NOTE #8: SAS/IML

SAS/IML (Interactive Matrix Language) is flexible and powerful programming language that operates on one or two dimensional data matrix. The software comes with lots of built-in operators and call routines that can be used for some complex calculations in matrix. We will discuss some basic features of IML trough examples.

There are two ways of generating matrix: defining a matrix in Proc IML statement or reading a SAS data set into a matrix. For example,

```
PROC IML;
      M1 = {11 12, 21 22, 31 32};
      M2 = M1`;
      PRINT M1 M2;
QUIT;
```

Note that the RUN statement is not used to execute IML statements. IML statements are executed immediately when submitted. Use the QUIT statement to terminate the IML procedure.

This will create a 3×2 matrix M1 and its transpose M2.

We can also create a matrix from a SAS data. For example,

```
DATA D1;
input X $ Y Z;
Datalines;
A 14 23
B 12 20
C 10 18
D 8 15
;

PROC IML;
 USE D1;
 READ ALL VAR{Y Z} INTO M1;
 PRINT M1;
QUIT;
```

This will read the SAS data D1 and generate the matrix M1 using the two numeric variables. Note that you can't mix numeric and character variables in a matrix.

You can create SAS dataset from matrices generated from PROC IML by adding the following statements:

```
CREATE dataname FROM matrixname;
APPEND FROM matrixname;
```

Following is a list of functions and operators used in PROC IML.

+, -, #, /	Element-by-element addition, subtraction, multiplication, and division
*	Matrix multiplication
**	Matrix power
##	Element by element power
sqrt(A)	Element-by-element square root

nrow(A), ncol(A)	number of rows, number of columns in A
det(A), inv(A), trace(A)	determinant, inverse, trace of A
eigval(A), eigvec(A)	a vector of eigenvalues and eigenvectors of A
diag(A)	make a diagonal matrix of A
vecdiag(A)	make a column vector out of the diagonal element of A

A[m,n]	extract (m,n) element of A
A[m,]	extract mth row of A (all columns)
A[,n]	extract nth column of A (all rows)
A[+], A[+,], A[,+]	overall sum, row vector of column sums, and column vector of row sums
A[:,], A[,:]	row vector of column means and column vector of row means
A[<>,], A[><,]	row vector of the maximum (minimum) of each of columns
A[,<>], A[,><]	column vector of the maximum (minimum) of each of rows
A[<:>,], A[>:<,]	row vector of index of the maximum (minimum) of each of columns
A[,<:>], A[,>:<]	column vector of index of the maximum (minimum) of each of rows

I(n)	identity matrix of size n
repeat(A, m, n)	create m by n matrix with matrix entry A
shape(A, m, n)	create m by n matrix with elements in A
J(m,n,x)	create m by n matrix with element x
A ‖ B	put the two matrices horizontally
A // B	put the two matrices vertically
t(A)	A transpose

```
/* Example 8-1 */
PROC IML;
   M = { 9 5 8, 6 1 7, 2 9 3};
   ColMin=M[><,]; ColMax= M[<>,];
   RowMin=M[,><]; RowMax= M[,<>];
   E=eigval(M); V=eigvec(M);

  sum=M[+,]; mean=M[:,]; ss=M[##];
  ss1= (M#M)[+];
  diag=diag(M); dvec=vecdiag(M);  *diag function also make a diagonal matrix ;
                                  * from a vector;

  X=repeat(M, 2,2);   XX=shape(M, 2,2);

  PRINT M ColMin ColMax RowMin RowMax;
  PRINT E V;
  PRINT sum mean ss ss1;
  PRINT diag dvec;
  PRINT X XX;

  CREAT D1 FROM M;
  APPEND FROM M;
QUIT;

PROC PRINT DATA = D1; RUN;


/* Example 8-2 */
/* OLS estimation for a multiple regression*/

PROC IML;
 USE SASHELP.GNP;
```

```
 READ point (1:100) VAR{GNP CONSUMP INVEST} INTO M;

 N=NROW(M);  r=NCOL(M);
 Y=M[,1];  X=SHAPE(1,N,1)||M[,2:3];

 BETA=INV(t(X)*X)*t(X)*Y;
 SIGMA=SQRT(t(Y-X*BETA)*(Y-X*BETA)/(N-r));
 SE=SIGMA*SQRT(VECDIAG(INV(t(X)*X)));
 TVALUE=BETA/SE;
 PVALUE=(1-PROBT(ABS(TVALUE),N-r))*2;
 Table=BETA||SE||TVALUE||PVALUE;

 Title "Parameter Estimates";
 PRINT Table(|colname={ESTIMATE SE TVALUE PVALUE}
              rowname={INTERCPT CONSUMP INVEST} FORMAT=8.2|);

 CREATE D1 FROM Table;
 APPEND FROM Table;
QUIT;


/* Example 8-3 */
/* Solving nonlinear Equation */
/* one dimension */

goptions
colors=(black );
axis1 style=0 major=none minor=none value=none label=none;
axis2 style=0 major=none minor=none value=none label=none;

PROC IML;
  Create Newton1 var{X F};
  X=30; F=10;
  Do i=1 to 20  until (abs(F) < .00001);
     F=X*X-cos(X);
     DF=2*X+sin(X);
     X=X-(F/DF); print X F;
  end;
 append;
quit;

data forplot; set Newton1;
  do c=0 to 2 by .01;
    y=c*c-cos(c); output;
  end;
run;
proc gplot;
  plot y*c F*X F*X/overlay
  haxis= axis1  vaxis= axis2 href = 0 vref = 0;
  axis1 order = 0 to 2 by .2 ;
  axis2 order = -1 to 2 by .1;
  SYMBOL1 V=o H=.5 CV=BLACK;
  SYMBOL2 V=X  H=3 CV=RED;
run;

run; quit;


/* Example 8-4 */
/* Solving nonlinear Equation System */
/* Two dimension */

PROC IML;
```

```
create final var{x y};
do a = -2 to 2  by 4 ;
  do b = -1 to 1 by 2 ;
    theta =( a ) // ( b ); F = { 1 , 1 };
    do jj = 1 to 20 until (abs(SUM(F)) < .00001 );
      x = theta[1,];
      y = theta[2,];
      F = (x##2 + (4* y##2) - 4)// (y - x##2 + (.4*x) + 1.96);
      DF = ( (2*x)  || (8 *y )  )  //( ( -2*x + .4  ) || 1  ) ;
      theta = theta - (INV(DF)*F);
      theta2 = theta`;
    end;
    print theta2;
   append;
   end;
end;
quit;

data graph;
  set final;
  do c = -2 to 2 by .001;
    z = sqrt((4 - c**2)/4);
    z1 = -sqrt((4 - c**2)/4) ;
    z2 = c**2 - .4*c - 1.96;
    output;
  end ;

proc gplot;
  plot z*c  z1*c z2*c y*x/ overlay
  haxis= axis1  vaxis= axis2 href = 0 vref = 0;
  axis1 order = -3 to 3 by 1 ;
  axis2 order = -3 to 3 by 1;
  SYMBOL1 V=o H=.5 CV=BLACK;
  SYMBOL2 V=o H=.5 CV=BLACK;
  SYMBOL3 V=o H=.5 CV=BLACK;
  SYMBOL4 V=X H=.5 CV=RED;
run;


/* IN-CLASS  PRACTICE 1

Using IML solve the equation
```

$$F(x,y)=\begin{pmatrix} y-x^2+\cos(x) \\ y-4x^2+2x+2 \end{pmatrix}=0 .$$

```
Plot the function.
*/


/* Example 8-5 */
/* Here we use IML to find the maxima and/or minima of a polynomial function */

PROC IML;
  Create Newton2 var{X F};
  Do  a = -5 to 5 by 10;
     X=a; F=10;
     Do i=1 to 20  until (abs(F) < .00001);
       F=6*X##2 - 6*X - 12;
       DF=12*X - 6;
       X=X-(F/DF); print X F;
     end;
```

```
   append;
  end;
quit;

data forplot; set Newton2;
  do c=-5 to 5 by .01;
       y1=2*c**3-3*c**2-12*c+12; output;
       y2=6*c**2-6*c-12; output;
       y3 = 12*c-6;
  end;
run;
proc gplot;
  plot y1*c y2*c y3*c F*X/overlay
  haxis= axis1  vaxis= axis2 href = 0 vref = 0;
  axis1 order = -5 to 5 by 1 ;
  axis2 order = -30 to 30 by 10;
  SYMBOL1 V=o H=.5 CV=BLACK;
  SYMBOL2 V=o  H=.5 CV=RED;
  SYMBOL3 V=o  H=.5 CV=BLUE;
  SYMBOL4 V=X  H=3 CV=BLACK;
run; quit;

/* In-class PRACTICE 2 */
/* Use IML to find the maxima and minima of the function

      F(x) = x^4 - 2x^2

Graph the function and the first and the second derivatives.
*/

/* Example 8-6*/
/* This example use IML to find Maximum Likelihood Estimates for the Normal */
/* distribution using the Newton method */
/* Detailed discussion will be given in class */

DM "OUTPUT;CLEAR;LOG;CLEAR";
%LET MU=1;
%LET STD=2;
%LET N=1000;
DATA NORM1;
DO I=1 TO &N;
X=&MU+&STD*RANNOR(0); OUTPUT;
END;

DATA NORM1; SET NORM1;
KEEP X ;

TITLE " MLE FOR NORMAL DISTRIBUTION ";
PROC IML;
USE NORM1; READ ALL INTO X; X=X`; LLD1= { 1, 1 };
NN=NCOL(X);
THETA={ 0 , .1 };
DO JJ=1 TO 20 UNTIL ( abs(SUM(LLD1)) < .0001 );
  MU=THETA[1,]; S2=THETA[2,];
  ONE=SHAPE(1,NN,1);

  LLD1 = ( ((X-MU)* ONE)/S2 )//( (-NN/(2*S2))+( (X-MU)*(X-MU)`/(2*S2*S2) ));

  LLD2 =(-NN/S2  ||  -( (X-MU)* ONE) / (S2*S2)  ) //
        (-((X-MU)* ONE)/(S2*S2) || (NN/(2*S2*S2))-((X-MU)*(X-MU)`/(S2**3)));

  THETA = THETA - INV(LLD2)*LLD1;
```

```
PRINT JJ THETA;
END;
XBAR=X[ ,+ ] / NN;
SIGMA2=((X-XBAR)*(X-XBAR)`) /NN ;
PRINT XBAR SIGMA2; RUN; QUIT;
```