character converted to uppercase.
IDYWK	Computes the day of the week for a given date.
IERCD and N1RTY	Retrieves the code for an informational error.
IFNAN(X)	Checks if a floating-point number is NaN (not a number).
IICSR	Compares two character strings using the ASCII collating sequence without regard to case.
IIDEX	Determines the position in a string at which a given character sequence begins without regard to case.
IMACH	Retrieves integer machine constants.
INCLD	Performs an includance test.
IRNSE	Computes estimates of the impulse response weights and noise series of a univariate transfer function model.
ISRCH	Searches a sorted integer vector for a given integer and returns its index.

K

Function	Purpose Statement
KALMN	Performs Kalman filtering and evaluate the likelihood function for the state-space model.
KAPMR	Computes Kaplan-Meier estimates of survival probabilities in stratified samples.
KENDL	Computes and tests Kendall's rank correlation coefficient.
KENDP	Computes the frequency distribution of the total score in Kendall's rank correlation coefficient.
KMEAN	Performs a *K*-means (centroid) cluster analysis.
KPRIN	Maximum likelihood or least-squares estimates for principle components from one or more matrices.
KRSKL	Performs a Kruskal-Wallis test for identical population medians.
KSONE	Performs a Kolmogorov-Smirnov one-sample test for continuous distributions.
KSTWO	Performs a Kolmogorov-Smirnov two-sample test.
KTBLE	Prints Kaplan-Meier estimates of survival probabilities in stratified samples.
KTRND	Performs a *k*-sample trends test against ordered alternatives.

L

Function	Purpose Statement
LETTR	Produces a letter value summary.
LILLF	Performs Lilliefors test for an exponential or normal distribution.
LOFCF	Performs lack-of-fit test for a univariate time series or transfers function given the appropriate correlation function.

M

Function	Purpose Statement
MAMME	Computes method of moments estimates of the moving average parameters of an ARMA model.
MAX_ARMA	Exacts maximum likelihood estimation of the parameters in a univariate ARMA (auto-regressive, moving average) time series model.
MCCF	Computes the multichannel cross-correlation function of two mutually stationary multichannel time series.
MCHOL	Computes an upper triangular factorization of a real symmetric matrix *A* plus a diagonal matrix *D*, where *D* is determined sequentially during the Cholesky factorization in order to make *A* + *D* nonnegative definite.
MEDPL	Computes a median polish of a two-way table.
MLSE	Computes least squares estimates of a linear regression model for a multichannel time series with a specified base channel.
MLE	Calculates maximum likelihood estimates for the parameters of one of several univariate probability distributions.
MSDBL	Obtains normalized product-moment (double centered) matrices from dissimilarity matrices.
MSDST	Computes distances in a multidimensional scaling model.
MSIDV	Performs individual-differences multidimensional scaling for metric data using alternating least squares.
MSINI	Computes initial estimates in multidimensional scaling models.
MSSTN	Transforms dissimilarity/similarity matrices and replace missing values by estimates to obtain standardized dissimilarity matrices.
MSTRS	Computes various stress criteria in multidimensional scaling.
MVIND	Computes a test for the independence of *k* sets of multivariate normal variables.
MVMMT	Computes Mardia's multivariate measures of skewness and kurtosis and tests for multivariate normality.
MVNAN	Moves any rows of a matrix with the IMSL missing value code NaN (not a number) in the specified columns to the last rows of the matrix.
MWFE	Computes least squares estimates of the multichannel Wiener filter coefficients for two mutually stationary multichannel time series.

N

Function	Purpose Statement
IERCD and N1RTY	Retrieves an error type for the most recently called IMSL routine.
NCTRD	Performs the Noether test for cyclical trend.
NDAYS	Computes the number of days from January 1, 1900, to the given date.
NDYIN	Gives the date corresponding to the number of days since January 1, 1900.
NGHBR	Searches a *k*-*d* tree for the *k* nearest neighbors of a key.
NNBRD	Performs a *k* nearest neighbor discrimination.
NRCES	Computes maximum likelihood estimates of the mean and variance from grouped and/or censored normal data.
NSBJF	Computes Box-Jenkins forecasts and their associated probability limits for a nonseasonal ARMA model.
NSLSE	Computes least squares estimates of parameters for a nonseasonal ARMA model.
NSPE	Computes preliminary estimates of the autoregressive and moving average parameters of an ARMA model.
NTIES	Computes tie statistics for a sample of observations.

O

Function	Purpose Statement
OPOLY	Generates orthogonal polynomials with respect to *x* values and specified weights.
OPT_DES	Allows for multiple channels for both the controlled and manipulated variables.
ORDST	Determines order statistics.
OWFRQ	Tallies observations into a one-way frequency table.

P

Function	Purpose Statement
PACF	Computes the sample partial autocorrelation function of a stationary time series.
PAIRS	Performs a pairs test.
PCORR	Computes partial correlations or covariances from the covariance or correlation matrix.
PERMA	Permutes the rows or columns of a matrix.
PERMU	Rearranges the elements of an array as specified by a permutation.
PFFT	Computes the periodogram of a stationary time series using a fast Fourier transform.
PGOPT	Sets or retrieves page width and length for printing.
PHGLM	Analyzes time event data via the proportional hazards model.
PLOTP	Prints a plot of up to ten sets of points.
PLSR	Performs partial least squares regression for one or more response variables and a set of one or more predictor variables.
POIDF	Evaluates the Poisson cumulative distribution function.
POIES	Estimates the parameter of the Poisson distribution.
POIPR	Evaluates the Poisson probability density function.
PRINC	Computes principal components from a variance-covariance matrix or a correlation matrix.
PROBP	Prints a probability plot.
PRPFT	Performs iterative proportional fitting of a contingency table using a loglinear model.

Q

Function	Purpose Statement
QTEST	Performs a Cochran *Q* test for related observations.
QUADT	Forms a *k-d* tree.

R

Function	Purpose Statement
RALDF	Evaluates the Rayleigh cumulative distribution function.
RALIN	Evaluates the inverse of the Rayleigh cumulative distribution function.
RALPR	Evaluates the Rayleigh probability density function.
RANKS	Computes the ranks, normal scores, or exponential scores for a vector of observations.
RBCOV	Computes a robust estimate of a covariance matrix and mean vector.
RBEST	Selects the best multiple linear regression models.
RCASE	Computes case statistics and diagnostics given data points, coefficient estimates $\hat{\beta}$, and the *R* matrix for a fitted general linear model.
RCASP	Computes case statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RCOMP	Generates an orthogonal central composite design.
RCOV	Fits a multiple linear regression model given the variance-covariance matrix.
RCOVB	Computes the estimated variance-covariance matrix of the estimated regression coefficients given the *R* matrix.
RCURV	Fits a polynomial curve using least squares.
REG_ARIMA	Fits a univariate, non seasonal ARIMA time series model with the inclusion of one or more regression variables.
RFORP	Fits an orthogonal polynomial regression model.
RGIVN	Fits a multivariate linear regression model via fast Givens transformations.
RGLM	Fits a multivariate general linear model.
RHPSS	Computes the matrix of sums of squares and crossproducts for the multivariate general linear hypothesis *HBU* = *G* given the coefficient estimates $\hat{\beta}$ and the *R* matrix.
RHPTE	Performs tests for a multivariate general linear hypothesis *HBU* = *G* given the hypothesis sums of squares and crossproducts matrix S_H and the error sums of squares and crossproducts matrix S_E.
RINCF	Performs response control given a fitted simple linear regression model.

Function	Purpose Statement
RINPF	Performs inverse prediction given a fitted simple linear regression model.
RLAV	Fits a multiple linear regression model using the least absolute values criterion.
RLEQU	Fits a multivariate linear regression model with linear equality restrictions $H\beta = G$ imposed on the regression parameters given results from IMSL routine RGIVN after IDO = 1 and IDO = 2 and prior to IDO = 3.
RLINE	Fits a line to a set of data points using least squares.
RLLP	Fits a multiple linear regression model using the L_p norm criterion.
RLMV	Fits a multiple linear regression model using the minimax criterion.
RLOFE	Computes a lack-of-fit test based on exact replicates for a fitted regression model.
RLOFN	Computes a lack-of-fit test based on near replicates for a fitted regression model.
RLSE	Fits a multiple linear regression model using least squares.
RNARM	Generates a time series from a specified ARMA model.
RNBET	Generates pseudorandom numbers from a beta distribution.
RNBIN	Generates pseudorandom numbers from a binomial distribution.
RNCHI	Generates pseudorandom numbers from a chi-squared distribution.
RNCHY	Generates pseudorandom numbers from a Cauchy distribution.
RNCOR	Generates a pseudorandom orthogonal matrix or a correlation matrix.
RNDAT	Generates pseudorandom numbers from a multivariate distribution determined from a given sample.
RNEXP	Generates pseudorandom numbers from a standard exponential distribution.
RNEXT	Generates pseudorandom numbers from a mixture of two exponential distributions.
RNEXV	Generates pseudorandom numbers from an extreme value distribution.
RNFDF	Generates pseudorandom numbers from the F distribution.
RNGAM	Generates pseudorandom numbers from a standard gamma distribution.
RNGCS	Sets up table to generate pseudorandom numbers from a general continuous distribution.

Function	Purpose Statement
RNGCT	Generates pseudorandom numbers from a general continuous distribution.
RNGDA	Generates pseudorandom numbers from a general discrete distribution using an alias method.
RNGDS	Sets up table to generate pseudorandom numbers from a general discrete distribution.
RNGDT	Generates pseudorandom numbers from a general discrete distribution using a table lookup method.
RNGEF	Retrieves the current value of the array used in the IMSL GFSR random number generator.
RNGEO	Generates pseudorandom numbers from a geometric distribution.
RNGES	Retrieves the current value of the table in the IMSL random number generators that use shuffling.
RNGET	Retrieves the current value of the seed used in the IMSL random number generators.
RNHYP	Generates pseudorandom numbers from a hypergeometric distribution.
RNIN32	Initializes the 32-bit Mersenne Twister generator using an array.
RNGE32	Retrieves the current table used in the 32-bit Mersenne Twister generator.
RNSE32	Sets the current table used in the 32-bit Mersenne Twister generator.
RNIN64	Initializes the 32-bit Mersenne Twister generator using an array.
RNGE64	Retrieves the current table used in the 64-bit Mersenne Twister generator
RNSE64	Sets the current table used in the 64-bit Mersenne Twister generator.
RNISD	Determines a seed that yields a stream beginning 100,000 numbers beyond the beginning of the stream yielded by a given seed used in IMSL multiplicative congruential generators (with no shufflings).
RNKSM	Performs the Wilcoxon rank sum test.
RNLGR	Generates pseudorandom numbers from a logarithmic distribution.
RNLIN	Fits a nonlinear regression model.
RNLNL	Generates pseudorandom numbers from a lognormal distribution.
RNMTN	Generates pseudorandom numbers from a multinomial distribution.

Function	Purpose Statement
RNMVGC	Generates pseudorandom numbers from a multivariate Gaussian Copula distribution.
RNMVTC	Generates a length N output vector R of pseudorandom numbers from a Student's *t* Copula distribution.
RNMVN	Generates pseudorandom numbers from a multivariate normal distribution.
RNNBN	Generates pseudorandom numbers from a negative binomial distribution.
RNNOA	Generates pseudorandom numbers from a standard normal distribution using an acceptance/rejection method.
RNNOF	Generates a pseudorandom number from a standard normal distribution.
RNNOR	Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.
RNNOS	Generates pseudorandom order statistics from a standard normal distribution.
RNNPP	Generates pseudorandom numbers from a nonhomogeneous Poisson process.
RNOPG	Retrieves the indicator of the type of uniform random number generator.
RNOPT	Selects the uniform (0, 1) multiplicative congruential pseudorandom number generator.
RNPER	Generates a pseudorandom permutation.
RNPOI	Generates pseudorandom numbers from a Poisson distribution.
RNRAL	Generates pseudorandom numbers from a Rayleigh distribution.
RNSEF	Initializes the array used in the IMSL GFSR random number generator.
RNSES	Initializes the table in the IMSL random number generators that use shuffling.
RNSET	Initializes a random seed for use in the IMSL random number generators.
RNSPH	Generates pseudorandom points on a unit circle or K-dimensional sphere.
RNSRI	Generates a simple pseudorandom sample of indices.
RNSRS	Generates a simple pseudorandom sample from a finite population.
RNSTA	Generates pseudorandom numbers from a stable distribution.

Function	Purpose Statement
RNSTT	Generates pseudorandom numbers from a Student's *t* distribution.
RNTAB	Generates a pseudorandom two-way table.
RNTRI	Generates pseudorandom numbers from a triangular distribution on the interval (0,1).
RNUN	Generates pseudorandom numbers from a uniform (0,1) distribution.
RNUND	Generates pseudorandom numbers from a discrete uniform distribution.
RNUNF	Generates a pseudorandom number from a uniform (0, 1) distribution.
RNUNO	Generates pseudorandom order statistics from a uniform (0, 1) distribution.
RNVMS	Generates pseudorandom numbers from a von Mises distribution.
RNWIB	Generates pseudorandom numbers from a Weibull distribution.
RONE	Analyzes a simple linear regression model.
RORDM	Reorders rows and columns of a symmetric matrix.
ROREX	Reorders the responses from a balanced complete experimental design.
ROTIN	Computes diagnostics for detection of outliers and influential data points given residuals and the *R* matrix for a fitted general linear model.
RPOLY	Analyzes a polynomial regression model.
RSTAP	Computes summary statistics for a polynomial regression model given the fit based on orthogonal polynomials.
RSTAT	Computes statistics related to a regression fit given the coefficient estimates $\hat{\beta}$ and the *R* matrix.
RSTEP	Builds multiple linear regression models using forward selection, backward selection, or stepwise selection.
RSUBM	Retrieves a symmetric submatrix from a symmetric matrix.
RUNS	Performs a runs up test.

S

Function	Purpose Statement
SCIPM	Computes simultaneous confidence intervals on all pairwise differences of means.
SCOLR	Sorts columns of a real rectangular matrix using keys in rows.
SCTP	Prints a scatterplot of several groups of data.
SDPLC	Performs the Cox and Stuart sign test for trends in dispersion and location.
SEASONAL_FIT	Determines an optimal differencing for seasonal adjustments of a time series.
SIGNT	Performs a sign test of the hypothesis that a given value is a specified quantile of a distribution.
SMPPR	Computes statistics for inferences regarding the population proportion and total, given proportion data from a simple random sample.
SMPPS	Computes statistics for inferences regarding the population proportion and total, given proportion data from a stratified random sample.
SMPRR	Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, or inferences regarding the population ratio, given a simple random sample.
SMPRS	Computes statistics for inferences regarding the population mean and total using ratio or regression estimation, given continuous data from a stratified random sample.
SMPSC	Computes statistics for inferences regarding the population mean and total using single-stage cluster sampling with continuous data.
SMPSR	Computes statistics for inferences regarding the population mean and total, given data from a simple random sample.
SMPSS	Computes statistics for inferences regarding the population mean and total, given data from a stratified random sample.
SMPST	Computes statistics for inferences regarding the population mean and total, given continuous data from a two-stage sample with equisized primary units.
SNKMC	Performs Student-Newman-Keuls multiple comparison test.
SNRNK	Performs a Wilcoxon signed rank test.
SPWF	Computes the Wiener forecast operator for a stationary stochastic process.

Function	Purpose Statement
SPWLK	Performs a Shapiro-Wilk *W*-test for normality.
SRCH	Searches a sorted vector for a given scalar and return its index.
SROWR	Sorts rows of a real rectangular matrix using keys in columns.
SSRCH	Searches a character vector, sorted in ascending ASCII order, for a given string and return its index.
SSWD	Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the time series data.
SSWP	Estimates the nonnormalized spectral density of a stationary time series using a spectral window given the periodogram.
STBLE	Estimates survival probabilities and hazard rates for various parametric models.
STMLP	Prints a stem-and-leaf plot.
SVGLM	Analyzes censored survival data using a generalized linear model.
SVIGN	Sorts an integer array by algebraic value.
SVIGP	Sorts an integer array by algebraic value and returns the permutations.
SVRGN	Sorts a real array by algebraic value.
SVRGP	Sorts a real array by algebraic value and returns the permutations.
SWED	Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the time series data.
SWEP	Estimation of the nonnormalized spectral density of a stationary time series based on specified periodogram weights given the periodogram.

T

Function	Purpose Statement
TCSCP	Transforms coefficients from a quadratic regression model generated from squares and crossproducts of centered variables to a model using uncentered variables.
TDATE	Gets today's date.
TDF	Evaluates the Student's *t* cumulative distribution function.

Function	Purpose Statement
TETCC	Categorizes bivariate data and compute the tetrachoric correlation coefficient.
TFPE	Computes preliminary estimates of parameters for a univariate transfer function model.
TIMDY	Gets time of day.
TIN	Evaluates the inverse of the Student's *t* distribution function.
TNDF	Evaluates the noncentral Student's *t* cumulative distribution function.
TNIN	Evaluates the inverse of the noncentral Student's *t* cumulative distribution function.
TNPR	This function evaluates the noncentral Student's *t* probability density function.
TPR	This function evaluates the Student's *t* probability density function.
TREEP	Prints a binary tree.
TRNBL	Computes Turnbull's generalized Kaplan-Meier estimates of survival probabilities in samples with interval censoring.
TS_OUTLIER_FORECAST	Detects and determines outliers and simultaneously estimates the model parameters in a time series.
TS_OUTLIER_IDENTIFICATION	Computes forecasts for an outlier contaminated time series.
TWFRQ	Tallies observations into a two-way frequency table.
TWOMV	Computes statistics for mean and variance inferences using samples from two normal populations.

U

Function	Purpose Statement
UMACH	Sets or retrieves input or output device unit numbers.
UNDDF	Evaluates the discrete uniform cumulative distribution function.
UNDF	Evaluates the uniform cumulative distribution function.
UNDIN	Evaluates the inverse of the discrete uniform cumulative distribution function.
UNDPR	Evaluates the discrete uniform probability density function.

Function	Purpose Statement
UNIN	Evaluates the inverse of the uniform cumulative distribution function.
UNPR	Evaluates the uniform probability density function.
UVSTA	Computes basic univariate statistics.

V

Function	Purpose Statement
VERSL	Obtains STAT/LIBRARY-related version and system information.
VHS2P	Prints a vertical histogram with every bar subdivided into two parts.
VHSTP	Prints a vertical histogram.

W

Function	Purpose Statement
WBLDF	Evaluates the Weibull cumulative distribution function.
WBLIN	Evaluates the inverse of the Weibull cumulative distribution function.
WBLPR	Evaluates the Weibull probability density function.
WRIRL	Prints an integer rectangular matrix with a given format and labels.
WRIRN	Prints an integer rectangular matrix with integer row and column labels.
WROPT	Sets or retrieves an option for printing a matrix.
WRRRL	Prints a real rectangular matrix with a given format and labels.
WRRRN	Prints a real rectangular matrix with integer row and column labels.

Appendix B References

Abdi

Abdi, Herve (2010), *Partial least squares regression and projection on latent structure regression (PLS regression)*, Wiley Interdisciplinary Reviews: Computational Statistics, 2, 97-106.

Abraham and Ledolter

Abraham, Bovas, and Johannes Ledolter (1983), *Statistical Methods for Forecasting*, John Wiley & Sons, New York.

Abramowitz and Stegun

Abramowitz, Milton, and Irene A. Stegun (editors) (1964), *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, Washington.

Afifi and Azen

Afifi, A.A. and S.P. Azen (1979), *Statistical Analysis: A Computer Oriented Approach*, second edition, Academic Press, New York.

Agresti, Wackerly, and Boyette

Agresti, Alan, Dennis Wackerly, and James M. Boyette (1979), Exact conditional tests for cross-classifications: Approximation of attained significance levels, *Psychometrika*, **44**, 75-83.

Ahrens and Dieter

Ahrens, J.H., and U. Dieter (1974), Computer methods for sampling from gamma, beta, Poisson, and binomial distributions, *Computing*, **12**, 223-246.

Ahrens, J.H., and U. Dieter (1985), Sequential random sampling, *ACM Transactions on Mathematical Software*, **11**, 157-169.

Aird and Howell

Aird, Thomas J., and Byron W. Howell (1991), IMSL Technical Report 9103, IMSL, Houston.

Akaike

Akaike, H. (1980) Seasonal Adjustment by Bayesian Modeling. Journal of Time Series Analysis, Vol 1, 1-13.

Akaike, H., et al

Akaike, H., Kitagawa, G., Arahata, E., and Tada, F. (1979) Computer Science Monographs, No. 13, Institute of Statistical Mathematics, Tokyo.

Akaike and Nakagawa

Akaike, H. and Nakagawa, T. (1972) *Statistical Analysis and Control of Dynamic Systems*, KTK Scientific Publishers, Tokyo.

Akima

Akima, Hirosha (1970), A new method of interpolation and smooth curve fitting based on local procedures, *Journal of the ACM*, **17**, 589 - 602.

Anderberg

Anderberg, Michael R. (1973), *Cluster Analysis for Applications*, Academic Press, New York.

Anderson

Anderson, T.W. (1971), *The Statistical Analysis of Time Series*, John Wiley & Sons, New York.

Anderson and Bancroft

Anderson, R.L., and T.A. Bancroft (1952), *Statistical Theory in Research*, McGraw-Hill Book Company, New York.

Anderson and Rubin

Anderson, T., and H. Rubin (1956), Statistical inference in factor analysis, *Proceedings of the Third Berkely Symposium on Mathematical Statistics and Probability*, Volume 5, University of California Press, Berkeley, 111 - 150.

Atkinson

Atkinson, A.C. (1973), Testing transformations to normality, *Journal of the Royal Statistical Society*, Series B: Methodological, **35**, 473 - 479.

Atkinson, A.C. (1979), A family of switching algorithms for the computer generation of beta random variates, *Biometrika*, **66**, 141 - 145.

Atkinson, A.C. (1985), *Plots, Transformations, and Regression*, Claredon Press, Oxford.

Atkinson, A.C. (1986), Diagnostic tests for transformations, *Technometrics*, **28**, 29 - 37.

Baker, Clarke, and Lane

Baker, R.J., M.R.B. Clarke, and P.W. Lane (1985). Zero entries in contingency tables, *Computational Statistics and Data Analysis*, **3**, 33-45.

Bartlett

Bartlett, M.S. (1935), Contingency table interactions, *Journal of the Royal Statistics Society Supplement*, **2**, 248-252.

Bartlett, M. (1937), The statistical conception of mental factors, *British Journal of Psychology*, **28**, 97 - 104.

Bartlett, M.S. (1946), On the theoretical specification and sampling properties of autocorrelated time series, *Supplement to the Journal of the Royal Statistical Society*, **8**, 27 - 41.

Bartlett, M.S. (1978), *Stochastic Processes*, 3rd. ed., Cambridge University Press, Cambridge.

Barrodale and Roberts

Barrodale, I., and F.D.K. Roberts (1973), An improved algorithm for discrete *L*1 approximation, *SIAM Journal on Numerical Analysis*, **10**, 839-848.

Barrodale, I., and C. Phillips (1975), Algorithm 495. Solution of an overdetermined system of linear equations in the Chebyshev norm, *ACM Transactions on Mathematical Software*, **1**, 264-270.

Barrodale, I., and F.D.K. Roberts (1974), Solution of an overdetermined system of equations in the *l*1 norm, *Communications of the ACM*, **17**, 319-320.

Barlow et al.

Barlow, R.E., D.J. Bartholomew, J.M. Bremner, and H.D. Brunk (1972), *Statistical Inference under Order Restrictions*, John Wiley & Sons, London.

Bendel and Mickey

Bendel, Robert B., and M. Ray Mickey (1978), Population correlation matrices for sampling experiments, *Communications in Statistics*, **B7**, 163-182.

Berk

Berk, Kenneth. N. (1976), Tolerance and condition in regression computations, *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, Prindle, Weber and Schmidt, Boston, 202-203.

Best and Fisher

Best, D.J., and N.I. Fisher (1979), Efficient simulation of the von Mises distribution, *Applied Statistics*, **28**, 152-157.

Bhapkar

Bhapkar, V.P. (1961), A nonparametric test for the problem of several samples, *Annals of Mathematical Statistics*, **32**, 1108-1117.

Bishop, Feinberg, and Holland

Bishop, Yvonne M. M., Stephen E. Feinberg, and Paul W. Holland (1975), *Discrete Multivariate Analysis*, The MIT Press, Cambridge, Mass.

Bjorck and Golub

Bjorck, Ake, and Gene H. Golub (1973), Numerical Methods for Computing Angles Between Subspaces, *Mathematics of Computation*, **27**, 579-594.

Blackman and Tukey

Blackman, R.B., and J. W. Tukey (1958), *The Measurement of Power Spectra from the Point of View of Communications Engineering*, Dover Publications, New York.

Blom

Blom, Gunnar (1958), *Statistical Estimates and Transformed Beta-Variables*, John Wiley & Sons, New York.

Boisvert, Howe, and Kahaner

Boisvert, Ronald F., Sally E. Howe, and David K. Kahaner (1985), GAMS: A framework for the management of scientific software, *ACM Transactions on Mathematical Software*, **11**, 313-355.

Boisvert, Howe, Kahaner, and Springmann

Boisvert, Ronald F., Sally E. Howe, David K. Kahaner, and Jeanne L. Springmann (1990), *Guide to Available Mathematical Software*, NISTIR 90-4237, National Institute of Standards and Technology, Gaithersburg, Maryland.

Bosten and Battiste

Bosten, Nancy E., and E.L. Battiste (1974), Incomplete beta ratio, *Communications of the ACM* **17**, 156 - 157.

Box and Cox

Box, G.E.P., and D.R. Cox (1964), An analysis of transformations, *Journal of the Royal Statistical Society*, Series B: *Methodological*, **26**, 211 - 243.

Box et al.

Box, George E.P., Jenkins, Gwilym M. and Reinsel G.C., (1994) *Time Series Analysis*, Third edition, Prentice Hall, Englewood Cliffs, New Jersey, **547**.

Box and Jenkins

Box, George E.P., and Gwilym M. Jenkins (1976), *Time Series Analysis: Forecasting and Control*, rev. ed., Holden-Day, Oakland, Calif.

Box and Pierce

Box, G.E.P., and David A. Pierce (1970), Distribution of residual autocorrelations in autoregressive-integrated moving average time series models, *Journal of the American Statistical Association*, **65**, 1509 - 1526.

Box and Tidwell

Box, G.E.P., and P.W. Tidwell (1962), Transformation of the independent variables, *Technometrics*, **4**, 531 - 550.

Boyette

Boyette, James M. (1979), Random RC tables with given row and column totals, *Applied Statistics*, **28**, 329 - 332.

Bradley

Bradley, J.V. (1968), *Distribution-Free Statistical Tests*, Prentice-Hall, New Jersey.

Bradley, J.V. (1968), *Distribution-Free Statistical Inference*, Prentice-Hall, New Jersey.

Breslow

Breslow, N.E. (1974), Covariance analysis of censored survival data, *Biometrics*, **30**, 89 - 99.

Brillinger

Brillinger, David R. (1981), *Time Series: Data Analysis and Theory*, expanded ed., Holden-Day, San Francisco.

Bross

Bross, I. (1950), Fiducial intervals for variance components, *Biometrics*, **6**, 136-144.

Brown

Brown, Morton B. (1983), BMDP4F, two-way and multiway frequency tables measures of association and the log-linear model (complete and incomplete tables), in *BMDP Statistical Software*, *1983 Printing with Additions*, (edited by W. J. Dixon), University of California Press, Berkeley.

Brown and Benedetti

Brown, Morton B, and Jacqualine K. Benedetti (1977), Sampling behavior and tests for correlation in two-way contingency tables, *Journal of the American Statistical Association*, **42**, 309-315.

Brown and Fuchs

Brown, Morton B., and C. Fuchs (1983), On maximum likelihood estimation in sparse contingency tables, *Computational Statistics and Data Analysis*, **1**, 3-15.

Bryson and Johnson

Bryson, Maurice C. and Mark E. Johnson (1981), The incidence of monotone likelihood in the Cox model, *Technometrics*, **23**, 381 - 384.

Butler and Paolella

Butler, Ronald W., and Marc S. Paolella (1999) *Calculating the Density and Distribution Function for the Singly and Doubly Noncentral F*, Paolella.pdf., Institute of Statistics and Econometrics, Christian Albrechts University at Kiel, Germany, Working Paper 120/99

Cantor

Cantor, Alan B. (1979), A computer algorithm for testing significance in M K contingency tables, *Proceedings of the Statistical Computing Section, American Statistical Association*, Washington, D.C., 220-221.

Carroll and Chang

Carroll, J.D., and J.J. Chang (1970), Analysis of individual differences in multidimensional scaling via an *n*-way generalization of "Eckart-Young" decomposition, *Psychometrika*, **35**, 283 - 319.

Chambers et al.

Chambers, J.M., C.L. Mallows, and B.W. Stuck (1976), A method for simulating stable random variates, *Journal of the American Statistical Association*, **71**, 340 - 344.

Chambers, John M., William S. Cleveland, Beat Kleiner, and Paul A. Tukey (1983), *Graphical Methods for Data Analysis*, Wadsworth, Belmont, Calif.

Chatfield

Chatfield, C. (1980), *The Analysis of Time Series: An Introduction*, 2d ed., Chapman and Hall, London.

Chiang

Chiang, Chin Long (1968), *Introduction to Stochastic Processes in Statistics*, John Wiley & Sons, New York.

Chiang, Chin Long (1972), On constructing current life tables, *Journal of the American Statistical Association*, **67**, 538 - 541.

Chen and Liu

Chen, C. and Liu, L., *Joint Estimation of Model Parameters and Outlier Effects in Time Series*, Journal of the American Statistical Association, Vol. 88, No.421, March 1993.

Cheng

Cheng, R.C.H. (1978), Generating beta variates with nonintegral shape parameters, *Communications of the ACM*, **21**, 317 - 322.

Christensen

Christensen, Ronald (1989), Lack-of-fit tests based on near or exact replicates, *Annals of Statistics*, **17**, 673 - 683.

Clarke

Clarke, M.R.B. (1982), The Gauss-Jordan sweep operator with detection of collinearity, *Applied Statistics*, **31**, 166 - 168.

Clarkson

Clarkson, Douglas B. (1988a), Remark on Algorithm AS 211: The F-G diagonalization algorithm, *Applied Statistics*, **38**, 147 - 151.

Clarkson, Douglas B. (1988b), A least-squares version of AS 211: The F-G diagonalization algorithm, *Applied Statistics*, **38**, 317-321.

Clarkson and Fan

Clarkson, Douglas B. and Yuan-An Fan (1989), *Some improvements to the network algorithm for exact probabilities in contingency tables*, IMSL Technical Report 8903, IMSL, Houston.

Clarkson and Gentle

Clarkson, Douglas B. and James E. Gentle (1986), Methods for multidimensional scaling, in *Computer Science and Statistics*, *Proceedings of the Seventeenth Symposium on the Interface*, (D.M. Allen, editor), North-Holland, Amsterdam, 185 - 192.

Clarkson and Jennrich

Clarkson, Douglas B. and Robert I. Jennrich (1988), Computing extended maximum likelihood estimates for linear parameter models, IMSL *Technical Report* 8804, IMSL, Houston.

Clarkson, Douglas B. and Robert I. Jennrich (1988), Quartic rotation criteria and algorithms, *Psychometrika*, **53**, 251 - 259.

Clarkson, Douglas B. and Robert I. Jennrich (1991), Computing extended maximum likelihood estimates for linear parameter models, submitted to *Journal of the Royal Statistical Society, Series B*, **53**, 417-426.

Cochran

Cochran, William G. (1977), *Sampling Techniques*, 3rd ed., John Wiley & Sons, New York.

Conover

Conover, W.J. (1980), *Practical Nonparametric Statistics*, 2d ed., John Wiley & Sons, New York.

Conover and Iman

Conover, W.J., and Ronald L. Iman (1983), *Introduction to Modern Business Statistics*, John Wiley & Sons, New York.

Cook and Weisberg

Cook, R. Dennis, and Sanford Weisberg (1982), *Residuals and Influence in Regression*, Chapman and Hall, New York.

Cooper

Cooper, B.E. (1968), Algorithm AS4, An auxiliary function for distribution integrals, *(Applied Statistics*), **17**, 190 - 192.

Coveyou and MacPherson

Coveyou, R.R., and R.D. MacPherson (1967), Fourier analysis of uniform random number generators, *Journal of the ACM*, **14**, 100-119.

Cox

Cox, David R. (1970), *The Analysis of Binary Data*, Methuen, London.

Cox, D.R. (1972), Regression models and life tables (with discussion), *Journal of the Royal Statistical Society*, Series B, *Methodology*, **34**, 187 - 220.

Cox and Lewis

Cox, D.R., and P.A.W. Lewis (1966), *The Statistical Analysis of Series of Events*, Methuen, London.

Cox and Oakes

Cox, D.R., and D. Oakes (1984), *Analysis of Survival Data*, Chapman and Hall, London.

Cox and Stuart

Cox, D.R., and A. Stuart (1955), Some quick sign tests for trend in location and dispersion, *Biometrika*, **42**, 80-95.

Craddock

Craddock, J.M. (1969), *Statistics in the Computer Age*, American Elsevier, New York.

Crawford and Ferguson

Crawford, C.B. and G.A. Ferguson (1970), A general rotation criteria and its use in orthogonal rotation, *Psychometrika*, **35**, 321 - 332.

D'Agostino and Stevens

D'Agostino, Ralph B. and Michael A. Stevens (1986) *Goodness-of-Fit Techniques*, Marcel Dekker, New York.

Dahlquist and Bjorck

Dahlquist, Germund, and Ake Bjorck (1974), *Numerical Methods*, translated by Ned Anderson, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Dallal and Wilkinson

Dallal, Gerald E. and Leland Wilkinson (1986), An analytic approximation to the distribution of Lilliefor's test statistic for normality, *The American Statistician*, **40**, 294-296.

Davison

Davison, Mark L. (1983), *Multidimensional Scaling*, John Wiley & Sons, New York.

Davis et al.

Davis, Richard A., Thomas C.M. Lee, and G. A. Rodriquez-Yam (2006), *Structural Break Estimation for Nonstationary Time Series Models*, Journal of the American Statistical Association, Volume **101**, No. **473**, 223-239.

De Jong

De Jong, Sijmen., (1993), "SIMPLS: an alternative approach to partial least squares regression." *Chemometrics and Intelligent Laboratory Systems*. Vol. 18, 251-263.

De Leeuw and Pruzansky

De Leeuw, Jan and Sandra Pruzansky (1978), A new computational method to fit the weighted Euclidean distance model, *Psychometrika*, **43**, 479 - 490.

Deming and Stephan

Deming, W.E., and F.F. Stephan (1940), On the least-squares adjustments of a sampled frequency table when the expected marginal totals are known, *Annals of Mathematical Statistics*, **11**, 427-444.

Dempster, Nan, and Rubin

Dempster, Arthur P., Nan, Laird, and Donald B. Rubin (1977), Maximum likelihood from incomplete data via the EM algorithm (with discussion), *Journal of the Royal Statistical Society*, Serie B, **39**, 1 - 38.

Dennis and Schnabel

Dennis, John E., Jr., and Robert B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

Dewey

Dewey, Michael E. (1984), A remark on Algorithm AS 169: An improved algorithm for scatter plots, *Applied Statistics*, **33**, 370 - 372.

Draper and Smith

Draper, N.R., and H. Smith (1981), *Applied Regression Analysis*, 2d ed., John Wiley & Sons, New York.

Durbin

Durbin, J. (1960), The fitting of time series models, *Revue Institute Internationale de Statistics*, **28**, 233 - 243.

Efroymson

Efroymson, M.A. (1960), Multiple regression analysis, in *Mathematical Methods for Digital Computers*, Volume 1, (edited by A. Ralston and H. Wilf), John Wiley & Sons, New York, 191-203.

Ekblom

Ekblom, Hakan (1973), Calculation of linear best L_p-approximations, *BIT*, **13**, 292-300.

Ekblom, Hakan (1987), The *L*1-estimate as limiting case of an L_p or Huber-estimate, in *Statistical Data Analysis Based on the L1-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 109-116.

Elandt-Johnson and Johnson

Elandt-Johnson, Regina C., and Norman L. Johnson (1980), *Survival Models and Data Analysis*, John Wiley & Sons, New York, 172—173.

Emerson and Hoaglin

Emerson, John D., and David C. Hoaglin (1983), Analysis of two-way tables by medians, in *Understanding Robust and Exploratory Data Analysis* (edited by David C. Hoaglin, Frederick Mosteller, and John W. Tukey), John Wiley & Sons, New York, 166 - 210.

Emmett

Emmett W.G. (1949), Factor analysis by Lawley's method of maximum likelihood, *British Journal of Psychology*, **2**, 90-97.

Fisher

Fisher, R.A. (1936), The use of multiple measurements in taxonomic problems, *The Annals of Eugenics*, **7**, 179-188.

Fishman

Fishman, George S. (1978), *Principles of Discrete Event Simulation*, John Wiley & Sons, New York.

Fishman et al.

Fishman, George F., and Louis R. Moore, III (1982), A statistical evaluation of multiplicative random number generators with modulus 2311, Journal of the American Statistical Association, **77**, 129-136.

Fishman, George F., and Louis R. Moore, III (1986), An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$, *SIAM Journal on Scientific and Statistical Computing*, **7**, 24-45.

Flury

Flury, Bernard H. (1984), Common principal components in *k* groups, *Journal of the American Statistical Association*, **79**, 892-898.

Flury, Bernard H. (1988), *Common Principal Components & Related Multivariate Models*, John Wiley & Sons, New York.

Flury and Constantine

Flury, Bernard H. and Gregory Constantine (1985), The F-G diagonalization algorithm, *Applied Statistics*, **35**, 177-183.

Flury and Gautschi

Flury, Bernard H. and Walter Gautschi (1986), An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form, *SIAM Journal of Scientific and Statistical Computing*, **7**, 169-185.

Forsythe

Forsythe, G.E. (1957), Generation and use of orthogonal polynomials for fitting data with a digital computer, *SIAM Journal on Applied Mathematics*, **5**, 74-88.

Forthofer and Koch

Forthofer, Ronald N., and Gary G. Koch (1973), An analysis of compounded functions of categorical data, *Biometrics*, **29**, 143-157.

Fox, Hall, and Schryer

Fox, P.A., A.D. Hall, and N.L. Schryer (1978), The PORT mathematical subroutine library, *ACM Transactions on Mathematical Software*, **4**, 104-126.

Frane

Frane, James W. (1977), A note on checking tolerance in matrix inversion and regression, *Technometrics*, **19**, 513-514.

Frank and Friedman

Frank, Ildiko E., and Jerome H. Friedman. (1993) A Statistical View of Some Chemometrics Regression Tools. Technometrics, Vol. 35, No. 2, 109-135.

Freeman and Halton

Freeman, G.H., and J.H. Halton (1951), Note on the exact treatment of contingency, goodness of fit, and other problems of significance, *Biometrika*, **38**, 141-149.

Friedman, Bentley, and Finkel

Friedman, Jerome H., Jon Louis Bentley, and Raphael Ari Finkel (1977), An algorithm for finding best matches in logarithmic expected time, *ACM Transactions on Mathematical Software*, **3**, 209-226.

Fuller

Fuller, Wayne A. (1976), *Introduction to Statistical Time Series*, John Wiley & Sons, New York.

Furnival and Wilson

Furnival, G.M., and R.W. Wilson, Jr. (1974), *Regressions by leaps and bounds*, *Technometrics*, **16**, 499-511.

Fushimi

Fushimi, Masanori (1990), Random number generation with the recursion $X_t = X_{t-3p} \oplus X_{t-3q}$, *Journal of Computational and Applied Mathematics*, **31**, 105-118.

Gentle

Gentle, James E. (1981), Portability considerations for random number generators, in *Computer Science and Statistics: The Interface*, (edited by William F. Eddy), SpringerVerlag, New York, 158-161.

Gentle, James E. (1990), Computer implementation of random number generators, *Journal of Computational and Applied Mathematics*, **31**, 119-125.

Gentleman

Gentleman, W. Morven (1974), Basic procedures for large, sparse or weighted linear least squares problems, *Applied Statistics*, **23**, 448-454.

Gibbons

Gibbons, J.D. (1971), *Nonparametric Statistical Inference*, McGraw-Hill, New York.

Giles and Feng

Giles, David E. and Hui Feng. (2009). "Bias-Corrected Maximum Likelihood Estimation of the Parameters of the Generalized Pareto Distribution." Econometrics Working Paper EWP0902, Department of Economics, University of Victoria.

Girshick

Girshick, M.A. (1939), On the sampling theory of roots of determinantal equations, *Annals of Mathematical Statistics*, **10**, 203 - 224.

Golub

Golub, Gene H. (1969), Matrix computations and statistical calculations, in *Statistical Computation*, (edited by Roy C. Milton and John A. Nelder), Academic Press, New York. 365 - 398.

Golub and Van Loan

Golub, Gene H. and Charles F. Van Loan (1983), *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland.

Gonin and Money

Gonin, Rene, and Arthur H. Money (1989), *Nonlinear L_p-Norm Estimation*, Marcel Dekker, New York.

Goodnight

Goodnight, James H. (1979), A tutorial on the SWEEP operator, *The American Statistician*, **33**, 149-158.

Granger and Newbold

Granger, C.W.J., and Paul Newbold (1977), *Forecasting Economic Time* Series, Academic Press, Orlando, Florida.

Graybill

Graybill, Franklin A. (1976), *Theory and Application of the Linear Model*, Duxbury Press, North Scituate, Mass.

Griffin and Redish

Griffin, R., and K.A. Redish (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **13**, 54.

Grizzle, Starmer, and Koch

Grizzle, J.E., C.F. Starmer, and G.G. Koch, (1969), Analysis of categorical data by linear models, *Biometrics*, **28**, 489-504.

Gross and Clark

Gross, Alan J., and Virginia A. Clark (1975), *Survival Distributions: Reliability Applications in the Biomedical Sciences*, John Wiley & Sons, New York.

Gruenberger and Mark

Gruenberger, F., and A.M. Mark (1951), The d^2 test of random digits, *Mathematical Tables and Other Aids in Computation*, **5**, 109-110.

Guerra et al.

Guerra, Victor O., Richard A. Tapia, and James R. Thompson (1976), A random number generator for continuous random variables based on an interpolation procedure of Akima, in *Proceedings of the Ninth Interface Symposium on Computer Science and Statistics*, (edited by David C. Hoaglin and Roy E. Welsch), Prindle, Weber & Schmidt, Boston, 228-230.

Haberman

Haberman, S.J. (1972), Log-linear fit for contingency tables, *Applied Statistics*, **21**, 218-225.

Haldane

Haldane, J.B.S. (1939), The mean and variance of X^2 when used as a test of homogeneity, when expectations are small, *Biometrika*, **31**, 346.

Hancock

Hancock, T.W. (1975), Remark on Algorithm 434: Exact probabilities for $R \times C$ contingency tables, *Communications of the ACM*, **18**, 117-119.

Hand

Hand, D.J. (1981), *Discrimination and Classification*, John Wiley & Sons, New York.

Harman

Harman, Harry H. (1976), *Modern Factor Analysis*, 3rd. ed. revised, University of Chicago Press, Chicago.

Harris and Kaiser

Harris, C., and H. Kaiser (1964), Oblique factor analysis solutions by orthogonal transformations, *Psychometrika*, **29**, 347 - 362.

Hart, et al.

Hart, John F., E.W. Cheney, Charles L. Lawson, Hans J. Maehly, Charles K. Mesztenyi, John R. Rice, Henry G. Thacher, Jr., and Christoph Witzgall (1968), *Computer Approximations*, John Wiley & Sons, New York.

Hartigan

Hartigan, John A. (1975), *Clustering Algorithms*, John Wiley & Sons, New York.

Hartigan and Wong

Hartigan, J.A., and M.A. Wong (1979), Algorithm AS 136: A *K*-means clustering algorithm, *Applied Statistics*, **28**, 100-108.

Harvey

Harvey, A.C. (1981a), *The Econometric Analysis of Time Series*, Philip Allen Publishers, Deddington, England.

Harvey, A.C. (1981b), *Time Series Models*, John Wiley & Sons, New York.

Hastie, Tibshirani, and Friedman

Hastie T., R. Tibshirani, and J. Friedman (2001) *The Elements of Statistical Learning*, 80-82. New York: Springer.

Hayter

Hayter, Anthony J. (1984), A proof of the conjecture that the Tukey-Kramer multiple comparisons procedure is conservative, *Annals of Statistics*, **12**, 61-75.

Heiberger

Heiberger, Richard M. (1978), Generation of random orthogonal matrices, *Applied Statistics*, **27**, 199-206.

Hemmerle

Hemmerle, William J. (1967), *Statistical Computations on a Digital Computer*, Blaisdell Publishing Company, Waltham, Mass.

Hendrickson and White

Hendrickson, A., and P. White (1964), PROMAX: A quick method for rotation to oblique simple structure, *British Journal of Statistical Psychology*, **17**, 65 - 70.

Herraman

Herraman, C. (1968), Sums of squares and products matrix, *Applied Statistics*, **17**, 289-292.

Hill

Hill, G.W. (1970), Student's *t*-distribution, *Communications of the ACM*, **13**, 617 - 620.

Hinkley

Hinkley, David (1977), On quick choice of power transformation, *Applied Statistics*, **26**, 67-69.

Hoaglin

Hoaglin, David C. (1983), Letter values: A set of selected order statistics, in *Understanding Robust and Exploratory Data Analysis* (edited by David C. Hoaglin, Frederick Mosteller, and John W. Tukey), John Wiley & Sons, New York, 33 - 57.

Hoaglin et al.

Hoaglin, David C., Frederick Mosteller, and John W. Tukey (editors) (1983), *Understanding Robust and Exploratory Data Analysis*, John Wiley & Sons, New York.

Hoaglin and Welsch

Hoaglin, D.C., and R. Welsch (1978), The hat matrix in regression and ANOVA, *American Statistician*, **32**, 17-22.

Hocking

Hocking, R.R. (1972), Criteria for selection of a subset regression: Which one should be used?, *Technometrics*, **14**, 967-970.

Hocking, R.R. (1973), A discussion of the two-way mixed model, *The American Statistician*, **27**, 148-152.

Hocking, R.R. (1985), *The Analysis of Linear Models*, Brooks/Cole Publishing Company, Monterey, California.

Hosking, et al.

Hosking, J.R.M., Wallis, J.R., and E.F. Wood. (1985). "Estimation of the Generalized Extreme Value Distribution by the Method of Probability-Weighted Moments." *Technometrics*. Vol 27. No. 3. pp 251-261.

Hosking and Wallis

Hosking, J.R.M. and J.R. Wallis. (1987). "Parameter and Quantile Estimation for the Generalized Pareto Distribution." *Technometrics*. Vol 29. No. 3. pp 339-349.

Huber

Huber, Peter J. (1977), Robust covariances, in *Statistical Decision Theory and Related Topics*, S.S. Gupta and D.S. Moore (editors), Academic Press, New York.

Huber, Peter J. (1981), *Robust Statistics*, John Wiley & Sons, New York.

Hughes and Saw

Hughes, David T., and John G. Saw (1972), Approximating the percentage points of Hotelling's generalized T_0^2 statistic, *Biometrika*, **59**, 224-226.

Hurley and Cattell

Hurley, J., and R. Cattell (1962), The Procrustes program: Producing direct rotation to test a hypothesized factor structure, *Behavioral Science*, **7**, 258 - 262.

IEEE

ANSI/IEEE Std 754-1985 (1985), *IEEE Standard for Binary Floating-Point Arithmetic*, The IEEE, Inc., New York.

Iman and Davenport

Iman, R.L., and J.M. Davenport (1980), Approximations of the critical region of the Friedman statistic, *Communications in Statistics*, **A9(6)**, 571 - 595.

Isogai

Isogai, Takafumi (1983), On measures of multivariate skewness and kurtosis, *Mathematica Japonica*, **28**, 251 - 261.

Jenkins and Watts

Jenkins, Gwilym M., and Donald G. Watts (1968), *Spectral Analysis and Its Applications*, Holden-Day, Oakland, Calif.

Jennrich

Jennrich, Robert I. (1973), Standard errors for obliquely rotated factor loadings, *Psychometrika*, **38**, 593 - 604.

Jennrich and Robinson

Jennrich, R.I., and S.M. Robinson (1969), A Newton-Raphson algorithm for maximum likelihood factor analysis, *Psychometrika*, **34**, 111 - 123.

Jennrich and Sampson

Jennrich, R.I. and P.F. Sampson (1966), Rotation for simple loadings, *Psychometrika*, **31**, 313 - 323.

John (1971)

John, Peter W.M. (1971), *Statistical Design and Analysis of Experiments*, Macmillan Company, New York.

Johnk

Johnk, M.D. (1964), Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen, *Metrika*, **8**, 5 - 15.

Johnson and Kotz

Johnson, Norman L., and Samuel Kotz (1969), *Discrete Distributions*, Houghton Mifflin Company, Boston.

Johnson, Norman L., and Samuel Kotz (1970a), *Continuous Univariate Distributions*-1, John Wiley & Sons, New York.

Johnson, Norman L., and Samuel Kotz (1970b), *Continuous Univariate Distributions*-2, John Wiley & Sons, New York.

Johnson and Welch

Johnson, D.G., and W.J. Welch (1980), The generation of pseudo-random correlation matrices, *Journal of Statistical Computation and Simulation*, **11**, 55 - 69.

Jonckheere

Jonckheere, A.R. (1954), A distribution-free *k*-sample test against ordered alternatives, *Biometrika*, **41**, 133 - 143.

Joreskog

Joreskog, K.G. (1977), Factor analysis by least squares and maximum-likelihood methods, in *Statistical Methods for Digital Computers*, (edited by Kurt Enslein, Anthony Ralston, and Herbert S. Wilf), John Wiley & Sons, New York, 125 - 153.

Kachitvichyanukul

Kachitvichyanukul, Voratas (1982), *Computer generation of Poisson, binomial, and hypergeometric random variates*, Ph.D. dissertation, Purdue University, West Lafayette, Indiana.

Kaiser

Kaiser, H.F. (1958), The varimax criterion for analytic rotation in factor analysis, *Psychometrika*, **23**, 187 - 200.

Kaiser, H.F. (1963), Image analysis, in *Problems in Measuring Change*, (edited by C. Harris), University of Wisconsin Press, Madison, Wisconsin.

Kaiser and Caffrey

Kaiser, H.F., and J. Caffrey (1965), Alpha factor analysis, *Psychometrika*, **30**, 1 - 14.

Kalbfleisch and Prentice

Kalbfleisch, John D., and Ross L. Prentice (1980), *The Statistical Analysis of Failure Time Data*, John Wiley & Sons, New York.

Kalman

Kalman, R. E. (1960), A new approach to linear filtering and prediction problems, *Journal of Basic Engineering*, **82**, 35 - 45.

Kelly

Kelly, L.G. (1967), *Handbook of Numerical Methods and Applications*, Addison-Wesley, Reading, Mass.

Kemp

Kemp, A.W., (1981), Efficient generation of logarithmically distributed pseudo-random variables, *Applied Statistics*, **30**, 249 - 253.

Kempthorne

Kempthorne, Oscar (1975), *The Design and Analysis of Experiments*, Robert E. Krieger Publishing Company, Huntington, New York.

Kendall

Kendall, Maurice G. (1962), *Rank Correlation Methods*, Charles Griffin & Company, 94 - 100.

Kendall, Stuart, and Ord

Kendall, Maurice G., Alan Stuart, and J. Keith Ord (1983), *The Advanced Theory of Statistics*, Volume 3: *Design and Analysis, and Time Series*, 4th ed., Oxford University Press, New York.

Kendall, Maurice G., Alan Stuart, and J. Keith Ord (1987), *The Advanced Theory of Statistics*, Volume 1: Distribution Theory, 5th ed., Oxford University Press, New York.

Kendall and Stuart

Kendall, Maurice G., and Alan Stuart (1979), *The Advanced Theory of Statistics*, Volume 2: *Inference and Relationship*, 4th ed., Oxford University Press, New York.

Kennedy and Gentle

Kennedy, William J., and James E. Gentle (1980), *Statistical Computing*, Marcel Dekker, New York.

Kim and Jennrich

Kim, P.J., and R.I. Jennrich (1973), Tables of the exact sampling distribution of the two sample Kolmogorov-Smirnov criterion D_{mn} ($m < n$), in *Selected Tables in Mathematical Statistics*, Volume 1, (edited by H. L. Harter and D.B. Owen), American Mathematical Society, Providence, Rhode Island.

Kinderman

Kinderman, A.J., and J.G. Ramage (1976), Computer generation of normal random variables, *Journal of the American Statistical Association*, **71**, 893-896.

Kinderman, A.J., J.F. Monahan, and J.G. Ramage (1977), Computer methods for sampling from Student's *t* distribution, *Mathematics of Computation* **31**, 1009-1018.

Kinnucan and Kuki

Kinnucan, P., and H. Kuki (1968), A single precision inverse error function subroutine, Computation Center, University of Chicago. Strecok, Anthony J. (1968), On the calculation of the inverse of the error function, *Mathematics of Computation*, **22**, 144 - 158.

Kirk

Kirk, Roger E. (1982), *Experimental Design: Procedures for the Behavioral Sciences*, 2d. ed., Brooks/Cole Publishing Company, Monterey, Calif.

Kitagawa and Akaike

Kitagawa, G. and Akaike, H. (1978) A procedure for the modeling of non-stationary time series. Ann. Inst. Statist. Math., Vol 30, B, 351-363.

Knuth

Knuth, Donald E. (1973), *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Addison-Wesley Publishing Company, Reading, Mass.

Knuth, Donald E. (1981), *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, 2d ed., Addison-Wesley, Reading, Mass.

Koch, Amara, and Atkinson

Koch, G.G., I.A. Amara, and S.S. Atkinson (1983), Mantel-Haenszel and related methods in analyzing ordinal categorical data with concomitant information, 39th Annual Conference on Applied Statistics, Newark, New Jersey.

Kotz and Johnson

Kotz, Samuel, and Norman L. Johnson (Editors) (1986), *Encyclopedia of Statistical Sciences*, **7**, John Wiley & Sons, New York.

Kronmal and Peterson

Kronmal, Richard A., and Arthur J. Peterson, Jr. (1979), On the alias method for generating random variables from a discrete distribution, *The American Statistician*, **33**, 214-218.

Kruskal

Kruskal, J.B. (1964), Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika*, **29**, 1 - 27.

Kruskal, Young, and Seery

Kruskal J.B., F.W. Young, and J.B. Seery (1977), How to use KYST, a very flexible program to do multidimensional scaling and unfolding, Unpublished manuscript, Bell Telephone Laboratories, Murray Hill, New Jersey.

Kshirsagar

Kshirsagar, Anant M. (1972), *Multivariate Analysis*, Marcel Dekker, New York.

Lachenbruch

Lachenbruch, Peter A. (1975), *Discriminant Analysis*, Hafner Press, London.

Landis, Cooper, Kennedy, and Koch

Landis, J. Richard, Murray M. Cooper, Thomas Kennedy, and Gary G. Koch (1979), A computer program for testing average partial association in three-way contingency tables (PARCAT), *Computer Programs in Biomedicine*, **9**, 223-246.

Landis, Stanish, Freeman, and Koch

Landis, J. Richard, William M. Stanish, Jean L. Freeman, and Gary G. Koch (1976), A computer program for the generalized chi-square analysis of categorical data using weighted least squares (GENCAT), *Computer Programs in Biomedicine*, **6**, 196-231.

Lawless

Lawless, J.F. (1982), *Statistical Models and Methods for Lifetime Data*, John Wiley & Sons, New York.

Lawley and Maxwell

Lawley, D.N., and A.E. Maxwell (1971), *Factor Analysis as a Statistical Method*, 2d ed., Butterworth, London.

Learmonth et al.

Learmonth, G.P., and P.A. W. Lewis (1973a), *Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW73061A*, Naval Postgraduate School, Monterey, Calif.

Learmonth, G. P., and P. A. W. Lewis (1973b), Statistical tests of some widely used and recently proposed uniform random number generators, in *Computer Science and Statistics: 7th Annual Symposium on the Interface*, (edited by William J. Kennedy), Statistical Laboratory, Iowa State University, Ames, Iowa, 163-171.

Lee (1977)

Lee, S. Keith (1977), On the asymptotic variances of $\hat{\mu}$ terms in log-linear models of multidimensional contingency tables, *Journal of the American Statistical Association*, **72**, 412-419.

Lee (1980)

Lee, Elisa T. (1980), *Statistical Methods for Survival Data Analysis*, Lifetime Learning Publications, Belmont, Calif.

Lehmann

Lehmann, E.L. (1975), *Nonparametrics: Statistical Methods Based on Ranks*, Holden-Day, San Francisco.

Levenberg

Levenberg, K. (1944), A method for the solution of certain problems in least squares, *Quarterly of Applied Mathematics*, **2**, 164-168.

Levin and Marascuilo

Levin, J.R., and L.A. Marascuilo (1983), *Multivariate Statistics in the Social Sciences:* A Researcher's Guide, Wadsworth, Inc., California.

Lewis et al.

Lewis, P.A.W., A. S. Goodman, and J.M. Miller (1969), A pseudorandom number generator for the System/360, *IBM Systems Journal*, **8**, 136 - 146.

Lewis, P.A.W., and G.S. Shedler (1979), Simulation of nonhomogeneous Poisson processes by thinning, *Naval Logistics Quarterly*, **26**, 403 - 413.

Lilliefors

Lilliefors, H.W. (1967), On the Kolmogorov-Smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association*, **62**, 534 - 544.

Lilliefors, H.W. (1969), On the Kolmogorov-Smirnov test for the exponential distribution with mean unknown, *Journal of the American Statistical Association*, **64**, 387 - 389.

Lin and Bendel

Lin, Shang P., and Robert B. Bendel (1985), Generation of population correlation matrices with specified eigenvalues, *Applied Statistics*, **34**, 193 - 198.

Longley

Longley, James W. (1967), An appraisal of least-squares programs for the electronic computer from the point of view of the user, *Journal of the American Statistical Association*, **62**, 819-841.

Ljung and Box

Ljung, G.M., and G.E.P. Box (1978), On a measure of lack of fit in time series models, *Biometrika*, **65**, 297 - 303.

Martens and Martens

Martens, Harlad., and Magni Martens, (2000),Modified Jack-knife Estimation of Parameter Uncertainty in Bilinear Modelling by Partial Least Squares Regression (PLSR). *Food Quality and Preference*. **11**, 5-16.

McCormack

McCormack, R.L. (1965), Extended tables of the Wilcoxon matched pair signed rank test, *Journal of the American Statistical Association*, **60**, 96 - 102.

McCullagh and Nelder

McCullagh, P., and J.A. Nelder, (1983), *Generalized Linear Models*, Chapman and Hall, London.

McKean and Schrader

McKean, Joseph W., and Ronald M. Schrader (1987), Least absolute errors analysis of variance, in *Statistical Data Analysis Based on the L1-Norm and Related Methods* (edited by Yadolah Dodge), North-Holland, Amsterdam, 297-305.

McKeon

McKeon, James J. (1974), *F* approximations to the distribution of Hotelling's T_0^2, *Biometrika*, **61**, 381-383.

Maindonald

Maindonald, J.H. (1984), *Statistical Computation*, John Wiley & Sons, New York.

Majumder and Bhattacharjee

Majumder, K. L., and G. P. Bhattacharjee (1973), Algorithm AS 63, The Incomplete Beta Integral, *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 22, No. 3, 409-411, www.jstor.org/stable/2346797.

Mandel

Mandel, J. (1961), Non-additivity in two-way analysis of variance, *Journal of the American Statistical Association*, **69**, 859-866.

Marazzi

Marazzi, Alfio (1985), Robust affine invariant covariances in ROBETH, ROBETH-85 document No. 6, Division de Statistique et Informatique, Institut Universitaire de Medecine Sociale et Preventive, Laussanne.

March

March, D.L. (1972), Algorithm 434: *Exact probabilities for R × C contingency tables*, *Communications of the ACM*, **15**, 991-992.

Mardia et al.

Mardia, K.V. (1970), Measures of multivariate skewness and kurtosis with applications, *Biometrics*, **57**, 519-530.

Mardia, K.V., J.T. Kent, J.M. Bibby (1979), *Multivariate Analysis*, Academic Press, New York.

Mardia and Foster

Mardia, K.V. and K. Foster (1983), Omnibus tests of multinormality based on skewness and kurtosis, *Communications in Statistics A, Theory and Methods*, **12**, 207-221.

Marquardt

Marquardt, D. (1963), An algorithm for least-squares estimation of nonlinear parameters, *SIAM Journal on Applied Mathematics*, **11**, 431-441.

Marsaglia

Marsaglia, George (1964), Generating a variable from the tail of a normal distribution, *Technometrics*, **6**, 101-102.

Marsaglia, G. (1968), Random numbers fall mainly in the planes, *Proceedings of the National Academy of Sciences*, **61**, 25-28.

Marsaglia, G. (1972), The structure of linear congruential sequences, in *Applications of Number Theory to Numerical Analysis*, (edited by S. K. Zaremba), Academic Press, New York, 249-286.

Marsaglia, George (1972), Choosing a point from the surface of a sphere, *The Annals of Mathematical Statistics*, **43**, 645-646.

Marsaglia and Bray

Marsaglia, G. and T.A. Bray (1964), A convenient method for generating normal variables, *SIAM Review*, **6**, 260-264.

Marsaglia et al.

Marsaglia, G., M.D. MacLaren, and T.A. Bray (1964), A fast procedure for generating normal random variables, *Communications of the ACM*, **7**, 4-10.

Martinson and Hamdan

Martinson, E.O., and M.A. Hamdan (1972), Maximum likelihood and some other asymptotically efficient estimators of correlation in two way contingency tables, *Journal of Statistical Computation and Simulation*, **1**, 45-54.

Matsumoto and Nishimure

Makoto Matsumoto and Takuji Nishimura, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, Pages 3 - 30.

McLeod and Bellhouse

McLeod, A.I., and D.R. Bellhouse (1983), A convenient algorithm for drawing a simple random sample, *Applied Statistics*, **32**, 182-184.

Mehta and Patel

Mehta, Cyrus R., and Nitin R. Patel (1983), A network algorithm for performing Fisher's exact test in *r c* contingency tables, *Journal of the American Statistical Association*, **78**, 427-434.

Mehta, C.R. and N.R. Patel (1986a), Algorithm 643: FEXACT: A FORTRAN subroutine for Fisher's exact test on unordered $r \times c$ contingency tables, *ACM Transactions on Mathematical Software*, **12**, 154-161.

Mehta, C.R. and N.R. Patel (1986b), A hybrid algorithm for Fisher's exact test in unordered $r \times c$ contingency tables, *Communication in Statistics, Series A*, **15**, 387-404.

Merle and Spath

Merle, G., and H. Spath (1974), Computational experiences with discrete L_p approximation, *Computing*, **12**, 315-321.

Meyers

Meyers, Raymond H. (1971), *Response Surface Methodology*, Allyn and Bacon, Boston.

Miller

Miller, Rupert G., Jr. (1980), *Simultaneous Statistical Inference*, 2d ed., SpringerVerlag, New York.

Milliken and Johnson

Milliken, George A., and Dallas E. Johnson (1984), *Analysis of Messy Data: Volume 1*, Designed Experiments, Van Nostrand Reinhold, New York.

Moran

Moran, P.A.P. (1947), Some theorems on time series `I`, *Biometrika*, **34**, 281 - 291.

More and Hillstrom

More, J.J., B.S. Garbow, and K. E. Hillstrom (1980), *User Guide for MINPACK-1*, Argonne National Labs Report ANL-80-74, Argonne, Ill.

Morrison

Morrison, Donald F. (1976), *Multivariate Statistical Methods*, 2nd. ed. McGraw-Hill Book Company, New York.

Mosier

Mosier, C. (1939), Determining a simple structure when loadings for certain tests are known, *Psychometrika*, **4**, 149 - 162.

Muller

Muller, M.E. (1959), A note on a method for generating points uniformly on N-dimensional spheres, *Communications of the ACM*, **2**, 19 - 20.

Neter and Wasserman

Neter, John, and William Wasserman (1974), *Applied Linear Statistical Models*, Richard D. Irwin, Homewood, Ill.

Neter, Wasserman, and Kutner

Neter, John, William Wasserman, and Michael H. Kutner (1983), *Applied Linear Regression Models*, Richard D. Irwin, Homewood, Ill.

Noether

Noether, G.E. (1956), Two sequential tests against trend, *Journal of the American Statistical Association*, **51**, 440 - 450.

Null and Sarle

Null, Cynthia H., and Warren S. Sarle (1982), Multidimensional Scaling by Least Squares, in *Proceedings of the Seventh Annual SAS Users Group International Conference*, SAS Institute Inc., Cary, North Carolina.

Owen

Owen, D.B. (1962), *Handbook of Statistical Tables*, Addison-Wesley Publishing Company, Reading, Mass.

Owen, D.B. (1965), A special case of the bivariate non-central *t*-distribution, *Biometrika*, **52**, 437 - 446.

Ozaki and Oda

Ozaki, T and Oda H (1978) Nonlinear time series model identification by Akaike's information criterion. *Information and Systems, Dubuisson eds*, Pergamon Press. 83-91.

Pagano and Halvorsen

Pagano, Marcello, and Katherine Taylor Halvorsen (1981), An algorithm for finding the exact significance levels in $r \times c$ contingency tables, *Journal of the American Statistical Association*, **76**, 931-934.

Park and Miller

Park, Stephen K., and Keith W. Miller (1988), Random number generators: good ones are hard to find, *Communications of the ACM*, **31**, 1192-1201.

Patefield

Patefield, W.M. (1981), An efficient method of generating $R \times C$ tables with given row and column totals, *Applied Statistics*, **30**, 91-97.

Patefield and Tandy

Patefield, W.M. (1981), and Tandy D. (2000) Fast and Accurate Calculation of Owen's T-Function, *J. Statistical Software*, **5**, Issue 5.

Peixoto

Peixoto, Julio L. (1986), Testable hypotheses in singular fixed linear models, Communications in Statistics: *Theory and Methods*, **15**, 1957-1973.

Peto

Peto, R. (1973), Experimental survival curves for interval-censored data, *Applied Statistics*, **22**, 86 - 91.

Petro

Petro, R. (1970), Remark on Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **13**, 624.

Pike

Pike, M.C. (1966), A method of analysis of a certain class of experiments in carcinogenesis, *Biometrics*, **22**, 1 - 39.

Pillai

Pillai, K.C.S. (1985), Pillai's trace, in *Encyclopedia of Statistical Sciences*, *Volume 6*, (edited by Samuel Kotz and Norman L. Johnson), John Wiley & Sons, New York, 725-729.

Pregibon

Pregibon, Daryl (1981), Logistic regression diagnostics, *The Annals of Statistics*, **9**, 705 - 724.

Priestley

Priestley, M.B. (1981), *Spectral Analysis and Time Series*, Volumes 1 and 2, Academic Press, New York.

Prentice

Prentice, Ross L. (1976), A generalization of the probit and logit methods for dose response curves, *Biometrics*, **32**, 761 - 768.

Ramsey

Ramsey, James O. (1977), Maximum likelihood estimation in multidimensional scaling, *Psychometrika*, **42**, 241 - 266.

Ramsey, J.O. (1978), Confidence regions for multidimensional scaling analysis, *Psychometrika*, **43**, 145 - 160.

Ramsey, J.O. (1983), *Multiscale II Manual*, Unpublished manuscript, McGill University, Montreal, Quebec, Canada.

Rao

Rao, C. Radhakrishna (1973), *Linear Statistical Inference and Its Applications*, 2d ed., John Wiley & Sons, New York.

Robinson

Robinson, Enders A. (1967), *Multichannel Time Series Analysis with Digital Computer Programs*, Holden-Day, San Francisco.

Romesburg and Marshall

Romesburg, C., and K. Marshall (1974), *LIFE: A computer program for stochastic life table analysis*, US/IBP Desert Research Memorandum 74-68, Utah State University, Logan, Utah.

Rosipal and Kramer

Rosipal, R., Nicole Krämer. N. (2006) "Overview and Recent Advances in Partial Least Squares." *Subspace, Latent Strucure and Feature Selection*, 3940: 34-51.

Royston

Royston, J.P. (1982a), An extension of Shapiro and Wilk's *W* test for normality to large samples, *Applied Statistics*, **31**, 115-124.

Royston, J.P. (1982b), The *W* test for normality, *Applied Statistics*, **31**, 176-180.

Royston, J.P. (1982c), Expected normal order statistics (exact and approximate), *Applied Statistics*, **31**, 161-165.

Sallas

Sallas, William M. (1988), Some Remarks on Algorithm AS 164. Least squares subject to linear constraints, *Applied Statistics*, **37**, 484-489.

Sallas, William M. (1990), An algorithm for an L_p norm fit of a multiple linear regression model, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 131-136.

Sallas and Harville

Sallas, William M., and David A. Harville (1981), Best linear recursive estimation for mixed linear models, *Journal of American Statistical Association*, **76**, 860-869.

Sallas, William M., and David A. Harville (1988), Noninformative priors and restricted maximum likelihood estimation in the Kalman filter, in *Bayesian Analysis of Time Series and Dynamic Models* (edited by James C. Spall), Marcel Dekker, New York, 477 - 508.

Sallas and Lionti

Sallas, William M. and Abby M. Lionti (1988), Some useful computing formulas for the nonfull rank linear model with linear equality restrictions, IMSL Technical Report 8805, IMSL, Houston.

Satterthwaite

Satterthwaite, F.E. (1946), An approximate distribution of estimates of variance components, *Biometrics Bulletin*, **2**, 110-114.

Savage

Savage, I. Richard (1956), Contributions to the theory of rank order statistics|the twosample case, *Annals of Mathematical Statistics*, **27**, 590-615.

Scheffe

Scheffe, Henry (1959), *The Analysis of Variance*, John Wiley & Sons, New York.

Schiffman, Reynolds, and Young

Schiffman, Susan S., M. Lance Reynolds, and Forrest W. Young (1981), *Introduction to Multidimensional Scaling: Theory*, *Methods*, *and Applications*, Academic Press, New York.

Schmeiser et al.

Schmeiser, Bruce W., and A.J.G. Babu (1980), Beta variate generation via exponential majorizing functions, *Operations Research*, **28**, 917-926.

Schmeiser, Bruce W., and Ram Lal (1980), Squeeze methods for generating gamma variates, *Journal of the American Statistical Association*, **75**, 679-682.

Schmeiser, Bruce, and Voratas Kachitvichyanukul (1981), *Poisson Random Variate Generation*, Research Memorandum 81-4, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

Schmeiser, Bruce (1983), Recent advances in generating observations from discrete random variates, in *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, (edited by James E. Gentle), North-Holland Publishing Company, Amsterdam, 154-160.

Schoneman

Schoneman, P.H. (1966), A generalized solution of the orthogonal Procrustes problem, *Psychometrika*, **31**, 1 - 10.

Scott

Scott, David W. (1976), Nonparametric probability density estimation by optimization theoretic techniques, *Technical Report* **476** 131-1, Rice University, Houston, Texas.

Scott, Tapia, and Thompson

Scott, D.W., R.W. Tapia, and J.R. Thompson (1980), Nonparametric probability density estimation by discrete penalized-likelihood criteria, *The Annals of Statistics*, **8**, 820 - 832.

Searle

Searle, S.R. (1971), *Linear Models*, John Wiley & Sons, New York.

Seber

Seber, G.A.F. (1984), *Multivariate Observations*, John Wiley & Sons, New York.

Shampine

Shampine, L.F. (1975), Discrete least-squares polynomial fits, *Communications of the ACM*, **18**, 179-180.

Siegel

Siegel, Sidney (1956), *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, New York.

Silverman

Silverman, Bernard W. (1982), Kernel density estimation using the fast Fourier transform, *Applied Statistics*, **31**, 93 - 99.

Silverman, Bernard W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.

Singleton

Singleton, R.C. (1969), Algorithm 347: An efficient algorithm for sorting with minimal storage, *Communications of the ACM*, **12**, 185-187.

Smirnov

Smirnov, N.V. (1939), Estimate of deviation between empirical distribution functions in two independent samples (in Russian), *Bulletin of Moscow University*, **2**, 3-16.

Snedecor and Cochran

Snedecor, George W., and William G. Cochran (1967), *Statistical Methods*, 6th. ed., Iowa State University Press, Ames, Iowa.

Sposito

Sposito, Vincent A. (1989), Some properties of L_p-estimators, in *Robust Regression*: *Analysis and Applications* (edited by Kenneth D. Lawrence and Jeffrey L. Arthur), Marcel Dekker, New York, 23-58.

Spurrier and Isham

Spurrier, John D. and Steven P. Isham (1985), Exact simultaneous confidence intervals for pairwise comparisons of three normal means, *Journal of the American Statistical Association*, **80**, 438-442.

Stablein, Carter, and Novak

Stablein, D.M, W.H. Carter, and J.W. Novak (1981), Analysis of survival data with nonproportional hazard functions, *Controlled Clinical Trials*, **2**, 149 - 159.

Stahel

Stahel, W. (1981), Robuste Schatzugen: Infinitesimale Opimalitat und Schatzugen von Kovarianzmatrizen, Dissertation no. 6881, ETH, Zurich.

Stephens

Stephens, M.A. (1974), EDF statistics for goodness of fit and some comparisons, *Journal of the American Statistical Association*, **69**, 730-737.

Stephens, M.A. (1986): Tests based on EDF statistics. In: D'Agostino, R.B. and Stephens, M.A., eds.: Goodness-of-Fit Techniques. Marcel Dekker, New York.

Stirling

Stirling, W.D. (1981), Algorithm AS 164. Least squares subject to linear constraints, *Applied Statistics*, **30**, 204-212 (see correction, page 357).

Stirling, W.D. (1981), Algorithm AS 169: An improved algorithm for scatter plots, *Applied Statistics*, **30**, 345 - 349.

Stoline

Stoline, Michael R. (1981), The status of multiple comparisons: simultaneous estimation of all pairwise comparisons in one-way ANOVA designs, *The American Statistician*, **35**, 134-141.

Swan

Swan, A.V. (1969a), Computing maximum-likelihood estimates for parameters of the normal distribution from grouped and censored data, *Applied Statistics*, **18**, 65-69.

Swan, A.V. (1969b), Maximum likelihood estimation from grouped and censored normal data, *Applied Statistics*, **18**, 110-114.

Takane and Carroll

Takane, Yoshio, and J. Douglas Carroll (1981), Nonmetric maximum likelihood multidimensional scaling from directional ranking of similarities, *Psychometrika*, **46**, 389 - 405.

Takane, Young, and De Leeuw

Takane, Y., F.W. Young, and J. De Leeuw (1977), Nonmetric individual differences multidimensional scaling: An alternating least-squares method with optimal scaling features, *Psychometrika*, **42**, 7 - 67.

Tanner

Tanner, Martin A. (1983), A note on the variable kernel estimator of the hazard function from censored data, *Annals of Statistics*, **11**, 994 - 998.

Tanner and Thisted

Tanner, Martin A., and Ronald A. Thisted (1982), Generation of random orthogonal matrices, *Applied Statistics*, **31**, 190-192.

Tanner and Wong

Tanner, Martin A., and Wing H. Wong (1983), The estimation of the hazard function from randomly censored data by the kernel method, *Annals of Statistics*, **11**, 989 - 993.

Tanner, Martin A., and Wing H. Wong (1984), Data-based nonparametric estimation of the hazard function with applications to model diagnostics and exploratory analysis, *Journal of the American Statistical Association*, **79**, 123 - 456.

Tapia

Tapia, R.A. (1974), A stable approach to Newton's method for general mathematical programming problems in R^n, *Journal of Optimization Theory and Applications*, **14**, 453 - 476.

Tapia and Thompson

Tapia, Richard A., and James R. Thompson (1978), *Nonparametric Probability Density Estimation*, Johns Hopkins University Press, Baltimore.

Tatsuoka

Tatsuoka, Maurice M. (1971), *Multivariate Analysis: Techniques for Educational and Psychological Research*, John Wiley & Sons, New York.

Taylor and Thompson

Taylor, Malcolm S., and James R. Thompson (1986), Data based random number generation for a multivariate distribution via stochastic simulation, *Computational Statistics & Data Analysis*, **4**, 93-101.

Thompson

Thompson, James R, (1989), *Empirical Model Building*, John Wiley & Sons, New York.

TIMSAC

TIMSAC, *TIMe Series Analysis and Control* (TIMSAC-72, TIMSAC-74, TIMSAC-78, TIMSAC-84). Developed by the Institute of Statistical Mathematics, Japan.

Tucker and Lewis

Tucker, Ledyard, and Charles Lewis (1973), A reliability coefficient for maximum likelihood factor analysis, *Psychometrika*, **38**, 1 - 10.

Tukey

Tukey, J.W. (1949), One degree of freedom for nonadditivity, *Biometrics*, **5**, 232.

Tukey, John W. (1962), The future of data analysis, *Annals of Mathematical Statistics*, **33**, 1-67.

Tukey, John W. (1977), *Exploratory Data Analysis*, Addison-Wesley Publishing Company, Reading, Mass.

Turnbull

Turnbull, Bruce W. (1976), The empirical distribution function with arbitrary grouped, censored, and truncated data, *Journal of the Royal Statistical Society*, Series B: *Methodology*, **38**, 290-295.

Van de Geer

Van de Geer, John P. (1971), *Introduction to Multivariate Analysis for the Social Sciences*, W.H. Freeman and Company, San Francisco.

Velleman and Hoaglin

Velleman, Paul F., and David C. Hoaglin (1981), *Applications, Basics, and Computing of Exploratory Data Analysis*, Duxbury Press, Boston.

Verdooren

Verdooren, L.R. (1963), Extended tables of critical values for Wilcoxon's test statistic, *Biometrika*, **50**, 177-186.

Walker

Walker, A.J. (1974), New fast method for generating discrete random numbers with arbitrary frequency distributions, *Electronics Letters*, **10**, 127-128.

Wallace

Wallace, D.L. (1959), Simplified Beta-approximations to the Kruskal-Wallis H-test, *Journal of the American Statistical Association*, **54**, 225-230.

Weisberg

Weisberg, S. (1985), *Applied Linear Regression*, 2d ed., John Wiley & Sons, New York.

Weeks and Bentler

Weeks, David G., and P.M. Bentler (1982), Restricted multidimensional scaling models for asymmetric proximities, *Psychometrika*, **47**, 201 - 208.

Wilks

Wilks, S.S. (1935), On the independence of *k* sets of normally distributed statistical variables, *Econometrika*, **3**, 309-326.

Williams

Williams, J.S. (1962), A confidence interval for variance component, *Biometrika*, **49**, 278 - 281.

Wold

Wold, Herman, (1966) "Estimation of principal components and related models by iterative least squares." In P.R. Krishnaiah (Editor) *Multivariate Analysis*, 391-420, Academic Press, New York.

Wold, Herman (1985). Partial least squares, 581-591 in Samuel Kotz and Norman L. Johnson, eds., *Encyclopedia of statistical sciences*, Vol. **6**, New York: Wiley.

Wold, Sjostrom, and Eriksson

Wold, S., M. Sjöström, and L. Eriksson (2001), PLS-regression: a basic tool of chemometrics, *Chemometrics and Intelligent Laboratory Systems*. Vol. 58, 109-130.

Woodfield

Woodfield, Terry J. (1990), Some notes on the Ljung-Box portmanteau statistic, *American Statistical Association 1990 Proceedings of the Statistical Computing Section*, 155 - 160.

Young and Lewyckyj

Young, F.W., Y. Takane, and R. Lewyckyj (1978), Three notes on ALSCAL, *Psychometrika*, **43**, 433 - 435.

Young, Forrest W., and Rostyslaw Lewyckyj (1979), *ALSCAL-4 Users Guide*, second edition, Data Analysis and Theory Associates, Chapel Hill, North Carolina.

Appendix C Product Support

Contacting IMSL Support

Users within support warranty may contact IMSL regarding the use of the IMSL Fortran Numerical Library. IMSL Support can consult on the following topics:

- Clarity of documentation
- Possible IMSL-related programming problems
- Choice of IMSL Libraries functions or procedures for a particular problem

Not included in these topics are mathematical/statistical consulting and debugging of your program.

Refer to the following for IMSL Product Support contact information:

https://www.imsl.com/support

The following describes the procedure for consultation with IMSL Support:

- Include your IMSL license number.
- Include the product name and version number.
- Include compiler and operating system version numbers.
- Include the name of the routine for which assistance is needed and a description of the problem.

Index

Q

R

S