Chapter 18
Module Library

Chapter Table of Contents

Chapter 18
Module Library

Overview

IMLMLIB is a library of modules written in the IML language. These modules can be used as though they were built-in functions of IML.

The library contains both functions and subroutines. You can invoke functions in assignment statements or expressions. You can invoke subroutines using CALL or RUN statements. IML automatically loads, resolves, and executes a module when you use it.

Contents of IMLMLIB

The library contains the following modules. Many of them are derived from the examples in the IML sample library. Each module is described in detail at the end of this chapter.

COLVEC	converts a matrix into a column vector
CORR	computes correlation coefficients
GBXWHSKR	draws box and whisker diagram
GPROBCNT	draws probability contours for x-y data
GXYPLOT	draws scatter plots of x-y data
MEDIAN	returns the median of numeric data
QUADREG	performs quadratic regression
QUARTILE	computes quartiles
REGRESS	performs regression analysis
ROWVEC	converts a matrix into a row vector
RSUBSTR	replaces substrings
STANDARD	standardizes numeric data
TABPRT	prints matrices in tabular format

IMLMLIB and the STORAGE library

IML enables you to store and load matrices and modules in your own STORAGE library (refer to the chapter on storage features in *SAS/IML Software: Usage and Reference, Version 6, First Edition*). The IMLMLIB library is different from this STORAGE library. IMLMLIB contains predefined modules that can be loaded only by IML.

The STORAGE library, on the other hand, is under the control of the user. You can store and load both matrices and modules. The STORE, LOAD, REMOVE, and RESET STORAGE commands apply only to the STORAGE library. You cannot store additional modules in IMLMLIB.

You can use the SHOW command to obtain information about the IMLMLIB and STORAGE libraries.

- SHOW OPTIONS displays the current settings of both STORAGE and IMLMLIB libraries and their open status.
- SHOW STORAGE displays the contents of the STORAGE library.
- SHOW IMLMLIB displays the contents of the IMLMLIB library.
- SHOW MODULES displays the names of the modules existing in the current environment. These include modules loaded from the STORAGE library or the IMLMLIB library and modules defined in the current session.

Accessing the IML Source Code

The IMLMLIB library is a catalog residing in the SASHELP directory. The catalog contains one entry of type IMOD for each module. Each entry is a module stored in its compiled form.

The IML source code defining the modules is available in the catalog SASHELP.IML. There is an entry of type SOURCE for each module. You can view the source code in the program editor window under DMS using the COPY command and specifying the four-level name:

SASHELP.IML.*modulename***.SOURCE**

The source code is generally followed by examples of its use.

The source code can be edited for customization or enhancements, and can be included in other IML applications. The modules also illustrate a variety of IML features that can be used to solve more complex problems.

Order of Resolution

SAS/IML resolves functions and subroutines in the following order:

- Functions
  1. IML's built-in functions
  2. User-defined IML modules existing in the current environment
  3. STORAGE library, if open
  4. SAS DATA step functions
  5. IMLMLIB library
- CALL Statement
  1. IML's built-in calls
  2. User-defined IML modules existing in the current enviornment
  3. STORAGE library, if open
  4. SAS DATA step call
  5. IMLMLIB library
- RUN Statement
  1. User-defined IML modules existing in the current environment
  2. STORAGE library, if open
  3. IML's built-in calls
  4. SAS DATA step call
  5. IMLMLIB library

Error Diagnostics

When an error occurs in any IML module, IML pauses in the module and prints error diagnostics with a full traceback that can help in locating the problem. In the case of loaded modules, however, the traceback includes line offsets instead of the absolute SAS LOG line numbers. The offsets can be used to track the problem into the actual source code that originally defined the module. The START statement at the beginning of the module definition is always at offset=1.

Note that offsets apply only to loaded modules. For modules explicitly defined in any given session, absolute line numbers are printed in the traceback.

Modules Reference

COLVEC Function

converts a matrix into a column vector

COLVEC(*matrix***)**

where *matrix* is any $n \times m$ matrix.

The COLVEC function returns an $nm \times 1$ vector. It converts a matrix into a column vector in row-major order. The returned vector has 1 column and nm rows. The first n elements in the vector correspond to the first row of the input matrix, the next n elements correspond to the second row, and so on.

CORR Function

computes correlation coefficients

CORR(*matrix***)**

where *matrix* is any $n \times m$ matrix, m is the number of variables, and n is the number of data points.

The CORR function returns an $m \times m$ matrix of correlation coefficients. It computes the correlation between variables for any multivariate numeric data.

GBXWHSKR Call

draws box and whisker diagrams

RUN GBXWHSKR(*matrix***);**

where *matrix* is any $n \times m$ matrix.

The GBXWHSKR module draws a box-and-whisker plot for univariate numeric data contained in the specified *matrix*. The box outlines the quartile range, and the minimum, median, and maximum points are labeled on the plot. You cannot produce graphics until you invoke the CALL GSTART statement. The plot created by the GBXWHSKR module remains open for further additions until you specify the CALL GCLOSE statement, which terminates the current graphics segment. The module source code can be edited for changes, such as adding viewports, text, or colors.

GPROBCNT Call

draws probability contours

RUN GPROBCNT(x, y <, p >**);**

The inputs to the GPROBCNT subroutine are as follows:

x	is any $n \times m$ matrix of x-values.
y	is a corresponding $n \times m$ matrix of y-values.
p	is an optional probability value matrix.

The GPROBCNT module draws one contour curve corresponding to each value in the matrix p, which must contain entries between zero and one. If you do not specify the matrix p, contours for the probability values of 0.5, 0.8 and 0.9 are drawn. You cannot produce graphics until you invoke the CALL GSTART statement. The contour plot remains open for further additions until you specify the CALL GCLOSE statement, which terminates the current graphics segment. Note that this module cannot be used for general contour plots of three-dimensional data.

GXYPLOT Call

draws a scatter plot of any x-y data

RUN GXYPLOT(x, y**);**

The inputs to the GXYPLOT subroutine are as follows:

x	is any $n \times m$ matrix of x-values.
y	is a corresponding $n \times m$ matrix of y-values.

The GXYPLOT module draws a simple scatter plot of any numeric x-y data. Axes with labeled tickmarks are drawn as well. You cannot produce graphics until you invoke the CALL GSTART statement. The plot remains open for further additions (such as a title and axis labels) until you specify the CALL GCLOSE staement, which terminates the current graphics segment. The module uses the GPOINT, GXAXIS, and GYAXIS calls to plot the points. The module source code can be edited to specify many of the options available for these calls.

MEDIAN Function

returns the median of numeric data

MEDIAN(*matrix***)**

where *matrix* is any $n \times m$ matrix.

The MEDIAN function returns the median value for each column in the *matrix*. It computes the median of univariate numeric data contained in the specified *matrix*. When the number of data points is odd, it returns the middle element from the sorted order. When the number of data points is even, it returns the mean of the middle two elements. If there are missing values, the module returns an error message.

QUADREG Call

performs quadratic response surface regression

RUN QUADREG(*xopt, yopt, type, parms,* x, y**);**

The inputs to the GPROBCNT subroutine are as follows:

xopt	is a returned value containing $m \times 1$ critical factor values.
yopt	is a returned value containing the critical response value.
type	is a returned character string containing the solution type (maximum or minimum).
parms	is a returned value containing the parameter estimates for the quadratic model.
x	is an $n \times m$ factor matrix, where m is the number of factor variables and n is the number of data points.
y	is an $n \times 1$ response vector.

The QUADREG module fits a regression model with a complete quadratic set of regressions across several factors. The estimated model parameters are divided into a vector of linear coefficients and a matrix of quadratic coefficients to obtain critical factor values that optimize the response. It further determines the type of the optima (maximum, minimum, or saddlepoint) by computing the eigenvalues of the estimated parameters.

QUARTILE Function

computes quartiles for any univariate numeric data

QUARTILE(*matrix***)**

where *matrix* is any $n \times m$ matrix.

The QUARTILE function returns a 5×1 column vector for each column in the *matrix*. The column vector contains the minimum, lower quartile, median, upper quartile, and maximum values for the numeric data in the specified *matrix*. If there are missing values, the module returns an error message.

REGRESS Call

performs regression analysis

RUN REGRESS(*x, y, name, <tval>, <l1>, <l2>, <l3>***);**

The inputs to the REGRESS subroutine are as follows:

x	is an $n \times m$ numeric matrix, where m is the number of variables and n is the number of data points.
y	is an $n \times 1$ response vector.
name	is an $m \times 1$ matrix of variable names.
tval	is an optional t-value.
l1, *l2*, *l3*	are optional $1 \times m$ vectors that specify linear combinations of coefficients for hypothesis testing.

The REGRESS module does regression analysis and prints results. The design matrix is given by x, and y is the response vector. The *name* vector identifies each of the variables. If you specify a t-value, the module prints a table of observed and predicted values, residuals, hat diagonal, and confidence limits for the mean and predicted values. If you also specify linear combinations with *l1*, *l2*, and *l3*, the module performs the hypothesis test $\mathbf{H} : l'b = 0$, where b is the vector of parameter estimates.

ROWVEC Function

converts a matrix into a row vector

ROWVEC(*matrix***)**

where *matrix* is any $n \times m$ matrix.

The ROWVEC function returns a $1 \times nm$ vector. The specified *matrix* is converted into a row vector in row-major order. The returned vector has 1 row and nm columns. The first n elements in the vector correpond to the first row of the input matrix, the next n elements correspond to the second row, and so on.

RSUBSTR Function

replaces substrings in each entry of a given matrix

RSUBSTR(x, p, l, r**)**

The inputs to the RSUBSTR subroutine are as follows:

x	is any $m \times n$ character matrix.
p	is an $m \times n$ matrix or a scalar that determines the starting positions for substrings to be replaced.
l	is an $m \times n$ matrix or a scalar that determines the lengths of substrings to be replaced.
r	is an $m \times n$ matrix or a scalar that specifies the replacement strings.

The RSUBSTR function returns an $m \times n$ matrix with substrings replaced. It replaces or substitutes substrings of the input matrix with new strings. If l is zero, the replacement string in r is simply inserted into the input matrix x at the position indicated by p.

For example, the following statements replace the first two characters of each entry in the matrix X with the the corresponding entry in the matrix R:

```
proc iml;
   x = {abc def ghi,jkl mno pqr};
   r = {z y x, w v u};
   p = 1;
   l = 2;
   c=rsubstr(x,p,l,r);
   print x;
   print c;
```

STANDARD Function

standardizes numeric data

STANDARD(*matrix***)**

where *matrix* is any $n \times m$ matrix, n is the number of data points and m is the number of variables.

The STANDARD function returns a standardized $n \times m$ matrix. It standardizes each column of the input matrix, so that the mean of each column is zero and the standard deviation for each column is one.

TABPRT Call

prints matrices in tabular format

RUN TABPRT(*matrix***);**

where *matrix* is any $n \times m$ matrix.

The TABPRT module prints any numeric or character matrix in table format. The regular PRINT command output is often difficult to read, especially for large matrices, where individual rows may wrap around. The module source code can be edited for further cosmetic changes, such as alternative format or field width, or for assigning specific row and column labels.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/IML User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. 846 pp.

SAS/IML User's Guide, Version 8

ISBN 1–58025–553–1

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999