Frequently Used SAS/IML® Functions and Subroutines

Functions That Create and Shape Matrices

Function	Description
DO	Creates an arithmetic sequence
I	Creates an identity matrix
J	Creates a constant matrix of given size
REPEAT	Creates a matrix with repeated values
SHAPE	Reshapes a matrix
T	Transposes a matrix

Matrix Query Functions

Function	Description
LENGTH	Number of characters in each element
LOC	Indices that satisfy a criterion
NCOL	Number of columns
NROW	Number of rows
NLENG	Maximum length of characters
TYPE	Type of matrix: 'C', 'N', or 'U'

Functions for Descriptive Statistics

Function	Description
CUSUM	Cumulative sum of elements
MIN,MAX	Minimum/maximum of elements
RANK	Ranks of elements
SUM	Sum of elements
SSQ	Sum of squares of elements

Functions for Descriptive Statistics (9.22)

Function	Description
CORR	Correlation matrix
COUNTMISS	Number of missing elements
COUNTN	Number of nonmissing elements
COV	Covariance matrix
MEAN	Means of columns
QNTL	Quantiles of columns
VAR	Variances of columns

Functions for Set Operations

Function	Description
SETDIF	Difference between sets
UNION	Elements in union of sets
UNIQUE	Sorted list of unique values
XSECT	Elements in intersection of sets

Mathematical Functions

Function	Description
ABS,MOD,LOG, EXP,SQRT,…	Basic functions
COS,SIN,TAN	Trigonometric functions
ARCOS,ARSIN, ATAN, ATAN2	Inverse trigonometric functions
INT,FLOOR, CEIL,ROUND	Truncation functions

Probability Functions and Subroutines

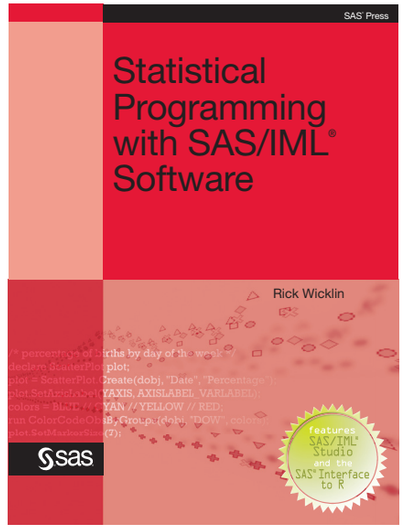
Function	Description
CDF	Cumulative distribution function
PDF	Probability distribution function
QUANTILE	Quantiles from distribution function
NORMAL	Pseudorandom normal (old style)
UNIFORM	Pseudorandom uniform (old style)
RANDSEED	Set pseudorandom seed (new style)
RANDGEN	Generate pseudorandom (new style)

Linear Algebra

Function	Description
DET	Determinant function
EIGEN	Eigenvectors and eigenvalues
INV	Invert matrix (see also GINV)
QR	QR decomposition
ROOT	Cholesky decomposition
SOLVE	Solves a linear system of equations
SVD	Singular value decomposition
TRACE	Sum of diagonal elements of matrix

Statements for Reading and Writing Data

Statement	Description
APPEND	Writes a data set from matrices
CLOSE	Closes a data set
CREATE	Creates a data set for output
READ	Reads data into matrices
USE	Opens a data set for input



This is a companion piece to the following book:

Wicklin, Rick. 2010. *Statistical Programming with SAS/IML® Software*. Cary, NC: SAS Institute Inc.

Copyright © 2010, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-60764-663-1
ISBN 978-1-60764-770-6 (electronic book)
All rights reserved. Produced in the United States of America.

support.sas.com/publishing/authors/wicklin.html

Author's blog: blogs.sas.com/iml

Questions? Comments? Tips?
Join the SAS/IML® and SAS/IML® Studio discussion forum at support.sas.com/forums

sas THE POWER TO KNOW.

SAS Institute Inc. World Headquarters +1 919 677 8000
To contact your local SAS office, please visit: **www.sas.com/offices**

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © 2011 SAS Institute Inc. All rights reserved. 285703_01.11

Tip Sheet
from
Statistical Programming with SAS/IML® Software

SAS/IML® Statements

The SAS/IML® language provides support for matrix-vector programming, including a rich library of functions for statistical programming and matrix computations.

You can call Base SAS® functions from the SAS/IML® language and you can extend SAS/IML® software by writing modules that encapsulate tasks and analyses.

This tip sheet presents the basic syntax of frequently used SAS/IML® statements. It also gives you a variety of examples to type in, study, and modify.

Included are examples of creating matrices, reading and writing data, computing statistics, controlling the logical flow of a program, and handling missing values.

Tip Sheet for the SAS/IML® Language

Create Matrices

```
proc iml;
m = {1 2 3, 4 5 6};/* 2 x 3 matrix        */
m = j(5, 2, 1);    /* 5 x 2 matrix of 1's */
c = 1:5;           /* sequence 1,2,3,4,5  */
c = do(0, 5, 0.5); /* 0, 0.5, 1,…, 5      */
c = repeat(1:2, 3, 2);/* replicate matrix */
pi = constant("pi");
```

Matrix Information

```
n = nrow(m);
p = ncol(m);
/* numeric, character, or undefined?      */
type = type(m); /* 'N', 'C', or 'U'       */
s = {"IML", "Tip", "Sheet"};
nl = nleng(s);  /* length of matrix (5)   */
nc = length(s); /* num characters {3,3,5} */
idx = loc(m>3); /* indices for which m>3  */
```

Pseudorandom Matrices

```
/* old style pseudorandom numbers */
seed = 12345;
v = j(5, 1, seed);/* 5 x 1 seed vector    */
x = ranuni(v);    /* 5 x 1 random uniform */
y = rannor(v);    /* 5 x 1 random normal  */

/* new style pseudorandom numbers */
call randseed(seed);
x = j(5, 1);           /* allocate vector */
call randgen(x, "Normal"); /* fill x      */
```

Subscripts

```
m = {1 2 3, 4 5 6, 7 8 9};
y = m[2, 3];        /* 2nd row, 3rd col    */
r = m[2,  ];        /* second row          */
c = m[ , 3];        /* third column        */

grandmean = m[:];   /* mean                */
colmean = m[:, ];   /* mean of each col    */
rowsum  = m[ ,+];   /* sum of each row     */
colssq  = m[##, ];  /* col sum of squares */
```

Concatenation

```
a = {1,2,3}; b = {4,5,6};
x = a || b;         /* horizontal concat */
y = a // b;         /* vertical concat   */
```

Comparison & Logical Operators

```
x = 1:5;
y = {3 1 2 5 4};
s1 = (x>2);                /* {0 0 1 1 1} */
s2 = (x>y);                /* {0 1 1 0 1} */

/* and (&), or (|), and not (^) */
s3 = (s1 & s2);            /* {0 0 1 0 1} */
s4 = (s1 | s2);            /* {0 1 1 1 1} */
s5 = (^s1);                /* {1 1 0 0 0} */

if x>0 then m="+";    /* all elements > 0 */
if all(x>0) then m="+";/* same expression */
if any(x>y) then m="x>y";         /* true */
```

Set Operations

```
A = 1:4;
B = do(2.5, 4.5, 0.5);
q = unique({2 2 1});    /* unique values  */
u = union(A, B);        /* union          */
inter = xsect(A, B);    /* intersection   */
dif = setdif(A, B);     /* set difference */
```

Matrix Operators

Elementwise

```
m = {1 2 3, 4 5 6};
x = m + (1:3);   /* add to rows           */
y = m - {1,2};   /* subtract from columns */
z = m / (1:3);   /* divide each row       */
w = m # {1,2};   /* multiply each column  */
u = 2*m + 1;     /* scalar mult and add   */
v = m##2;        /* elementwise power     */
```

Matrix

```
m = {7 7, 7 35}; x = {2, 1};
y = m * x;      /* matrix multiplication  */
p = m**2;       /* matrix power (integer) */
t = T(m);       /* transpose (also m`)    */
d = vecdiag(m); /* diagonal: {7, 35}      */
z = solve(m, y);/* solve linear system    */
minv = inv(m);  /* explicit inverse       */
call eigen(v,u,m);/* eigenvalues, vectors */

A = {1 1 1, 1 0 0, 0 1 0, 0 0 1};
call svd(u,q,v,A); /* A = u*diag(q)*v`    */
Ainv = ginv(A);    /* generalized inverse */
rankA=round(trace(Ainv*A));  /* rank of A */
```

Control Statements

```
x = 1;
if x>0 then   y = x;
else          y = -x;
y = choose(x>0, x, -x);    /* elementwise */

if x>0 then do;
   /* more statements */
end;

/* compute Fibonacci sequence */
f = j(1,10,1);  /* allocate result vector */
do i = 3 to ncol(f);
   f[i] = f[i-1] + f[i-2];  /* ith result */
end;

do while(x<5);
   x = x + 1;
end;
```

Read a SAS Data Set

```
/* read vars into vectors of same name */
use sashelp.class;
read all var {sex age weight height};
close sashelp.class;

/* read numeric variables into matrix */
use sashelp.class;
read all var {age weight height} into x;
close sashelp.class;
```

Write a SAS Data Set

```
x = {1,2,3};
y = {4,5,6};
create temp var {x y};
append;
close temp;

x = {1 4 7, 2 5 8, 3 6 9};
varnames = 'x1':'x3';
create temp from x[colname=varnames];
append from x;
close temp;
```

Timing and Performance

```
t0 = time();              /* start timing */
inv = ginv(rannor(j(100,100))); /* stmts  */
t = time() - t0;          /* elapsed time */
```

Handle Missing Values

```
x = {3 1 4 . 5 9 .};
idx = loc(x=.);     /* indices of missing */
if ncol(idx)>0 then     /* are there any? */
   y = x[idx];
```

Descriptive Statistics

```
x = {0,1,2,8,0,3,2,5};
min = min(x);
max = max(x);
sum = sum(x);    /* sum                    */
cus = cusum(x);  /* cumulative sum         */
ssq = ssq(x);    /* sum of squares         */
r = rank(x);     /* r[i] is order of x[i] */
```

SAS/IML 9.22 Statistics

```
x = {1 2 3, . 5 6, 7 8 9, 2 2 3, 1 1 3};
mean = mean(x);
nm = countmiss(x);  /* num. of missing    */
n = countn(x);      /* num. of nonmissing */
call qntl(q, x, {0.25 0.5});  /* quantile */
sd = sqrt(var(x));  /* std deviation      */
cov = cov(x);       /* covariance matrix  */
cor = corr(x);      /* correlation matrix */
```

Define Modules

```
start MyMod(x);         /* begin module    */
   return( 2#x );       /* return a value  */
finish;                 /* module defined  */
store module=MyMod;     /* store for later */

load module=MyMod;      /* load later      */
t = MyMod( 1:5 );       /* call module     */
```

From
Wicklin, Rick. 2010. *Statistical Programming with SAS/IML® Software*. Cary, NC: SAS Institute Inc.