The George Washington University
Department of Statistics

Stat 6197 – Spring 2020

Week 12 – April 10, 2020

Major Topic: Matrix Operations and Functions in SAS/IML

Detailed Topics:

1. SAS/IML Basics: Comparison with SAS Data Steps
2. Statements in SAS IML
3. Ways to Create Matrices in SAS/IML
4. Subscript Operations in SAS/IML
5. Ways to Extract Elements from Matrices (Using Subscripts vs. Using Indices)
6. Matrix Operations and Functions in SAS/IML
7. Creating SAS Data Sets from Vectors and Matrices
8. Data Processing in SAS/IML
9. Accessing Rows and Columns by Names

Miscellaneous:

1. Review for Quiz 3
2. Tips for Take-Home Assignment 2

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

The following references are clickable URLs.

Getting Started with the SAS/IML® Language

SAS/IML 14.3 User's Guide

The chi-square test: An example of working with rows and columns in SAS

The WHERE clause in SAS/IML

Complex assignment statements: CHOOSE wisely

An easy way to specify dates and times

Finding matrix elements that satisfy a logical expression

Finding observations that satisfy multiple conditions: The LOC-ELEMENT technique
Calling Functions in the R Language

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

The following is the clickable URL.
Understanding the SAS/IML Language

- Defining a Matrix
- Matrix Names and Literals
- Creating Matrices from Matrix Literals
- Types of Statements
- Missing Values
- Summary

The above bulleted headings are from the same SAS/IML Language documentation.

Salient SAS/IML Features

- Matrix computations
- Data processing
- Control statements
- Modules and subroutines
- Numerical analysis and statistical functions
- Time series functions
- Optimization algorithms
- Data simulation

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Creating Vectors with SAS/IML

```
 1 * Example_creating_Vectors.sas;
 2 proc IML;
 3 Col_vector1 = {2, 4, 6, 8};    /** r X 1 matrix, column vector,
 4                                     evenly spaced values **/
 5 Col_vector2 = j(4, 1, 5);      /** 4 X 1 matrix,
 6                                    column vector of 5's  **/
 7 Row_vector1 = 1:3;             /** 1 X c matrix, row vector,
 8                                    increasing seq. of numbers **/
 9 Row_vector2 = 1:-1;            /** 1 X c matrix, row vector,
10                                    decreasing seq. of numbers **/
11 Row_vector3 = do(10, 70, 20); /** 1 X c matrix, row vector,
12                                    positive increment **/
13 Row_vector4 = do(15, -10, -5);/** 1 X c matrix, row vector,
14                                    negative increment **/
15 x_scalar = 5;                  /** 1 X 1 matrix, scalar **/
16 print Col_vector1 Col_vector2;
17 print Row_vector1, Row_vector2;
18 print Row_vector3;
19 print Row_vector4;
20 print x_scalar;
21 quit;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Ways to create matrices using SAS/IML software

- Entering data yourself as a matrix literal (i.e., scalar, row vector, column vector or matrix) - braces to enclose the data values and, if needed, commas to separate rows
- Using assignment statements
- Using matrix-generating functions
- Creating submatrices from existing matrices with subscripts
- Using SAS data sets

Matrix Computations with SAS/IML

- Matrix as the fundamental data element
- Two-dimensional array (row-by-column array of values
- Matrices in SAS/IML can be character or numeric (not mixed character and numeric)
- Dimension of a matrix can be changed
- Type of a matrix can be changed
- Elements of matrix can be replaced

Matrix Operations with SAS/IML

- Solving linear systems
- matrix decompositions

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Categories of Statements in PROC IML

- Control Statements
- Commands
- CALL Statements and Subroutines
- Assignment Statements

Control Statements

- IF-THEN/ELSE
- DO/END
- Iterative DO
- Start/Finish

Commands (mostly for data processing purposes)

- Free
- Print
- Read
- Use

CALL Statements and Subroutines

Assignment Statements

```
proc iml;
 A = {1 2 3, 4 5 6};
 Print A;
Quit;
```

```
A

1       2       3
4       5       6
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

- Creating Matrices by Entering Data Values Manually

  The braces {} encloses the values into a matrix (i.e., row vectors, column vectors or matrices). Commas are used to separate rows, particularly in column vectors and matrices.

Matrix Addition

```
2  *Example_matrix_addition.sas;
3  options nodate nonumber;
4  proc iml;
5  M1 = {1 2 3,4 5 6, 7 8 9};  *3 X 3 matrix ;
6       M2 = {7 8 9, 4 5 6, 1 2 3};  *3 X 3 matrix;
7       M1_M2_Addition = M1+M2; *Matrix addition;
8       print M1 M2; print M1_M2_Addition;
9  quit;
```

```
M1                              M2

 1         2         3         7         8         9
 4         5         6         4         5         6
 7         8         9         1         2         3
```

```
M1_M2_Addition

 8        10        12
 8        10        12
 8        10        12
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Subtraction

```
2  *Example_matrix_subtraction.sas;
3  options nodate nonumber;
4  proc iml;
5       M1 = {1 2 3,4 5 6, 7 8 9} ;  *3 X 3 matrix;
6       M2 = {7 8 9, 4 5 6, 1 2 3} ;  *3 X 3 matrix ;
7       M1_M2_Subtract = M1-M2; *Matrix Subtraction;
8       print M1; print M2; print M1_M2_Subtract;
9  quit;
```

```
        M1

 1          2          3
 4          5          6
 7          8          9


        M2

 7          8          9
 4          5          6
 1          2          3


 M1_M2_Subtract

-6         -6         -6
 0          0          0
 6          6          6
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Elementwise Multiplication

```
2  *Example_Matrix_Elwise_Multi.sas;
3  options nodate nonumber;
4  proc iml;
5       M1 = {1 2 3,4 5 6, 7 8 9} ;  *3 X 3 matrix;
6       M2 = {7 8 9, 4 5 6, 1 2 3} ;  *3 X 3 matrix ;
7       M1_M2_E_Times= M1#M2; *Elemetwise multplication;
8       print M1; print M2; print M1_M2_E_Times;
9  quit;
```

```
        M1

 1          2          3
 4          5          6
 7          8          9


        M2

 7          8          9
 4          5          6
 1          2          3


 M1_M2_E_Times

 7         16         27
16         25         36
 7         16         27
```

Matrix Elementwise Power

```
2  *Example_Matrix_Elwise_Power.sas;
3  options nodate nonumber;
4  proc iml;
5       M1 = {1 2 3,4 5 6, 7 8 9} ;  *3 X 3 matrix;
6       M1_E_Power2= M1##2;
7       print M1; print M1_E_Power2;
8  quit;
```

```
        M1

 1          2          3
 4          5          6
 7          8          9


   M1_E_Power2

 1          4          9
16         25         36
49         64         81
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Elementwise Square Root

https://www.ptcusercommunity.com/thread/33064

MikeArmstrong Oct 30, 2010 6:57 AM (in response to MySchizoBuddy)

Re: How to find square root of a matrix

$$MM := \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \end{pmatrix} \qquad \overrightarrow{\sqrt{MM}} = \begin{pmatrix} 1 & 1.4142 & 1.7321 & 2 & 2.2361 \\ 2.4495 & 2.6458 & 2.8284 & 3 & 3.1623 \end{pmatrix}$$

```
2  *Example_Matrix_Elwise_sr.sas;
3  options nodate nonumber;
4  proc iml;
5       M1 = {1 2 3 4 5, 6 7 8 9 10} ;
6       M1_E_sq_root= M1##.5; *Elementwise square root;
7       print M1; print M1_E_SQ_ROOT;
8  quit;
```

```
                          M1

          1         2         3         4         5
          6         7         8         9        10


                    M1_E_sq_root

          1 1.4142136 1.7320508         2  2.236068
  2.4494897 2.6457513 2.8284271         3 3.1622777
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Scalar Addition

```
2  *Example_Matrix_scalar_add.sas;
3  options nodate nonumber;
4  proc iml;
5         M1 = {1 2 3,4 5 6, 7 8 9} ;
6         M1_Scalar_Addition = M1+2; *Scalar addition;
7         print M1; print M1_Scalar_Addition;
8  quit;
```

```
        M1

  1         2         3
  4         5         6
  7         8         9


M1_Scalar_Addition

  3         4         5
  6         7         8
  9        10        11
```

Matrix Scalar Subtraction

```
2  *Example_Matrix_scalar_subtract.sas;
3  options nodate nonumber;
4  proc iml;
5         M1 = {1 2 3,4 5 6, 7 8 9} ;
6         M1_Scalar_Subtract= M1-2; *Scalar subtraction;
7         print M1; print M1_Scalar_Subtract;
8  quit;
9
```

```
        M1

  1         2         3
  4         5         6
  7         8         9


M1_Scalar_Subtract

 -1         0         1
  2         3         4
  5         6         7
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Scalar Division

```
2  *Example_Matrix_scalar_div.sas;
3  options nodate nonumber;
4  proc iml;
5         M1 = {1 2 3,4 5 6, 7 8 9} ;
6         M1_Scalar_Division= M1/2; *Scalar division;
7       print M1; print M1_Scalar_Division;
8  quit;
```

```
          M1

    1         2         3
    4         5         6
    7         8         9


  M1_Scalar_Division

  0.5         1       1.5
    2       2.5         3
  3.5         4       4.5
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Product

```
2  *Example_Matrix_product.sas;
3  options nodate nonumber;
4  proc iml;
5       M1 = {1 2 3, 4 5 6};
6       M2 = {7 8, 9 10, 11 12};
7       M1_M2_Product=M1*M2;
8       print M1; print M2; print  M1_M2_Product;
9  quit;
```

```
          M1

  1         2         3
  4         5         6


          M2

        7         8
        9        10
       11        12


  M1_M2_Product

       58        64
      139       154
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Division

http://www.soph.uab.edu/Statgenetics/People/MBeasley/Courses/IML-Guide.pdf

```
 2 *Example_Matrix_division.sas;
 3 options nodate nonumber;
 4 proc iml;
 5      M1 = {1 2 3, 4 5 6, 7 8 9};
 6      M2 = {8 2 3, 4 9 6, 7 1 5};
 7      M1_M2_Div=M1/M2;
 8      print M1; print M2;
 9      print  M1_M2_Div;
10 quit;
```

```
          M1

     1         2         3
     4         5         6
     7         8         9


          M2

     8         2         3
     4         9         6
     7         1         5


       M1_M2_Div

 0.125         1         1
     1 0.5555556         1
     1         8       1.8
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Matrix Power

http://www.soph.uab.edu/Statgenetics/People/MBeasley/Courses/IML-Guide.pdf

```
3  *Example_Matrix_Power.sas;
4  proc iml;
5       M1 = {1 2 3, 4 5 6, 7 8 9};
6       M1_Power2=M1**2;
7       print M1;
8       print  M1_Power2;
9  quit;
```

M1

1	2	3
4	5	6
7	8	9

M1_Power2

30	36	42
66	81	96
102	126	150

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Horizontal Concatenation of Matrices

```
3  *Example_Horizontal_Concat.sas;
4  proc iml;
5       M1 = {1 2 3,4 5 6, 7 8 9} ;
6       M2 = {7 8 9, 4 5 6, 1 2 3} ;
7       H_concat= M1 || M2;
8       print M1; print M2; print H_concat;
9  quit;
```

M1

1	2	3
4	5	6
7	8	9

M2

7	8	9
4	5	6
1	2	3

H_concat

1	2	3	7	8	9
4	5	6	4	5	6
7	8	9	1	2	3

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Vertical Concatenation of Matrices

```
3  *Example_Vertical_Concat.sas;
4  proc iml;
5       M1 = {1 2 3,4 5 6, 7 8 9} ;
6       M2 = {7 8 9, 4 5 6, 1 2 3};
7       V_concat= M1 // M2;
8       print M1; print M2; print V_concat;
9  quit;
```

M1

1	2	3
4	5	6
7	8	9

M2

7	8	9
4	5	6
1	2	3

V_concat

1	2	3
4	5	6
7	8	9
7	8	9
4	5	6
1	2	3

(The following is the clickable URL)

Extracting elements from a matrix: rows, columns, submatrices, and indices

Two Ways to Extract Elements from a Matrix

- Using subscripts (extracting a rectangular portion of matrix)
- Using indices (extracting values from a non-rectangular pattern)

(The following is the clickable URL)

Empty subscripts: How to fill rows and columns of a matrix

Filling Rows and Columns of a Matrix

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Selecting a Row or Column

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- Access a row by eliminating the column subscript:
  ```
  row1=X[1, ];
  ```
- Access a column by eliminating the row subscript:
  ```
  col2=X[ ,2];
  ```

36

Reduction Operators: Examples

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix}$$

- `Y=X[+, ]` produces Y=[14 15 16]
- `Y=X[+, <>]` produces Y=[16]
- `Y=X[,<>][+,]` produces Y=[18]
- `Y=X[<:>,]` produces Y=[3 3 3]

42

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Use of Matrix Reduction Operators

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
row_sum=A[,+ ];
print A row_sum ;
quit;
```

A			row_sum
1	2	3	6
4	5	6	15
9	8	7	24
3	2	1	6
5	4	2	11

`row_sum=A[,+ ];` The row dimension's subscript is missing, so the resulting matrix (i.e., row_sum) has the same number of rows, five. A positive (+) sign for the column dimension's subscript means summing across rows. The print statement prints the entire matrix and the total for each row.

```
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
col_sum=A[+, ];
print A, col_sum ;
quit;
```

A		
1	2	3
4	5	6
9	8	7
3	2	1
5	4	2

col_sum		
22	21	19

`col_sum=A[+, ];` The column dimension's subscript is missing, so the resulting matrix (i.e., Col_sum) has the same number of columns, three. A positive (+) sign for the row dimension's subscript means summing across columns. The print statement prints the entire matrix and the total for each column.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
max_c_sum= A[+, <>];
print A,max_c_sum ;
quit;
```

A	
1 2 3 4 5 6 9 8 7 3 2 1 5 4 2	In the second assignment statement (above), the expression produces a vector named **max_c_sum** that contains the maximum of the column totals. The print statement prints the entire matrix **A** as well as the vector **max-c_sum**.

max_c_sum
22

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
min_c_sum= A[+, ><];
print A, min_c_sum ;
quit;
```

A	
1 2 3 4 5 6 9 8 7 3 2 1 5 4 2	In the second assignment statement (above), the expression produces a vector named **min_c_sum** that contains the minimum of the column totals. The print statement prints the entire matrix **A** as well as the vector **min_c_sum**.

min_c_sum
19

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
i_col_maxima=A[<:>,];
print A, i_col_maxima ;
quit;
```

A 1 2 3 4 5 6 9 8 7 3 2 1 5 4 2 i_col_maxima 3 3 3	In the second assignment statement (above), the expression produces a 1 X 3 matrix named **i_col_maxima** that contains the index of the column maxima. The print statement prints the matrices **A** and **i_col_maxima**.

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 4 1, 5 7 2};
i_col_minima=A[>:<,];
print A, i_col_minima ;
quit;
```

A 1 2 3 4 5 6 9 8 7 3 4 1 5 7 2 i_col_minima 1 1 4	In the second assignment statement (above), the expression produces a 1 X 3 matrix named **i_col_minima** that contains the index of the column minima. The print statement prints the **A** and **i_col_minima**. .

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
sum_max_r=A[,<>] [+, ];
print A ,sum_max_r ;
quit;
```

A 1 2 3 4 5 6 9 8 7 3 2 1 5 4 2 **sum_max_r** 26	In the second assignment statement (above), the expression produces a vector named **sum-max-r** that contains the sum of the rowwise maximum values. The print statement prints the matrices A and **sum-max-r**.

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
sum_c12 = A[+, 1:2];
print A, sum_c12 ;
quit;
```

A 1 2 3 4 5 6 9 8 7 3 2 1 5 4 2 sum_c12 22 21	In the second assignment statement (above), the expression produces a submatrix named **sum_c12** that contains the totals of columns 1 and 2, respectively. The print statement prints the matrix **A** as well as the 1 X 2 matrix **sum_c12**.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
max_row_1_2_col_1_3 = A[1:2, 1:3] [<>];
print A, max_row_1_2_col_1_3 ;
quit;
```

A	
1 2 3	In the second assignment statement (above), the expression produces a vector named **max_row_1_2_col_1_3** that contains the maximum value of the row-column dimension specified. Notice the use of vectors in the subscript operator in the first argument of the expression. The print statement prints the matrix **A** and the vector **max_row_1_2_col_1_3**.
4 5 6	
9 8 7	
3 2 1	
5 4 2	
max_row_1_2_col_1_3	
6	

```
ods html;
proc iml;
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
min_row_1_2_col_1_3 = A[1:2, 1:3] [><];
print A, min_row_1_2_col_1_3;
quit;
```

A			
1	2	3	In the second assignment statement (above), the expression produces a vector named **min_row_1_2_col_1_3** that contains the minimum value of the row-column dimension specified. Notice the use of vectors in the subscript operator in the first argument of the expression. The print statement prints the matrix **A** and the vector **min_row_1_2_col_1_3**.
4	5	6	
9	8	7	
3	2	1	
5	4	2	
min_row_1_2_col_1_3			
1			

```
proc iml;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
A = {1 2 3, 4 5 6, 9 8 7, 3 2 1, 5 4 2};
mean_1_2_3 = A[:, ] ;
print A, mean_1_2_3;
quit;
```

A 1 2 3 4 5 6 9 8 7 3 2 1 5 4 2 **mean_1_2_3** 4.4 4.2 3.8	In the second assignment statement (above), the expression produces a 1 X 3 matrix named **mean_1_2_3** that contains the mean of each of the 3 columns. The print statement prints the matrices **A** and **mean_1_2_3**.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
1  *Example_marg_prob.sas;
2  *Author: Ksharp - 04/08/2018
3    - Comments on Wicklin 's Code;
4  proc iml;
5  cName = {"black" "dark" "fair" "medium" "red"};
6  rName = {"blue" "brown" "green"};
7  C = { 6 51 69 68 28,
8  16 94 90 94 47,
9  0 37 69 55 38};
10 /* marginal probability of each column */
11 colMarg = C[+, ]/c[+];
12 /* marginal probability of each row */
13 rowMarg = C[ ,+]/c[+];
14 expect=(rowMarg*colMarg)#c[+];
15 print c;
16 print colMarg;
17 print rowMarg;
18 print expect[r=rName c=cName];
19 quit;
```

```
                              C

          6         51         69         68         28
         16         94         90         94         47
          0         37         69         55         38

                        colMarg

  0.0288714 0.2388451 0.2992126 0.2847769  0.148294

                        rowMarg

                       0.2913386
                       0.4475066
                       0.2611549

                        expect

           black      dark       fair     medium       red
blue   6.4094488 53.023622 66.425197 63.220472  32.92126
brown  9.8451444 81.446194  102.0315 97.108924 50.568241
green  5.7454068 47.530184 59.543307 56.670604 29.510499
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Creating Matrices Using Operators and Functions

```
 3  *Example_Create_matrices_op_func.sas;
 4  proc iml;
 5       rv=1:3; *create a row vector;
 6       cv=t(1:3); *create a column vector;
 7       reverse_rv= 3:1; *create a row vector;
 8       Char_vec  = 'Day1': 'Day7';   *character vector;
 9       print rv, cv, reverse_rv, Char_vec;
10  quit;
```

```
                rv

          1         2         3
                cv

                  1
                  2
                  3
          reverse_rv

          3         2         1
             Char_vec

Day1 Day2 Day3 Day4 Day5 Day6 Day7
```

Creating an identity matrix

```
 3  *Example_Create_identity_matrix.sas;
 4  proc iml;
 5       I_mat=I(5);    *5x5 identity matrix;
 6       print i_mat;
 7  quit;
```

```
              I_mat

1         0         0         0         0
0         1         0         0         0
0         0         1         0         0
0         0         0         1         0
0         0         0         0         1
```

```
 3  *Example_create_matrix_series.sas;
 4  proc iml;
 5   /*... arithmetic series with some increment*/
 6  series= do(5,50,10);
 7  print series;
 8  quit;
```

```
               series

 5        15        25        35        45
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
 3  *Example_Create_matrix_J_Function.sas;
 4  proc iml;
 5    obs= {8 4 4 3 6 11};
 6    n=sum(obs);
 7    ncol_obs = ncol(obs);
 8    *constant matrix - J(nrow,ncol,value);
 9    p=j(1, ncol_obs, 1/ncol_obs);
10    np=n*p;
11    print obs, n, ncol_obs, p, np;
12   quit;
```

```
                              obs

        8         4         4         3         6        11
                               n

                             36
                          ncol_obs

                              6
                               p

 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
                              np

        6         6         6         6         6         6
```

```
 3  *Example_Create_matrix_J_Function2.sas;
 4  proc iml;
 5  /*Creating a  matrix with a J function
 6    5 X 1 column vector of 1's*/
 7  J_b=j(5,1);       *5 X 1 column vector of 1's;
 8  print J_b;
 9  quit;
```

```
          J_b

                1
                1
                1
                1
                1
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
3  *Example_Create_diagoanal_matrix.sas;
4  proc iml;
5   Diag_mat=diag( {1  2 3} );
6  print Diag_mat;
7  quit;
```

```
Diag_mat

1         0         0
0         2         0
0         0         3
```

```
3  *Example_Create_col_vector_vecdig_func.sas;
4  proc iml;
5      X={1 2 3, 4 5 6, 7 8 9};
6      vec_diag = vecdiag(X);
7      print X, vec_diag;
8  quit;
```

```
X

1         2         3
4         5         6
7         8         9
vec_diag

          1
          5
          9
```

```
3  *Example_transposing_matrix.sas;
4  proc iml;
5    m = {1 2, 3 4, 5 6};
6      T_m = t(m) ;
7  print m, T_m;
8  quit;
```

```
m

1         2
3         4
5         6
T_m

1         3         5
2         4         6
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
3  *Example_inverting_matrix.sas;
4  proc iml;
5       m = {7 1 2, 3 8 5, 6 7 8};
6       I_m = inv(m) ;
7  print m, I_m;
8  quit;
```

```
                m

          7         1         2
          3         8         5
          6         7         8
              I_m

    0.1870968 0.0387097 -0.070968
    0.0387097  0.283871 -0.187097
    -0.174194 -0.277419 0.3419355
```

```
3  *Example_Repeat_matrix.sas;
4  proc iml;
5       X={1 2, 0 4};
6       repeat_x22= repeat(X, 3, 2);
7  print X, repeat_x22;
8  quit;
```

Repeat matrix - with 2 blocks rowwise and then 3 blocks columnwise

```
                X

          1         2
          0         4
       repeat_x22

    1         2         1         2
    0         4         0         4
    1         2         1         2
    0         4         0         4
    1         2         1         2
    0         4         0         4
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
 3  *Example_matrix_functions.sas;
 4  proc iml;
 5       X={1 2 3, 4 5 6, 7 8 9};
 6       Y={7 1 2, 3 8 5, 6 7 8};;
 7       sum_X_Y = sum(X, Y);
 8       max_X= max(X);
 9       ncol_Y = ncol(Y);
10       nrow_Y=nrow(Y);
11    print x, y, sum_X_Y, max_X, ncol_y, nrow_y ;
12    quit;
```

```
                X

          1           2           3
          4           5           6
          7           8           9
                Y

          7           1           2
          3           8           5
          6           7           8
     sum_X_Y

                92
      max_X

                 9
      ncol_Y

                 3
      nrow_Y

                 3
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

http://blogs.sas.com/content/iml/2011/08/15/complex-assignment-statements-choose-wisely/
Complex assignment statements: CHOOSE wisely. By Rick Wicklin

What the CHOOSE function does
"The CHOOSE function is like a compact alternative to a compound assignment statement. In other words, it can eliminate an IF-THEN/ELSE statement. The CHOOSE function takes three arguments: a condition to evaluate as true or false, a value to use when the condition is true, and a value to use when the condition is false. Each argument can be vector-valued. For example, the following statements use the CHOOSE function to return the absolute value of a vector… The CHOOSE function is a compact way to implement a complex assignment statement, but it is important to remember that all arguments are evaluated prior to being passed into the function."

```
1  *Example_Choose1.sas;
2  proc iml;
3  id = {1,2,3,4,5};
4  quiz1= {12,18,13,9,7};
5  makeup={10, 17, 17, 12, 13};
6  r_quiz1=choose(makeup>quiz1, makeup, quiz1);
7  print  id quiz1 makeup r_quiz1;
8  quit;
```

id	quiz1	makeup	r_quiz1
1	12	10	12
2	18	17	18
3	13	17	17
4	9	12	12
5	7	13	13

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
1  *Example_update.sas;
2  data master;
3   input id quiz1 @@;
4   datalines;
5   1 12 2 18 3 13 4 9 5 7
6   ;
7  data transact;
8   input id quiz1  @@;
9   datalines;
10  3 17 4 12 5 13
11 ;
12 data updated;
13   UPDATE master transact; by ID;
14 run;
15 proc print data=updated noobs; run;
```

id	quiz1
1	12
2	18
3	17
4	12
5	13

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
1  *Example_Choose_Keshan_sas-L.sas;
2  *Acknowledgements: Xia Kesan SAS-L;
3  proc iml;
4    faulty = {15   12   13,
5              16   29   13,
6              15   12   13,
7              18   58   11 };
8    updates = {.    .    .,
9              16   29   .,
10              .    .    .,
11             18   58   .};
12   want=choose(updates^=.,updates,faulty );
13   print want;
14   quit;
```

want

15	12	13
16	29	13
15	12	13
18	58	11

The LOC Function in SAS IML

The LOC function is used to find the elements of a matrix that satisfies a condition in SAS/IML language as the WHERE and IF statements are used to locate observations and subset data in data step.
http://blogs.sas.com/content/iml/2015/01/20/elements-satisfy-logical-expression.html - by Rick Wicklin

- The argument to the LOC function is an expression that resolves to a vector of 0s and 1s.
- The result of the LOC function is always a row vector.
- The LOC function returns indices of x, not values of x. To obtain the values, use x[k]. (Indices and subscripts are related; for vectors, they are the same.)

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
 3 proc iml;
 4  x = {. -5 2  5,
 5      -2  . 3  4,
 6       4  . . -1};
 7  missingLoc = loc(x=.);                /* missing values */
 8  negativeLoc = loc(x^=. & x<0);        /* nonmissing and negative  */
 9  evenLoc = loc(mod(x,2)=0);            /* n is even if (n mod 2)=0 */
10  print missingLoc, negativeLoc, evenLoc;
11
```

```
        missingLoc

 1          6          10          11

       negativeLoc

      2          5          12

          evenLoc

 3          5          8           9
```

Storing a Matrix into the Default Library

```
*Example_store_matrix.sas;
PROC IML;
  USE sashelp.class;
  READ all var _num_ INTO Mymat;
  store Mymat;
  CLOSE sashelp.class;
QUIT;
```

The STORE statement saves the matrix MYMAT to storage in the defaults library **work**.

Loading and Printing the Matrix from the Default Library

```
PROC IML;
 LOAD Mymat;
 PRINT Mymat;
 QUIT;
```

The LOAD statement recalls entry from back into IML workplace from storage.
The PRINT statement prints the matrix MYMAT.

Saving Matrices (or Modules) Permanently

```
libname imlin "D:\";
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
PROC IML;
reset deflib=imlin;
  USE sashelp.class;
  READ all var _num_ INTO Mymat2;
  reset storage=imlin.cat1;
  store Mymat2;
  show storage;
  CLOSE sashelp.class;
QUIT;
```

The RESET statement with DEFLIB = operand is used to specify the library name. The RESET STORAGE statement is used to specify both library reference and catalog. The STORE statement saves the matrix MYMAT2 in the catalog named CAT1 in the IMLIN library.

RESET and LOAD statements

```
libname xmlin "D:\";
PROC IML;
 RESET deflib=imlin;
 RESET STORAGE=imlin.cat1;
 LOAD Mymat2;
 PRINT Mymat2;
 QUIT;
```

The RESET statement with DEFLIB = operand is used to specify the library name. The RESET STORAGE statement is used to specify both library reference and catalog. The LOAD statement recalls the matrix MYMAT2 that was earlier saved permanently in the catalog named CAT1 in the IMLIN library.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Reading Variables from SAS Data Set into a Matrix

```
1  *Example_read_INTO1.sas;
2  PROC IML;
3     USE sashelp.class;
4     READ all var _num_ INTO c_m_nummat
5           where(sex='M');
6     CLOSE sashelp.class;
7     PRINT c_m_nummat;
8  QUIT;
```

```
c_m_nummat

14        69     112.5
14      63.5     102.5
12      57.3        83
13      62.5        84
12        59      99.5
16        72       150
12      64.8       128
15        67       133
11      57.5        85
15      66.5       112
```

Line 3: The USE statement opens the SAS dataset SASHELP.CLASS for read-access and sets it as the current dataset.

Line 4: The READ statement reads all obdervations in the curent data set and produces a matrix that contains all numeric variables (_NUM_) specified by the VAR clause for those observations whose value for **SEX** is **M.** Notice the INTO and WHERE clauses.
Line 6: The CLOSE statement is used to close the SAS data set opened with the USE statement.
Line 7: The PRINT sratement prints the matrix.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Reading Variables from SAS Data Set into Column Vectors

```
21 PROC IML;
22    USE sashelp.class;
23    READ all var _num_ where(sex='M');
24    CLOSE sashelp.class;
25    PRINT age height weight;
26 QUIT;
```

Notice that there is no INTO clause in the READ statement, and each numeric variable in the VAR clause becomes a column vector with the same name as the variable in the SAS data set SASHELP.CLASS.
The output is not shown here.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Options with the PRINT Statement in SAS/IML

```
 9  PROC IML;
10    USE sashelp.class;
11    READ all var _num_ INTO nummat  where(sex='M');
12    READ all var {name} INTO charmat where(sex='M');
13           cols = {'Age' 'Height' 'Weight'};
14    CLOSE sashelp.class;
15    PRINT nummat [rowname=charmat
16                  colname=cols
17                  label= ' '
18                  format=5.0];
19  QUIT;
```

Line 15-18: The SAS/IML PRINT statement has the following four useful options that control the display of a matrix.

ROWNAME=matrix specifies a character matrix used for row headings.
COLNAME=matrix specifies a character matrix used for column headings.
LABEL=label specifies a label for the matrix. If this option is not specified, the name of the matrix is used as a label.

FORMAT=format specifies a valid SAS or user-defined format to use when printing matrix values.

```
            Age Height Weight

Alfred       14     69    113
Henry        14     64    103
James        12     57     83
Jeffrey      13     63     84
John         12     59    100
Philip       16     72    150
Robert       12     65    128
Ronald       15     67    133
Thomas       11     58     85
William      15     67    112
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Creating a Matrix with Selected Variables and Observations from a SAS Data Set

```
28 PROC IML;
29    USE sashelp.class where(name=:'J');
30    READ all var {name sex} INTO class_J_mat;
31    CLOSE sashelp.class;
32    PRINT class_J_mat;
33 QUIT;
```

Line 29: The WHERE clause is used in the USE statement rather than READ statement.
Line 30: The variables of interest are specified in the VAR clause.

Reading Numeric and Character Variables Separately and Creating Numeric and Character Vectors

```
PROC IML;
  USE sashelp.class;
  READ all var _num_ INTO num_mat;
  READ all var _char_ INTO char_mat;
  CLOSE sashelp.class;
  PRINT num_mat, char_mat;
QUIT;
```

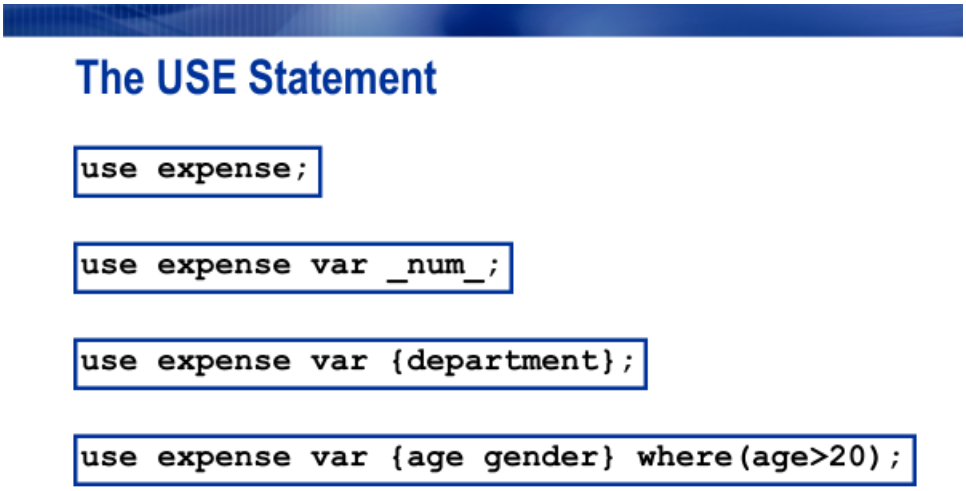
We have used the _NUM_ or _CHAR_ keyword in the VAR clause in conjunction with the INTO clause in two separate READ statements to read all numeric and all character variables (respectively) into two separate vectors.
With a comma (,) placed between two matrices, the PRINT statement prints two vectors one after the other vertically.
Note: Without the INTO clause, each variable in the VAR clause, becomes a column vector with the same name as the variable in the SAS data set. [SAS IML Documentation]

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Summary

An *open* data set is one that is ready for read and/or write access. You can use one of the following three statements to open a data set with PROC IML:

- USE allows read access to an existing data set.
- EDIT allows read and write access to an existing data set.
- CREATE creates a new data set for both read and write access.



Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
*Selecting 2 top and 2 bottom values;
data a;
input a @@;
infile datalines dlm = ' ';
datalines;
1 2 3 4 6 99 118 190 0 4 5 8
;

* Solution 1;
ODS select ExtremeObs;
proc univariate data=a;
run;



*Solution 2;
proc summary data=a;
output out=top
  idgroup(min(A) out[2](A)=);
output out=bot
 idgroup(max(A) out[2](A)=);
run;
data topbotsum(keep=fro A_:);
  retain fro;
  set top(in=t) bot;
  if t then fro='TOP';
  else fro='BOT';
run;
proc print data=topbotsum; run;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
*Solution 3;
proc iml;
use a;
   read all var _NUM_ into a;
close a;
call sort(a);
rows = nrow(a);
fin = (a[1:2, ])//
      (a[rows - 1:rows, ]);
print fin ;
create res
  var {fin};
append;
close res;
quit;
proc print data=res; run;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

XSECT and SDIF Functions in SAS/IML

"The XSECT function returns as a row vector the sorted set (without duplicates) of the element values that are present in all of its arguments. This set is the intersection of the sets of values in its argument matrices. When the intersection is empty, the XSECT function returns an empty matrix with zero rows and zero columns. There can be up to 15 arguments, which must all be either character or numeric." SAS® Documentation.
"The SETDIF function returns as a row vector the sorted set (without duplicates) of all element values present in *A* but not in *B*. If the resulting set is empty, the SETDIF function returns an empty matrix with zero rows and zero columns." SAS® Documentation.

```
 1 *Example_XSECT_SDIF.sas;
 2 proc iml;
 3 varNames_d1 = contents("sashelp","class");
 4 varNames_d2 = contents("sashelp","classfit");
 5 In = xsect(upcase(varNames_d1), upcase(varNames_d2));
 6 Out = setdif(upcase(varNames_d2), upcase(varNames_d1));
 7 print varNames_d1[label="Vars in Data Set D1"], '  ',
 8       varNames_d2 [label="Vars in Data Set D2"], '  ',
 9       In[label="Vars in Both Data Sets - D1 and D2"], '  ',
10       Out[label="Vars in Data Set D2 but Not in Data Set D1"];
11 quit;
```

```
                Vars in Data Set D1

                Name
                Sex
                Age
                Height
                Weight

                Vars in Data Set D2

                Name
                Sex
                Age
                Height
                Weight
                predict
                lowermean
                uppermean
                lower
                upper

          Vars in Both Data Sets - D1 and D2

          AGE    HEIGHT NAME    SEX    WEIGHT

      Vars in Data Set D2 but Not in Data Set D1

   LOWER     LOWERMEAN PREDICT   UPPER     UPPERMEAN
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

DATASETS Function in PROC IML

The DATASETS function returns the names of all SAS data sets in a specified libref

```
libname new 'D:\';
proc iml;
lib = "new";
A = datasets(lib);
First3 = A[1:3];
print First3;
quit;
```

CONTENTS Function in PROC IML

The CONTENTS function returns the names of all variables in a specified data set.

```
proc iml;
lib = "SASHELP";
ds = datasets(lib);

/*Find data sets that contain the same variable names*/

do i = 1 to nrow(ds);
   varnames = upcase( contents(lib, ds[i]) );
   if any(varnames = "WEIGHT") then
      print (ds[i]);
end;
quit;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Creating SAS Data Sets from SAS/IML Vectors

The CREATE and APPEND statements can be used to write a SAS data set from SAS/IML vectors.

```
 1 OPTIONS nocenter ps=58 ls=72 nodate nonumber
 2   FORMCHAR="|----|+|---+=|-/\<>*" ;
 3 *Example_create_SDS_from_vectors;
 4 proc iml;
 5 Calorie_intake = {2500,2000,2200,2400,3000, 2000};
 6 Excercise_minutes = {40,50,15,25,30,.A, .B};
 7 Day = {'Sun', 'Mon', 'Tue', 'Wed','Thu', 'Fri', 'Sat'};
 8 create data_from_vector
 9     var {Day Calorie_intake Excercise_minutes};
10 append;
11 list all;
12 close data_from_vector;
13 quit;
```

Line 5-7: The character vector creates a character variable, and the numeric vector creates a numeric variable.
Lines 8-9: The CREATE statement creates a new data set containing three variables corresponding to three vectors CALORIE_INTAKE, EXERCISE_MINUTES, and DAY. Each element of a vector (character or numeric) is written to an observation in the data set.
Line 10: The APPEND statement after the CREATE statement is required to add observations to the open data set created by the CREATE statement.
Line 11: The LIST statement with the ALL option is used to display all variables and observations from the data set that is open.

OBS	DAY	CALORIE_INTAKE	EXCERCISE_MINUTES
1	Sun	2500	40.0000
2	Mon	2000	50.0000
3	Tue	2200	15.0000
4	Wed	2400	25.0000
5	Thu	3000	30.0000
6	Fri	2000	A
7	Sat	.	B

Note the following:

1) The variable CALORIE_INTAKE is padded with a missing value for observation # 7 because it results from a shorter vector.

2) The variable EXERCISE_MINURES is also padded with a missing value for observations # 7 and 8 because it also results from a shorter vector.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Creating a SAS Data Set from a Matrix

```
 1  *Example_create_SDS_from_matrix.sas;
 2  proc iml;
 3  A = {11 22 33,
 4       44 55 66,
 5       77 88 99,
 6       33 22 21 };
 7  create data_from_matrix
 8     from A[colname={"Test1" "Test2" "Test3"}];
 9  append from A;
10  list all;
11  close data_from_matrix;
12  quit;
```

OBS	Test1	Test2	Test3
1	11.0000	22.0000	33.0000
2	44.0000	55.0000	66.0000
3	77.0000	88.0000	99.0000
4	33.0000	22.0000	21.0000

EDIT and LIST Statements with the WHERE Expression

```
 1   OPTIONS nofmterr FORMCHAR="|----|+|---+=|-/\<>*";
 2  *Example_USE_EDIT_Statements.SAS;
 3   %LET Path=C:\Users\Pradip Muhuri\SASClassGWU\TopicsByWeek\Week12;
 4  libname imlin "&Path";
 5   options fmtsearch=(imlin.formats);
 6  proc iml;
 7  reset deflib=imlin;
 8   use imlin.for_iml_pop2013 var {State_Name Pop}; show contents;
 9  close imlin.for_iml_pop2013;
10
11  edit imlin.for_iml_pop2013
12    var {State_FIPS State_Name Pop} where(State_FIPS <=8);
13  list all;
14
15  var_group = {State_Name Pop};
16  list all var var_group where(State_FIPS <=4);
17  show contents;
18  show datasets;
19  quit;
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Line 11: The EDIT statement open a data set and make it current for both input and output.

Line 13: The LIST statement is used to display all observations in the current SAS data set.

Line 17: This statement lists all open SAS data sets and their status.

Line 18: This statement displays the variable names and types, the size, and the number of observations in the current input data set.

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

POINT *operand* and VAR *operand*

```
 1 *Example_USE_LIST_POINT_VAR_WHERE.sas;
 2 OPTIONS nocenter ps=58 ls=72 nodate nonumber
 3   FORMCHAR="|----|+|---+=|-/\<>*" ;
 4 proc iml;
 5  use sashelp.class ;
 6  *list all;
 7  list point 3;
 8  list point {2 4};
 9  p= {1 3 5};
10  v={name height weight};
11  list point p var v;
12  list all var v where (weight >=150);
13 quit;
```

Line 7: the LIST statement is used to display (POINT) observation # 3 in the current SAS data set.

Line 9: P is a row vector of observations 1, 3, and 5.
Line 10: V is a vector of variable names.
Line 11: P is a POINT operand.
Line 12: V is a VAR operand.
Line 12: The expression in the WHERE clause is evaluated true or false.

```
  OBS Name     Sex       Age     Height    Weight
----- -------- --- --------- --------- ---------
    3 Barbara  F      13.0000   65.3000   98.0000

  OBS Name     Sex       Age     Height    Weight
----- -------- --- --------- --------- ---------
    2 Alice    F      13.0000   56.5000   84.0000
    4 Carol    F      14.0000   62.8000  102.5000

  OBS Name        Height    Weight
----- -------- --------- ---------
    1 Alfred     69.0000  112.5000
    3 Barbara    65.3000   98.0000
    5 Henry      63.5000  102.5000

  OBS Name        Height    Weight
----- -------- --------- ---------
   15 Philip     72.0000  150.0000
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
*Example_Summary.sas;
options nodate nonumber;
proc iml;
USE sashelp.heart;
 SUMMARY class {sex} var {AgeAtDeath  weight};
CLOSE;
quit;
```

The USE statement opens the SAS dataset SASHELP.HEART for read-access and sets it as the current dataset.

The SUMMARY statement computes statistics for numeric variables specified in the VAR clause for the current data set. The statistics are stratified by the variable SEX specified in the CLASS statement. The computed statistics are displayed in tabular form below.

Sex	Nobs	Variable	MIN	MAX	MEAN	STD
Female	2873	AgeAtDeath	36.00000	93.00000	71.56696	10.83126
		Weight	67.00000	300.00000	141.38864	26.28804
Male	2336	AgeAtDeath	36.00000	91.00000	69.69315	10.25983
		Weight	99.00000	276.00000	167.46615	25.29070
All	5209	AgeAtDeath	36.00000	93.00000	70.53641	10.55941
		Weight	67.00000	300.00000	153.08668	28.91543

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

Generating Summary Statistics

```
proc iml;
USE sashelp.heart;
summary class {sex} var {AgeAtDeath weight}
        stat {mean std var} opt {noprint save};
CLOSE;
show names;
print AgeAtDeath[r=sex c={"Mean" "Std"} format=5.1],
      weight[r=sex c={"Mean" "Std"} format=5.1];
quit;
```

The operand STAT computes the specified statistics (mean, standard deviation, and variance). The operand is a character matrix that contains the names of statistics.

In the operand OPT operand, the NOPRINT option suppresses the printing of the results from the SUMMARY statement. The SAVE option requests that the SUMMARY statement save the resultant statistics in matrices.

The options NAMES in the SHOW statement displays attributes matrices AgeAtDeath and Weight having values. The attributes include number of rows, number of columns, data type, and size.

The PRINT statement print the matrices AgeAtDeath and Weight.

```
SYMBOL            ROWS   COLS TYPE   SIZE
------          ------ ------ ---- ------
AgeAtDeath           2      3 num       8
Sex                  2      1 char      6
Weight               2      3 num       8
_NOBS_               2      1 num       8
 Number of symbols = 18  (includes those without values)
```

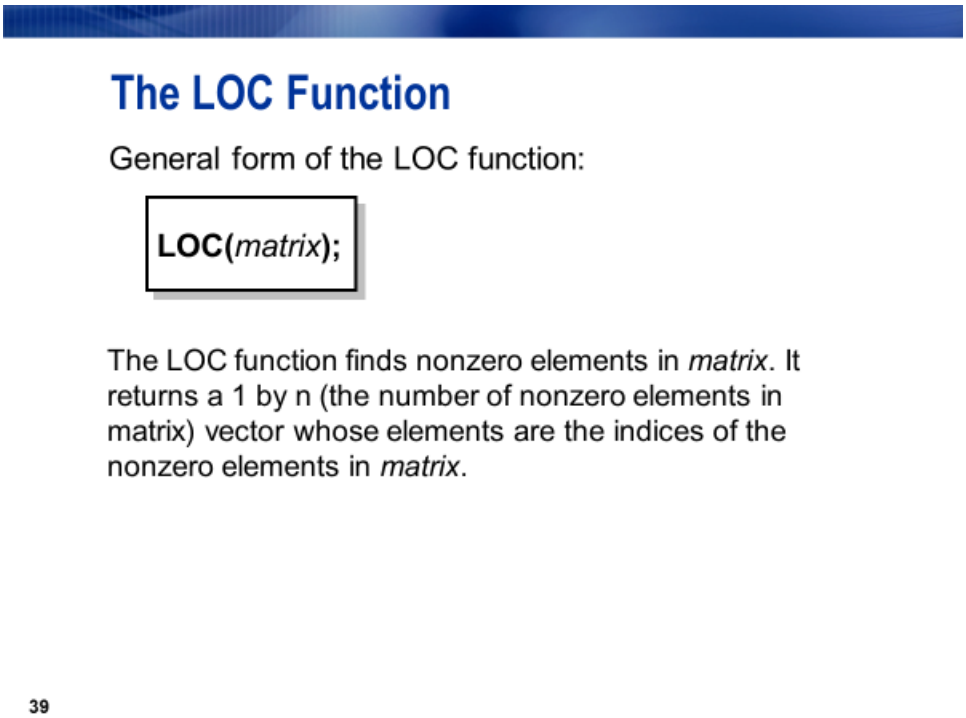
```
AgeAt
Death    Mean  Std

Female   71.6  10.8 117.3
Male     69.7  10.3 105.3
Weight   Mean  Std

Female  141.4  26.3 691.1
Male    167.5  25.3 639.6
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

The LOC Function

Use the LOC function to subset vectors or matrices.
Use this function to locate elements that satisfy a condition.
Get the positions of the elements in row-major order

- For vector, this is simply the position of the element.
- For matrices, you need to do some further manipulation to use the results as an index.



Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

The LOC Function

$$X = \begin{bmatrix} 1 & 2 & . \\ 4 & -5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Use the LOC function to locate elements that meet some condition:

`X[loc(X<0)]=0;` results in $X = \begin{bmatrix} 1 & 2 & 0 \\ 4 & 0 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

40

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
* Example_LOC.sas;
proc iml;
 sort Sashelp.demographics out=Sorted_countries
    by descending pop;
 varnames = {'Name', 'Pop'};
 use Sorted_countries;
 read all var varnames;
 close sashelp.demographics;
 idx = loc(pop>140000000);

 mean_world_pop1=pop[:];
 mean_world_pop2=mean(pop);
 mean_world_pop3=sum(pop)/nrow(pop);
 Number_of_countries=nrow(name);
 print (name[idx])
       (pop[idx])[format=comma15.];
  print
   Number_of_countries,
   mean_world_pop1 [format=comma15.],
   mean_world_pop2 [format=comma15.],
   mean_world_pop3 [format=comma15.];
 quit;
```

```
CHINA                                  1,323,344,591
INDIA                                  1,103,370,802
UNITED STATES                            298,212,895
INDONESIA                                222,781,487
BRAZIL                                   186,404,913
PAKISTAN                                 157,935,075
RUSSIA                                   143,201,572
BANGLADESH                               141,822,276
```

```
Number_of_countries

                197
mean_world_pop1

     33,870,294
mean_world_pop2

     33,870,294
mean_world_pop3

     33,870,294
```

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

CMISS Function in Base SAS vs. COUNTMISS Function in SAS/ IML

http://blogs.sas.com/content/iml/2012/04/02/count-missing-values-in-observations.html

```
 2  *Example_CMISS_Base_IML.sas;
 3  data Missing;
 4   input A B C;
 5   datalines;
 6   2 1 1
 7   4 . .
 8   1 3 1
 9   . 6 1
10   . 1 .
11   3 4 2
12   ;
13  data A;
14   set Missing;
15   array vars(3) A--C;
16   numMissing = cmiss(of vars[*]);
17   run;
18  proc print data=A; run;
```

Obs	A	B	C	num Missing
1	2	1	1	0
2	4	.	.	2
3	1	3	1	0
4	.	6	1	1
5	.	1	.	2
6	3	4	2	0

```
21  proc iml;
22   use Missing;
23     read all var _NUM_ into x;
24   close Missing;
25   rowMiss = countmiss(x, "ROW");
26   print (x || rowmiss);
27   jdx = loc(rowMiss>0);
28   print jdx;
29   idx = loc(rowMiss=0);
30   y = x[idx,];
31   print y;
32   quit;
```

2	1	1	0
4	.	.	2
1	3	1	0
.	6	1	1
.	1	.	2
3	4	2	0

jdx

2	4	5

y

2	1	1
1	3	1
3	4	2

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
*Example_LOC.sas;
proc sql outobs=8;
select name, pop
 from sashelp.demographics
 order by pop desc;
 quit;
```

GLC Country Name	Population (2005)
CHINA	1,323,344,591
INDIA	1,103,370,802
UNITED STATES	298,212,895
INDONESIA	222,781,487
BRAZIL	186,404,913
PAKISTAN	157,935,075
RUSSIA	143,201,572
BANGLADESH	141,822,276

```
proc sql;
  select (a.pop/b.pop) -1
format=percent7.2
          as India_over_China
          from sashelp.demographics a
          join sashelp.demographics b
         on a.name='INDIA'
         and b.name='CHINA';
quit;
```

India_over_China
(16.6%)

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

```
proc iml;
 use sashelp.demographics;
 read all var {name pop};
 close sashelp.demographics;
 pop_India_over_china=pop[loc(name='INDIA')]
          /pop[loc(name='CHINA')]-1;
 print
pop_India_over_china[format=percent7.2];
 quit;
```

pop_India_over_china
(16.6%)

```
proc iml;
  use sashelp.demographics;
 read all var {pop}
      into x[rowname=name colname=pop];
 close sashelp.demographics;
 pop_India_over_china=(x['India',
'pop']/x['China', 'pop'])-1;
 print pop_India_over_china[format=percent7.2];
 quit;
```

pop_India_over_china
(16.6%)

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.

https://blogs.sas.com/content/iml/2011/11/01/the-unique-loc-trick-a-real-treat.html

The UNIQUE-LOC trick: A real treat!

12

By Rick Wicklin on The DO Loop NOVEMBER 1, 2011

```
proc iml;
use sashelp.class; /* 1. Read in the data */
   read all var {sex} into C;    /* categorical variable, C */
   read all var {height} into x; /* numerical variable, x */
close;

u = unique(C);     /* 2. Unique values (levels) of categorical variabl
e. */
s = j(1, ncol(u)); /* 3. Allocate vector to hold results */
do i = 1 to ncol(u);   /* 4. For each level... */
   idx = loc(C=u[i]);  /* 5. Find observations in level */
   s[i] = mean(x[idx]);/* 6. Compute a statistic for those values */
end;
print s[colname=u]; /* assumes that u is character vector */
```

s	
F	M
60.588889	63.91

Acknowledgements: Texts are primarily adapted/obtained from SAS/IML Documentation, the SAS Blog (The DO LOOP by Rick Wicklin), papers presented by Rick Wicklin at SAS Global Forum, and other web sources. Some of the SAS/IML code-examples here are adapted from these sources. Please let me know of any errors or typos you have found. Do not circulate the handouts.