

A Primer on Using PROC IML

SAS Interactive Matrix Language (IML) is a SAS procedure. IML is a matrix language and has built-in operators and functions for most standard matrix operations. To invoke the procedure, start with the `PROC IML` statement and end with `quit;`.

Reading in Data

A matrix can be defined in two ways in IML:

- (1) manually enter them using code, and
- (2) define a file containing the matrix which is later invoked in IML.

Entering Data in a Matrix:

```
proc iml; *invokes IML;
  a = 2; *scalar;
  b = {1 2 3 4}; *1 x 4 row vector;
  c = {1.0 0 0, 0.2 1.0 0, 0.8 0.4 1.0}; *3 x 3 (correlation) matrix;
quit; *closes IML;
```

The braces `{ }` encloses the values into a matrix and commas are used to separate rows.

Providing a file of elements:

Assume that you have a space delimited file containing elements of a 9 x 9 (correlation) matrix called “holzinger.txt”. The following code is used to read in the file and determine a matrix with the elements from the file “matrix.txt”.

```
filename matrix 'C:\... '; *directory where file is located;
data example; *creating a dataset called 'example';
  infile matrix(holzinger.txt); *linking the file 'holzinger.txt';
                                *to the filename 'matrix';
  input x1 x2 x3 x4 x5 x6 x7 x8 x9; *arbitrary column labels;
run;

proc iml;
  use example;
  read all var {x1 x2 x3 x4 x5 x6 x7 x8 x9}
          into X; *use all 9 columns of data for matrix X;
quit;
```

Printing out matrices of interest

IML can either be used interactively – where statements are executed immediately – or non-interactively. For interactive use, it is convenient to use the `reset log print;` statement which causes IML to display results in the log file, together with input statements. The `print` option causes IML to display the result of each assignment statement in the output.

For example, we want to specify a correlation matrix $A = \begin{bmatrix} 1.0 & & & \\ 0.5 & 1.0 & & \\ 0.6 & 0.7 & 1.0 & \\ 0.2 & 0.1 & 0.8 & 1.0 \end{bmatrix}$ and wanted to

have SAS print out the elements of A.

Using `reset log print;`

```
proc iml; *invokes IML;
  reset log print; *displays results in the log file;
  A = {1.0 0 0 0,
       0.5 1.0 0 0,
       0.6 0.7 1.0 0,
       0.2 0.1 0.8 1.0}; *manually entering the matrix;
quit; *closes IML;
```

The log file will display this:

```
proc iml;
NOTE: IML Ready
  reset log print;
  A = {1.0 0 0 0,
       0.5 1.0 0 0,
       0.6 0.7 1.0 0,
       0.2 0.1 0.8 1.0};
```

Using `print;`

```
proc iml; *invokes IML;
A = {1.0 0 0 0,
     0.5 1.0 0 0,
     0.6 0.7 1.0 0,
     0.2 0.1 0.8 1.0}; *manually entering the matrix;
print A; *displays matrix A in the output;
quit; *closes IML;
```

The output will display this:

		A			
	1	0	0	0	0
	0.5	1	0	0	0
	0.6	0.7	1	0	0
	0.2	0.1	0.8	1	1

Note that since IML can be used interactively, matrix A was printed in the output without executing the `quit;` statement.

Functions which create matrices

a. `I(size)`

Identity matrix where `size` = number of diagonal 1s.

b. `j(nrow, ncol, x)`

Constant matrix where `nrow` = number of rows, `ncol` = number of columns and `x` = element.

c. `diag(vector)`

Creates a square matrix with the elements of the vector along the main diagonal.

d. `diag(matrix)`

Creates a square matrix which retains the main diagonal of the argument matrix.

Example code:

```
proc iml;
  a = I(6); * 6x6 identity matrix;
  b = j(5,5,0); *5x5 matrix of 0's;
  c = j(6,1); *6x1 column vector of 1's;
  d=diag({1 2 4});
  e=diag({1 2, 3 4});
  print a; print b; print c; print d; print e;
quit;
```

Note, in using the `diag()` function, the vector or matrix has to be specified in braces `{}` and not in brackets `()`.

Output:

```

              A
    1      0      0      0      0      0
    0      1      0      0      0      0
    0      0      1      0      0      0
    0      0      0      1      0      0
    0      0      0      0      1      0
    0      0      0      0      0      1

              B
    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

    C
    1
    1
    1
    1
    1
    1

              D
    1      0      0
    0      2      0
    0      0      4

    E
    1      0
    0      4
```

Matrix Operations

- a. Addition operator: +
- b. Subtraction operator: -
- c. Sign reverse operator: -
- d. Absolute function: `abs(matrix)`
- e. Matrix product operator: *
- f. Square root function: `sqrt(matrix)`
- g. Transpose operator: `t(matrix)` or ` (backquote character)
- h. Determinant: `det(matrix)`
- i. Matrix inverse operator: `inv(matrix)`
- j. Trace of a matrix: `tr(matrix)`
- k. Eigen values: `eigval(matrix)`
- l. Eigen vectors: `eigvec(matrix)`
- m. Minimum value: `min(matrix)`
- n. Maximum value: `max(matrix)`

Example code:

```
proc iml;
  X = {1 2, 3 4};
  Y = {-4 3, -2 -1};

  a = X+X; *addition;
  b = X-X; *subtraction;
  c = -Y; *sign reversal;
  d = abs(Y); *taking absolute value;
  e = X*X; *matrix multiplication;
  f = sqrt(X); *taking square roots of the elements;
  g1 = X`; *transpose using back quote;
  g2 = t(X); *transpose using function;
  h = det(X); *matrix determinant;
  i = inv(X); *matrix inverse;
  j = tr(X); *trace of a matrix;
  k = eigval(X); *eigen values of matrix;
  l = eigvec(X); *eigen vector of matrix;
  m = min(X); *smallest matrix element;
  n = max(X); *largest matrix element;

  print a; print b; print c; print d; print e; print f; print g1; print g2;
  print h; print i; print j; print k; print l; print m; print n;
quit;
```

Output:

A		
2	4	
6	8	
B		
0	0	
0	0	
C		
4	-3	
2	1	
D		
4	3	
2	1	
E		
7	10	
15	22	
F		
1		1.4142136
1.7320508		2

G1		
1	3	
2	4	
G2		
1	3	
2	4	
H		
-2		
I		
-2	1	
1.5	-0.5	
K		
5.3722813	0	
-0.372281	0	
L		
0.4159736	-0.824565	
0.9093767	0.5657675	
M		
1		
N		
4		