SAS Tutorial

Table of Contents

Before beginning this tutorial, you need to insure that your SAS environment is connected with an R environment so that the R code that SAS calls at the end of this tutorial from the IML Procedure run successfully.

In SAS,

```
*  This is a single line comment ;
/* This is a paragraph
   comment */
```

1 Reading in Data and Basic Statistical Functions

1.1 Read in the data.

The IMPORT Procedure is useful for reading in SAS data sets of a variety of different types.

a) Read the data in as a .csv file.

```
proc import out = student
  datafile = 'C:/Users/class.csv'
  dbms = csv replace;
  getnames = yes;
run;
```

b) Read the data in as a .xls file.

```
proc import out = student_xls
  datafile = 'C:/Users/class.xls'
  dbms = xls replace;
  getnames = yes;
run;
```

c) Read the data in as a .json file.

There is more code involved in reading a .json file into SAS so that all the format is correct, however we will not at this time dive into the explanation for all this code, but please see the links below.

```
data student_json;
  INFILE 'C:/Users/class.json' LRECL  = 3456677  TRUNCOVER SCANOVER
    dsd
    dlm=",}";
  INPUT
    @'"Name":' Name : $12.
    @'"Sex":' Sex : $2.
    @'"Age":' Age :
    @'"Height":' Height :
    @'"Weight":' Weight :
    @@;
run;
```

DATA step: infile & input statements

1.2 Find the dimensions of the data set.

The shape of a SAS data set is available by running the IMPORT Procedure and looking at the notes in the log file.

```
proc import out = student
  datafile = 'C:/Users/class.csv'
  dbms = csv replace;
```

```
  getnames = yes;
run;
```

```
NOTE: The infile 'C:/Users/class.csv' is:
      Filename=C:¥Users¥class.csv,
      RECFM=V,LRECL=32767,File Size (bytes)=411,
      Last Modified=2017年06月06日 11時46分38秒,
      Create Time=2017年06月06日 08時46分20秒

NOTE: 19 records were read from the infile 'C:/Users/class.csv'.
      The minimum record length was 17.
      The maximum record length was 21.
NOTE: The data set WORK.STUDENT has 19 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.11 seconds
      cpu time            0.10 seconds


NOTE: This SAS session is using a registry in WORK.  All changes will be lost at the end of this session.
19 rows created in WORK.STUDENT from C:/Users/class.csv.



NOTE: WORK.STUDENT data set was successfully created.
NOTE: The data set WORK.STUDENT has 19 observations and 5 variables.
NOTE: PROCEDURE IMPORT used (Total process time):
      real time           0.51 seconds
      cpu time            0.32 seconds
```

Output:

1.3 Find basic information about the data set.

The CONTENTS procedure prints information about a SAS data set.

```
proc contents data = student;
run;

                          The CONTENTS Procedure

Data Set Name          WORK.STUDENT                   Observations          19
Member Type            DATA                           Variables             5
Engine                 V9                             Indexes               0
Created                06/27/2017 14:36:37            Observation Length    32
Last Modified          06/27/2017 14:36:37            Deleted Observations  0
Protection                                            Compressed            NO
Data Set Type                                         Sorted                NO
Label
Data Representation    WINDOWS_64
Encoding               wlatin1  Western (Windows)

                     Engine/Host Dependent Information

Data Set Page Size            65536
Number of Data Set Pages      1
First Data Page               1
Max Obs per Page              2039
Obs in First Data Page        19
Number of Data Set Repairs    0
ExtendObsCounter              YES
Filename                      C:\Users\ElainePC\AppData\Local\Temp\SAS
```

```
                            Temporary
                            Files\_TD8628_ELAINEHP_\student.sas7bdat
Release Created             9.0401M4
Host Created                X64_10PRO
Owner Name                  ElaineHP\ElainePC
File Size                   128KB
File Size (bytes)           131072

                Alphabetic List of Variables and Attributes

            #    Variable    Type    Len    Format      Informat

            3    Age         Num       8    BEST12.     BEST32.
            4    Height      Num       8    BEST12.     BEST32.
            1    Name        Char      7    $7.         $7.
            2    Sex         Char      1    $1.         $1.
            5    Weight      Num       8    BEST12.     BEST32.
```

1.4 Look at the first 5 observations.

The PRINT procedure prints a SAS data set, according to the specifications and options provided.

```
/* obs= option tells SAS how many observations to print, starting
   with the first observation */
proc print data = student (obs=5);
run;

   Obs    Name       Sex               Age          Height          Weight

     1    Alfred      M                 14              69           112.5
     2    Alice       F                 13            56.5              84
     3    Barbara     F                 13            65.3              98
     4    Carol       F                 14            62.8           102.5
     5    Henry       M                 14            63.5           102.5
```

1.5 Calculate mean of numeric variables.

The MEANS procedure prints the mean of all numeric variables of a SAS data set, as well as other descriptive statistics.

```
proc means data = student mean;
run;
                            The MEANS Procedure

                         Variable            Mean
                         ------------------------
                         Age           13.3157895
                         Height        62.3368421
```

```
                         Weight       100.0263158
                         ------------------------
```

1.6 Compute summary statistics of the data set.

Summary statistics of a SAS data set are available by running the MEANS procedure and specifying statistics to return.

```
/* SAS uses a different method than Python and R to compute
   quartiles, but the method in each language can be changed */
/* maxdec= option tells SAS to print at most 2 numbers behind
   the decimal point */
proc means data = student min q1 median mean q3 max n maxdec=2;
run;

                            The MEANS Procedure

                                        Lower
 Variable         Minimum           Quartile           Median             Mean
 -------------------------------------------------------------------------
 Age                11.00              12.00            13.00            13.32
 Height             51.30              57.50            62.80            62.34
 Weight             50.50              84.00            99.50           100.03
 -------------------------------------------------------------------------

                                     Upper
              Variable            Quartile          Maximum      N
              ----------------------------------------------
              Age                    15.00            16.00     19
              Height                 66.50            72.00     19
              Weight                112.50           150.00     19
              ----------------------------------------------
```

1.7 Descriptive statistics functions applied to columns of the data set.

```
/* The var statement tells SAS which variable to use for the
   procedure */
proc means data = student stddev sum n max min median maxdec=2;
  var Weight;
run;

                            The MEANS Procedure

                        Analysis Variable : Weight

      Std Dev              Sum    N          Maximum          Minimum          Median
 -------------------------------------------------------------------------
        22.77          1900.50   19           150.00            50.50           99.50
 -------------------------------------------------------------------------
```

1.8 Produce a one-way table to describe the frequency of a variable.

The FREQ procedure prints the frequency of categorical or discrete variables of a SAS data set.

a) Produce a one-way table of a discrete variable.

```
proc freq data = student;
  tables Age / nopercent norow nocol;
run;

                            The FREQ Procedure

                                                Cumulative
                      Age     Frequency      Frequency
                      ------------------------------
                       11            2               2
                       12            5               7
                       13            3              10
                       14            4              14
                       15            4              18
                       16            1              19
```

b) Produce a one-way table of a categorical variable.

```
proc freq data = student;
  tables Sex / nopercent norow nocol;
run;

                            The FREQ Procedure

                                                Cumulative
                      Sex     Frequency      Frequency
                      ------------------------------
                      F              9               9
                      M             10              19
```

The tables statement allows you to specify multiple variables at once, separated only by a space, so both of these tables could have been created with one FREQ procedure call. The options on the tables statement (nopercent norow nocol) prevent SAS from printing percents in the table, which are printed by default.

TRY THIS AT HOME: Run this procedure without the options on the tables statment.

1.9 Produce a two-way table to visualize the frequency of two categorical (or discrete) variables.

```
/* The "*" between two variables on the tables statement
   indicates to produce a two-way table of the two variables */
proc freq data = student;
  tables Age*Sex / nopercent norow nocol;
run;
```

```
                         The FREQ Procedure

                         Table of Age by Sex

                Age       Sex

                Frequency|F       |M       |  Total
                ---------+--------+--------+
                      11 |      1 |      1 |      2
                ---------+--------+--------+
                      12 |      2 |      3 |      5
                ---------+--------+--------+
                      13 |      2 |      1 |      3
                ---------+--------+--------+
                      14 |      2 |      2 |      4
                ---------+--------+--------+
                      15 |      2 |      2 |      4
                ---------+--------+--------+
                      16 |      0 |      1 |      1
                ---------+--------+--------+
                Total           9       10       19
```

FREQ Procedure

1.10 Select a subset of the data that meets a certain criterion.

The SAS DATA step is used for all things data manipulation and in Section 2 we will explore it further.

```
data females;
  set student;
  where Sex = "F";
run;
proc print data = females(obs=5);
run;

   Obs    Name       Sex         Age        Height       Weight

    1     Alice       F           13         56.5           84
    2     Barbara     F           13         65.3           98
    3     Carol       F           14         62.8        102.5
    4     Jane        F           12         59.8         84.5
    5     Janet       F           15         62.5        112.5
```

DATA step: set & where statements

TRY THIS AT HOME: Run this procedure to return all *male* students.

1.11 Determine the correlation between two continuous variables.

```
/* The nosimple option reduces the output of this procedure */
proc corr data = student pearson nosimple;
var Height Weight;
run;
                            The CORR Procedure

                     2  Variables:    Height   Weight

                 Pearson Correlation Coefficients, N = 19
                        Prob > |r| under H0: Rho=0

                                  Height        Weight

                    Height       1.00000       0.87779
                                                <.0001

                    Weight       0.87779       1.00000
                                  <.0001
```

CORR Procedure

TRY THIS AT HOME: Run this procedure and do not reduce the output.

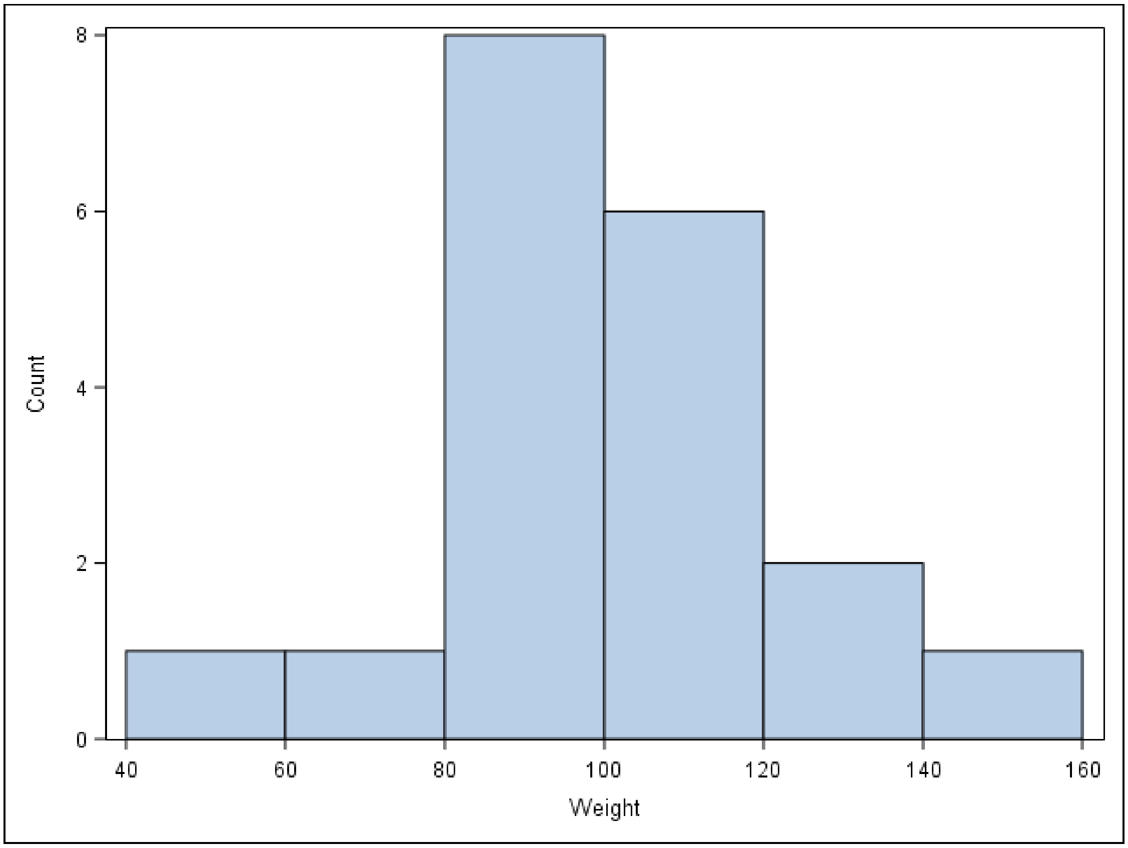
---

2 Basic Graphing and Plotting Functions

The SGPLOT procedure is a very useful SAS procedure for producing plots from data. For more information on other statements within the SGPLOT procedure, please see the Appendix Section 2.

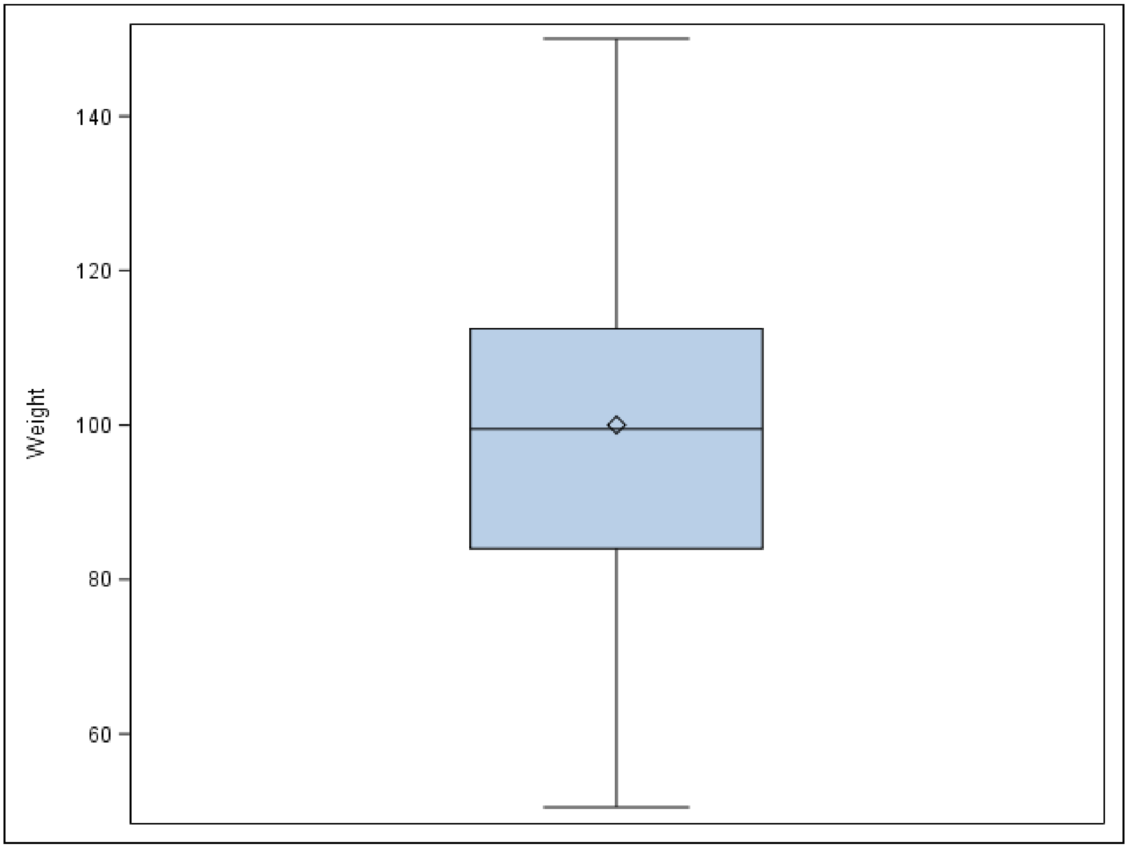
2.1 Visualize a single continuous variable by producing a histogram.

```
/* Notice the specifcation of the bins, as well as the xaxis values */
/* SAS denotes "count" as what R & Python denote as "frequency" */
proc sgplot data = student;
  histogram weight / binwidth=20 binstart=40 scale=count;
  xaxis values=(40 to 160 by 20);
run;
```

Output: 

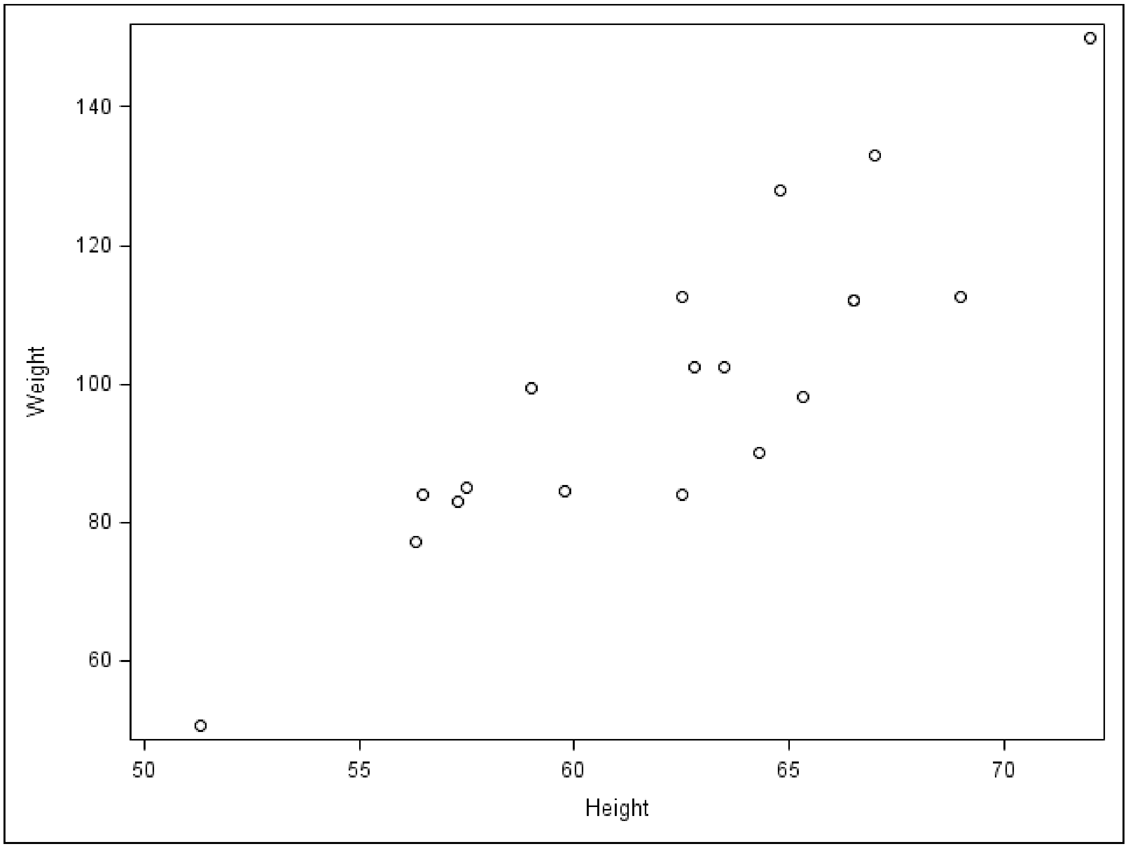
2.2 Visualize a single continuous variable by producing a boxplot.

```
/* SAS automatically prints the mean on the boxplot */
proc sgplot data = student;
  vbox Weight;
run;
```

Output: 

2.3 Visualize two continuous variables by producing a scatterplot.

```
/* Notice here you specify the y variable followed by the x variable */
proc sgscatter data = student;
  plot Weight * Height;
run;
```

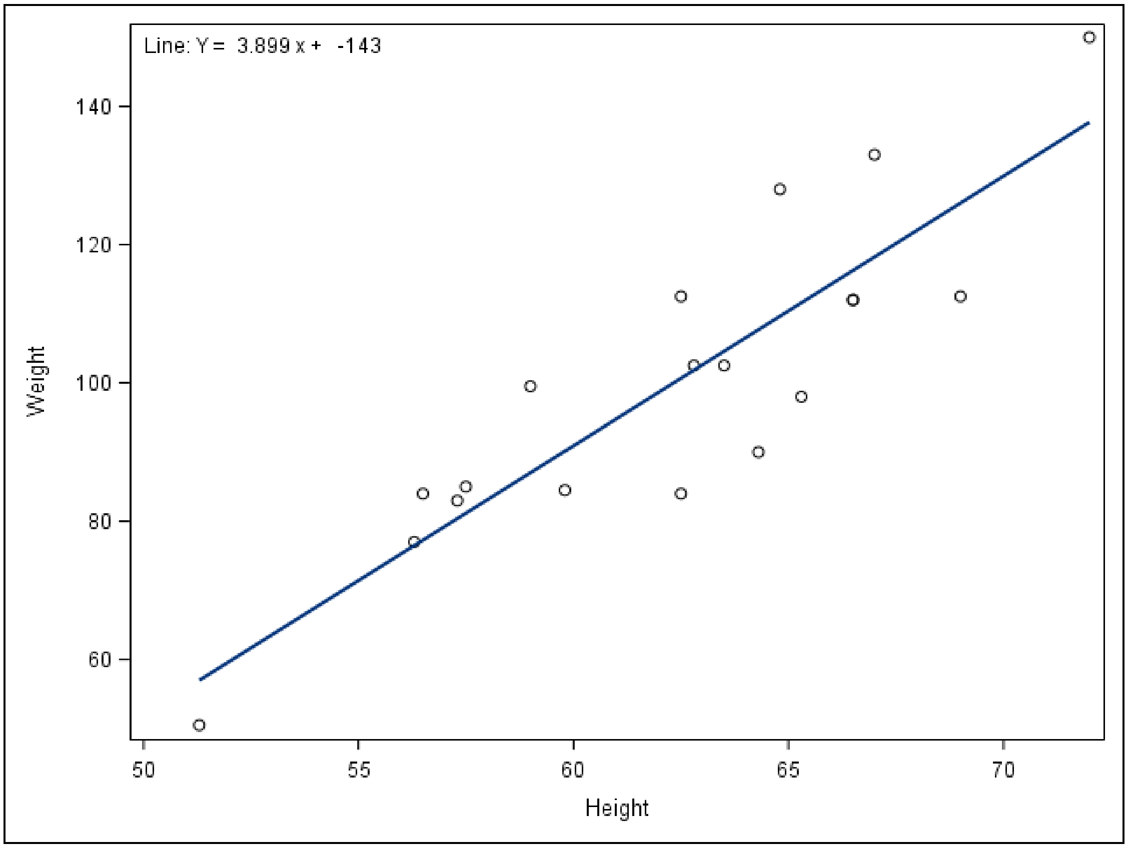
Output: 

SGSCATTER Procedure

2.4 Visualize a relationship between two continuous variables by producing a scatterplot and a plotted line of best fit.

```
/* Use proc reg to get the parameter estimates for the line of best fit,
   but don't print the graph (ods graphics off) */
ods graphics off;
proc reg data = student;
  /* Syntax indicates Weight as a function of Height */
  model Weight = Height;
  ods output ParameterEstimates=PE;
run;
ods graphics on;

/* data _null_ indicates to not create a data set, but
   run the code within the data step to create macro
   variables to store the parameter estimates */
data _null_;
  set PE;
  if _n_=1 then call symput('Int', put(estimate, BEST6.));
  else call symput('Slope', put(estimate, BEST6.));
run;
```

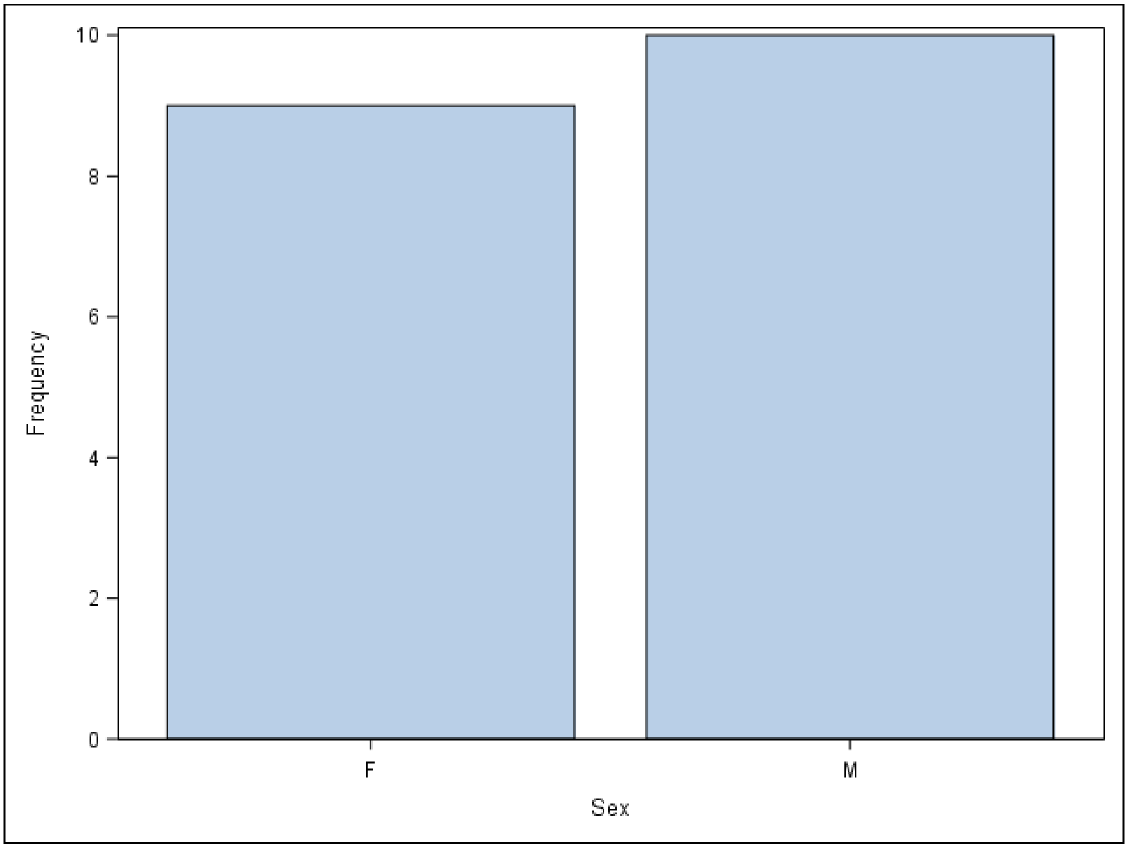
```
/* Use proc sgplot with the reg statement so it prints the line of best fit,
   and use the inset statement to print the equation of the line
   of best fit */
proc sgplot data = student noautolegend;
  reg y = Weight x = Height;
  inset "Line: Y = &Slope x + &Int" / position=topleft;
run;
```

Output: 

REG Procedure | set statement | macro variables | call symput() function

2.5 Visualize a categorical variable by producing a bar chart.

```
/* Notice here you must first sort by Sex and then plot the vertical
  bar chart */
proc sort data = student;
  by Sex;
run;
proc sgplot data = student;
  vbar Sex;
run;
```

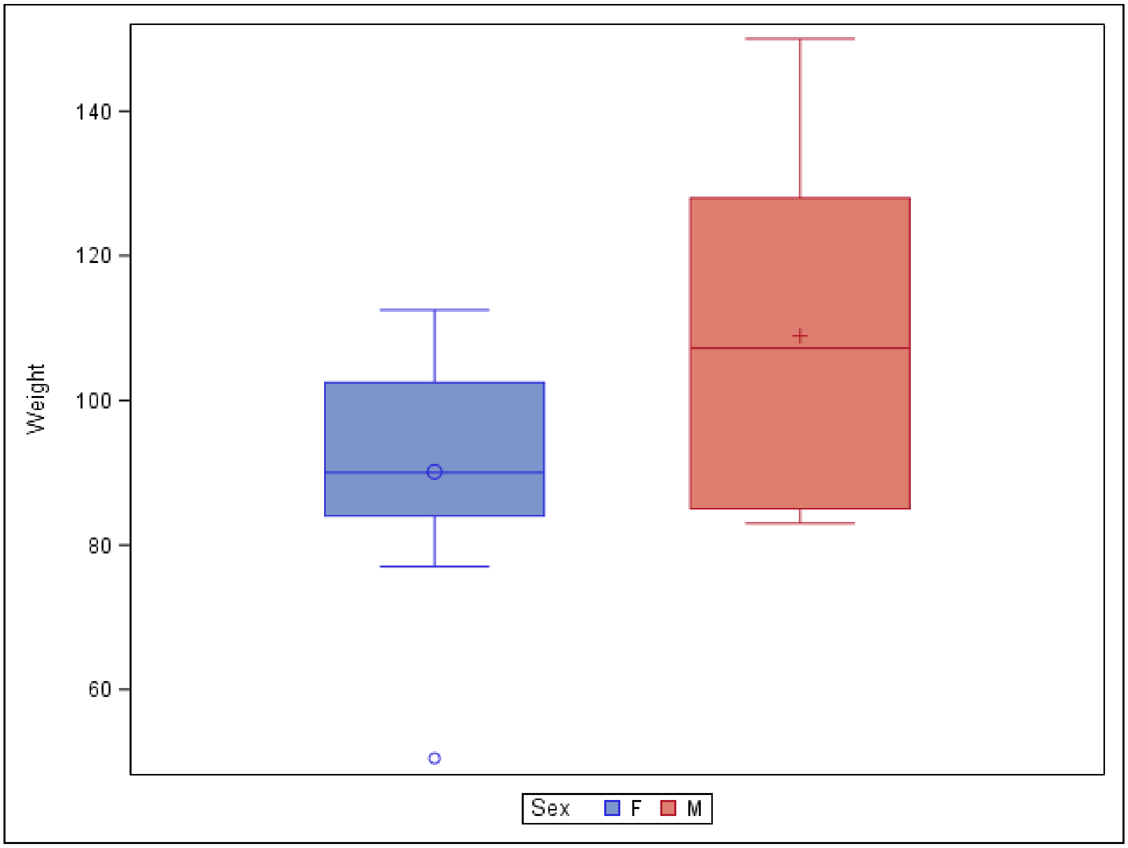
Output: 

SORT Procedure

2.6 Visualize a continuous variable, grouped by a categorical variable, using side-by-side boxplots.

More advanced side-by-side boxplot with color.

```
proc sgplot data = student;
  vbox Weight / group=Sex;
run;
```

Output: 

---

3 Basic Data Wrangling and Manipulation

Many of the following examples make use of the SAS DATA step for manipulating and altering data sets, and a main part of the DATA step is the set statement.

3.1 Create a new variable in a data set as a function of existing variables in the data set.

```
data student;
  set student;
  BMI = Weight / (Height**2) * 703;
run;
proc print data = student(obs=5);
run;

Obs    Name       Sex             Age          Height          Weight      BMI

  1    Alfred      M               14              69           112.5    16.6115
  2    Alice       F               13            56.5              84    18.4986
  3    Barbara     F               13            65.3              98    16.1568
```

```
  4   Carol       F               14           62.8          102.5   18.2709
  5   Henry       M               14           63.5          102.5   17.8703
```

3.2 Create a new variable in a data set using if/else logic of existing variables in the data set.

```
data student;
  set student;
  if (BMI < 19.0) then BMI_class = "Underweight";
  else BMI_class = "Healthy";
run;
proc print data = student(obs=5);
run;

Obs Name     Sex           Age        Height       Weight   BMI     BMI_class

  1 Alfred    M             14            69        112.5 16.6115 Underweight
  2 Alice     F             13          56.5           84 18.4986 Underweight
  3 Barbara   F             13          65.3           98 16.1568 Underweight
  4 Carol     F             14          62.8        102.5 18.2709 Underweight
  5 Henry     M             14          63.5        102.5 17.8703 Underweight
```

if-then/else statement

3.3 Create a new variable in a data set using mathemtical functions applied to existing variables in the data set.

Using the log(), exp(), sqrt(), & abs() functions.

```
data student;
  set student;
  LogWeight = log(Weight);
  ExpAge = exp(Age);
  SqrtHeight = sqrt(Height);
  if (BMI < 19.0) then BMI_Neg = -BMI;
  else BMI_Neg = BMI;
  BMI_Pos = abs(BMI_Neg);
  /* Create a boolean variable, which is handled differently
     in SAS than in Python and R */
  BMI_Check = (BMI_Pos = BMI);
run;
proc print data = student(obs=5);
run;


Obs Name     Sex           Age        Height       Weight   BMI     BMI_class

  1 Alfred    M             14            69        112.5 16.6115 Underweight
  2 Alice     F             13          56.5           84 18.4986 Underweight
  3 Barbara   F             13          65.3           98 16.1568 Underweight
```

```
  4 Carol     F            14          62.8         102.5 18.2709 Underweight
  5 Henry     M            14          63.5         102.5 17.8703 Underweight

      Log                     Sqrt                             BMI_
Obs  Weight       ExpAge      Height     BMI_Neg    BMI_Pos    Check

  1 4.72295    1202604.28    8.30662    -16.6115    16.6115      1
  2 4.43082     442413.39    7.51665    -18.4986    18.4986      1
  3 4.58497     442413.39    8.08084    -16.1568    16.1568      1
  4 4.62986    1202604.28    7.92465    -18.2709    18.2709      1
  5 4.62986    1202604.28    7.96869    -17.8703    17.8703      1
```

if-then/else statement

3.4 Drop variables from a data set.

```
data student;
  set student (drop = LogWeight ExpAge SqrtHeight BMI_Neg BMI_Pos BMI_Check);
run;
proc print data = student(obs=5);
run;

Obs Name     Sex           Age        Height       Weight    BMI     BMI_class

  1 Alfred    M            14            69         112.5 16.6115 Underweight
  2 Alice     F            13          56.5            84 18.4986 Underweight
  3 Barbara   F            13          65.3            98 16.1568 Underweight
  4 Carol     F            14          62.8         102.5 18.2709 Underweight
  5 Henry     M            14          63.5         102.5 17.8703 Underweight
```

drop= data set option

3.5 Sort a data set by a variable.

a) Sort data set by a continuous variable.

```
proc sort data = student;
  by Age;
run;
proc print data = student(obs=5);
run;

Obs Name     Sex           Age        Height       Weight    BMI    BMI_class

  1 Joyce     F            11          51.3          50.5 13.4900 Underweight
  2 Thomas    M            11          57.5            85 18.0733 Underweight
  3 James     M            12          57.3            83 17.7715 Underweight
  4 Jane      F            12          59.8          84.5 16.6115 Underweight
  5 John      M            12            59          99.5 20.0944 Healthy
```

b) Sort data set by a categorical variable.

```
proc sort data = student;
  by Sex;
run;
/* Notice that the data is now sorted first by Sex and
   then within Sex by Age */
proc print data = student(obs=5);
run;

Obs Name     Sex          Age       Height       Weight    BMI     BMI_class

  1 Joyce     F            11         51.3         50.5 13.4900 Underweight
  2 Jane      F            12         59.8         84.5 16.6115 Underweight
  3 Louise    F            12         56.3           77 17.0777 Underweight
  4 Alice     F            13         56.5           84 18.4986 Underweight
  5 Barbara   F            13         65.3           98 16.1568 Underweight
```

SORT Procedure

3.6 Compute descriptive statistics of continuous variables, grouped by a categorical variable.

```
proc means data = student mean;
  by Sex;
  var Age Height Weight BMI;
run;

---------------------------------- Sex=F ----------------------------------

                            The MEANS Procedure

                         Variable           Mean
                         ------------------------
                         Age          13.2222222
                         Height       60.5888889
                         Weight       90.1111111
                         BMI          17.0510391
                         ------------------------



---------------------------------- Sex=M ----------------------------------

                         Variable           Mean
                         ------------------------
                         Age          13.4000000
                         Height       63.9100000
                         Weight      108.9500000
                         BMI          18.5942434
                         ------------------------
```

MEANS Procedure

3.7 Add a new row to the bottom of a data set.

```
/* Look at the tail of the data currently */
proc print data = student(firstobs=15);
run;

Obs Name     Sex            Age        Height       Weight    BMI    BMI_class

 15 Alfred    M              14            69        112.5 16.6115 Underweight
 16 Henry     M              14          63.5        102.5 17.8703 Underweight
 17 Ronald    M              15            67          133 20.8285 Healthy
 18 William   M              15          66.5          112 17.8045 Underweight
 19 Philip    M              16            72          150 20.3414 Healthy

data student;
  set student end = eof;
  output;
  if eof then do;
    Name = 'Jane';
    Sex = 'F';
    Age = 14;
    Height = 56.3;
    Weight = 77.0;
    BMI = 17.077695;
    BMI_Class = 'Underweight';
    output;
  end;
run;
proc print data = student(firstobs=16);
run;

Obs Name     Sex            Age        Height       Weight    BMI    BMI_class

 16 Henry     M              14          63.5        102.5 17.8703 Underweight
 17 Ronald    M              15            67          133 20.8285 Healthy
 18 William   M              15          66.5          112 17.8045 Underweight
 19 Philip    M              16            72          150 20.3414 Healthy
 20 Jane      F              14          56.3           77 17.0777 Underweight
```

if-then/else & output statements | do loop, end= & firstobs= data set options

3.8 Create a user-defined function and apply it to a variable in the data set to create a new variable in the data set.

```
proc fcmp outlib=sasuser.userfuncs.myfunc;
  function toKG(lb);
    kg = 0.45359237 * lb;
  return(kg);
endsub;
```

```
options cmplib=sasuser.userfuncs;

data studentKG;
  set student;
  Weight_KG = toKG(Weight);
run;

proc print data = studentKG(obs=5);
run;


   Obs     Name        Sex              Age          Height          Weight

     1     Joyce        F                11            51.3            50.5
     2     Jane         F                12            59.8            84.5
     3     Louise       F                12            56.3              77
     4     Alice        F                13            56.5              84
     5     Barbara      F                13            65.3              98

                                      Weight_
   Obs       BMI        BMI_class        KG

     1     13.4900    Underweight    22.9064
     2     16.6115    Underweight    38.3286
     3     17.0777    Underweight    34.9266
     4     18.4986    Underweight    38.1018
     5     16.1568    Underweight    44.4521
```

FCMP Procedure

---

4 More Advanced Data Wrangling

4.1 Drop observations with missing information.

```
/* Notice the use of the fish data set because it has some missing
   observations */
proc import out = fish
  datafile='C:/Users/fish.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* First sort by Weight, requesting those with NA for Weight first,
   which SAS does automatically */
proc sort data = fish;
  by Weight;
```

```
run;
proc print data = fish(obs=5);
run;

      Obs     Species           Weight         Length1         Length2

        1      Bream                 .           29.5              32
        2      Roach                 0             19            20.5
        3      Perch               5.9            7.5             8.4
        4      Smelt               6.7            9.3             9.8
        5      Smelt                 7           10.1            10.6

      Obs          Length3           Height           Width

        1             37.3          13.9129          5.0728
        2             22.8           6.4752          3.3516
        3              8.8            2.112           1.408
        4             10.8           1.7388          1.0476
        5             11.6           1.7284          1.1484

data new_fish;
  set fish;
  /* Notice the not-equal operator (^=) and how SAS denotes
     missing values (.) */
  if (Weight ^= .);
run;
proc print data = new_fish(obs=5);
run;

      Obs     Species           Weight         Length1         Length2

        1      Roach                 0             19            20.5
        2      Perch               5.9            7.5             8.4
        3      Smelt               6.7            9.3             9.8
        4      Smelt                 7           10.1            10.6
        5      Smelt               7.5             10            10.5

      Obs          Length3           Height           Width

        1             22.8           6.4752          3.3516
        2              8.8            2.112           1.408
        3             10.8           1.7388          1.0476
        4             11.6           1.7284          1.1484
        5             11.6            1.972            1.16
```

SORT Procedure | if-then/else statement

4.2 Merge two data sets together on a common variable.

a) First, select specific columns of a data set to create two smaller data sets.

```
/* Notice the use of the student data set again, however we want to reload it
   without the changes we've made previously  */
proc import out = student
  datafile = 'C:/Users/class.csv'
  dbms = csv replace;
  getnames = yes;
run;
data student1;
  set student(keep= Name Sex Age);
run;
proc print data = student1(obs=5);
run;

                   Obs     Name       Sex               Age

                     1     Alfred      M                 14
                     2     Alice       F                 13
                     3     Barbara     F                 13
                     4     Carol       F                 14
                     5     Henry       M                 14

data student2;
  set student(keep= Name Height Weight);
run;
proc print data = student2(obs=5);
run;

              Obs     Name                Height           Weight

                1     Alfred                  69            112.5
                2     Alice                 56.5               84
                3     Barbara               65.3               98
                4     Carol                 62.8            102.5
                5     Henry                 63.5            102.5
```

keep= data set option

b) Second, we want to merge the two smaller data sets on the common variable.

```
data new;
  merge student1 student2;
  by Name;
run;
proc print data = new(obs=5);
run;
```

```
 Obs    Name       Sex         Age      Height      Weight

   1    Alfred      M           14        69         112.5
   2    Alice       F           13        56.5        84
   3    Barbara     F           13        65.3        98
   4    Carol       F           14        62.8       102.5
   5    Henry       M           14        63.5       102.5
```

DATA step: merge statement

c) Finally, we want to check to see if the merged data set is the same as the original data set.

```
proc compare base = student compare = new brief;
run;

                          The COMPARE Procedure
                Comparison of WORK.STUDENT with WORK.NEW
                             (Method=EXACT)

NOTE: No unequal values were found. All values compared are exactly equal.
```

COMPARE Procedure

4.3 Merge two data sets together by index number only.

a) First, select specific columns of a data set to create two smaller data sets.

```
data newstudent1;
  set student(keep= Name Sex Age);
run;
proc print data = newstudent1(obs=5);
run;

                   Obs    Name       Sex         Age

                     1    Alfred      M           14
                     2    Alice       F           13
                     3    Barbara     F           13
                     4    Carol       F           14
                     5    Henry       M           14

data newstudent2;
  set student(keep= Height Weight);
run;
proc print data = newstudent2(obs=5);
run;

                    Obs      Height      Weight

                      1        69         112.5
                      2        56.5        84
```

```
                        3            65.3             98
                        4            62.8          102.5
                        5            63.5          102.5
```

keep= data set option

b) Second, we want to join the two smaller data sets.

```
data new2;
  merge newstudent1 newstudent2;
run;
proc print data = new2(obs=5);
run;

   Obs     Name        Sex              Age          Height          Weight

     1     Alfred       M                14              69           112.5
     2     Alice        F                13            56.5              84
     3     Barbara      F                13            65.3              98
     4     Carol        F                14            62.8           102.5
     5     Henry        M                14            63.5           102.5
```

merge statement

c) Finally, we want to check to see if the joined data set is the same as the original data set.

```
proc compare base = student compare = new2 brief;
run;

                           The COMPARE Procedure
                 Comparison of WORK.STUDENT with WORK.NEW2
                              (Method=EXACT)

NOTE: No unequal values were found. All values compared are exactly equal.
```

COMPARE Procedure

4.4 Create a pivot table to summarize information about a data set.

```
/* Notice we are using a new data set that needs to be read into the
   environment */
proc import out = price
  datafile = 'C:/Users/price.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* The following code is used to remove the "," and "$" characters from the
   ACTUAL column so that values can be summed */
data price;
  set price;
  num_actual = input(actual, dollar10.);
```

```
run;

proc sql;
  create table categorysales as
    select country, state, prodtype,
    product, sum(num_actual) as REVENUE
    from price
  group by country, state, prodtype, product;
quit;
proc print data = categorysales(obs=5);
run;

     Obs    COUNTRY    STATE          PRODTYPE     PRODUCT     REVENUE

       1    Canada     British Co     FURNITURE     BED        197706.6
       2    Canada     British Co     FURNITURE     SOFA       216282.6
       3    Canada     British Co     OFFICE        CHAI       200905.2
       4    Canada     British Co     OFFICE        DESK       186262.2
       5    Canada     Ontario        FURNITURE     BED        194493.6
```

input() function | SQL Procedure

4.5 Return all unique values from a text variable.

```
proc iml;
  use price;
    read all var {STATE};
  close price;

  unique_states = unique(STATE);
  print(unique_states);
quit;

                              unique_states
       COL1       COL2       COL3       COL4       COL5       COL6

  ROW1 Baja Calif British Co California Campeche   Colorado   Florida

                              unique_states
       COL7       COL8       COL9       COL10      COL11      COL12

  ROW1 Illinois   Michoacan  New York   North Caro Nuevo Leon Ontario

                              unique_states
                  COL13      COL14      COL15      COL16

             ROW1 Quebec     Saskatchew Texas      Washington
```

IML Procedure

5 Preparation & Basic Regression

5.1 Pre-process a data set using principal component analysis.

```
/* Notice we are using a new data set that needs to be read into the
   environment */
proc import out = iris
  datafile = 'C:/Users/iris.csv'
  dbms = csv replace;
  getnames = yes;
run;

data features;
  set iris(drop=Target);
run;

proc princomp data = features noprint outstat = feat_princomp;
  var SepalLength SepalWidth PetalLength PetalWidth;
run;

data eigenvectors;
    set feat_princomp;
    where _TYPE_ = "SCORE";
run;
proc print data = eigenvectors;
run;

                              Sepal        Sepal       Petal        Petal
  Obs     _TYPE_     _NAME_      Length       Width      Length       Width

   1      SCORE      Prin1       0.52237    -0.26335    0.58125      0.56561
   2      SCORE      Prin2       0.37232     0.92556    0.02109      0.06542
   3      SCORE      Prin3      -0.72102     0.24203    0.14089      0.63380
   4      SCORE      Prin4      -0.26200     0.12413    0.80115     -0.52355
```

drop= data set option | PRINCOMP Procedure

5.2 Split data into training and testing data and export as a .csv file.

```
/* outall option tells SAS to add a flag showing which observations were
   chosen */
/* seed = 29 specifies the seed for random values so the results are
   reproducible */
proc surveyselect data = iris outall out = all method = srs samprate = 0.7
                              seed = 29;
run;

data train (drop = selected);
    set all;
    where (selected = 1);
run;
```

```
data test (drop = selected);
    set all;
    where (selected = 0);
run;

proc export data = train
   outfile = 'C:\Users\iris_train.csv'
   dbms = csv;
run;
proc export data = test
   outfile = 'C:\Users\iris_test.csv'
   dbms = csv;
run;
```

SURVEYSELECT Procedure | drop= data set option | EXPORT Procedure

5.3 Fit a logistic regression model.

```
/* Notice we are using a new data set that needs to be read into the
   environment */
proc import out = tips
  datafile = 'C:/Users/tips.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* The following code is used to determine if the individual left more than
   a 15% tip */
data tips;
  set tips;
  if (tip > 0.15*total_bill) then greater15 = 1;
  else greater15 = 0;
run;

/* The descending option tells SAS to model the probability that
  greater15 = 1 */
proc genmod data=tips descending;
  model greater15 = total_bill / dist = bin link = logit lrci;
run;
```

```
                           The GENMOD Procedure

                             Model Information

                      Data Set                WORK.TIPS
                      Distribution             Binomial
                      Link Function               Logit
                      Dependent Variable      greater15

                  Number of Observations Read         244
```

Number of Observations Used	244
Number of Events	135
Number of Trials	244

Response Profile

Ordered Value	greater15	Total Frequency
1	1	135
2	0	109

PROC GENMOD is modeling the probability that greater15='1'.

Criteria For Assessing Goodness Of Fit

Criterion	DF	Value	Value/DF
Log Likelihood		-156.8714	
Full Log Likelihood		-156.8714	
AIC (smaller is better)		317.7428	
AICC (smaller is better)		317.7926	
BIC (smaller is better)		324.7371	

Algorithm converged.

Analysis Of Maximum Likelihood Parameter Estimates

Parameter	DF	Estimate	Standard Error	Likelihood Ratio 95% Confidence Limits		Wald Chi-Square
Intercept	1	1.6477	0.3547	0.9722	2.3667	21.58
total_bill	1	-0.0725	0.0168	-0.1069	-0.0408	18.65
Scale	0	1.0000	0.0000	1.0000	1.0000	

Analysis Of Maximum Likelihood Parameter Estimates

Parameter	Pr > ChiSq
Intercept	<.0001
total_bill	<.0001
Scale	

NOTE: The scale parameter was held fixed.

if-then/else statement | GENMOD Procedure

5.4 Fit a linear regression model.

```
/* Fit a linear regression model of tip by total_bill */
proc reg data = tips outest=RegOut;
    tip_hat: model tip = total_bill;
quit;

                              The REG Procedure
                               Model: tip_hat
                          Dependent Variable: tip

                  Number of Observations Read         244
                  Number of Observations Used         244

                            Analysis of Variance

                                    Sum of           Mean
 Source                   DF       Squares         Square    F Value    Pr > F

 Model                     1     212.42373      212.42373     203.36    <.0001
 Error                   242     252.78874        1.04458
 Corrected Total         243     465.21248

           Root MSE              1.02205    R-Square     0.4566
           Dependent Mean        2.99828    Adj R-Sq     0.4544
           Coeff Var            34.08782

                             Parameter Estimates

                              Parameter       Standard
   Variable       DF          Estimate          Error     t Value     Pr > |t|

   Intercept       1           0.92027        0.15973        5.76       <.0001
   total_bill      1           0.10502        0.00736       14.26       <.0001
```

REG Procedure

6 Regression & Machine Learning: Modeling & Prediction

6.1 Fit a logistic regression model on training data and assess against testing data.

a) Fit a logistic regression model on training data.

```
/* Notice we are using new data sets that need to be read into the
   environment */
proc import out = train
```

```
  datafile = 'C:/Users/tips_train.csv'
  dbms = csv replace;
  getnames = yes;
run;
proc import out = test
  datafile = 'C:/Users/tips_test.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* The following code is used to determine if the individual left more than
   a 15% tip */
data train;
  set train;
  if (tip > 0.15*total_bill) then greater15 = 1;
  else greater15 = 0;
run;
data test;
  set test;
  if (tip > 0.15*total_bill) then greater15 = 1;
  else greater15 = 0;
run;

/* The descending option tells SAS to model the probability that
  greater15 = 1 */
proc genmod data=train descending;
  model greater15 = total_bill / dist = bin link = logit lrci;
  store out = logmod;
run;
                           The GENMOD Procedure

                            Model Information

                     Data Set               WORK.TRAIN
                     Distribution             Binomial
                     Link Function               Logit
                     Dependent Variable      greater15

                  Number of Observations Read         195
                  Number of Observations Used         195
                  Number of Events                    109
                  Number of Trials                    195

                             Response Profile

                     Ordered                     Total
                       Value    greater15    Frequency
```

```
                           1    1                  109
                           2    0                   86

PROC GENMOD is modeling the probability that greater15='1'.

                  Criteria For Assessing Goodness Of Fit

     Criterion                    DF          Value        Value/DF

     Log Likelihood                       -125.2918
     Full Log Likelihood                  -125.2918
     AIC (smaller is better)               254.5836
     AICC (smaller is better)              254.6461
     BIC (smaller is better)               261.1296


Algorithm converged.

            Analysis Of Maximum Likelihood Parameter Estimates

                                            Likelihood Ratio
                                 Standard     95% Confidence         Wald
 Parameter    DF    Estimate       Error          Limits          Chi-Square

 Intercept     1      1.6461      0.3946      0.8973      2.4501        17.40
 total_bill    1     -0.0706      0.0185     -0.1088     -0.0359        14.59
 Scale         0      1.0000      0.0000      1.0000      1.0000

                                Analysis Of Maximum
                               Likelihood Parameter
                                     Estimates

                              Parameter    Pr > ChiSq

                              Intercept        <.0001
                              total_bill       0.0001
                              Scale

NOTE: The scale parameter was held fixed.
```

b) Assess the model against the testing data.

```
/* Prediction on testing data */
proc plm source = logmod noprint;
    score data = test out = preds pred = pred / ilink;
run;

/* Determine how many were correctly classified */
data preds;
```

```
    set preds;
    if (pred < 0.5) then label = 0;
    else label = 1;
    if (label = greater15) then Result = "Correct";
    else Result = "Wrong";
run;

proc freq data = preds;
tables Result;
run;

                            The FREQ Procedure

                                               Cumulative    Cumulative
       Result     Frequency     Percent     Frequency       Percent
       ---------------------------------------------------------------
       Correct          34       69.39            34         69.39
       Wrong            15       30.61            49        100.00
```

GLM Procedure | PLM Procedure |FREQ Procedure

6.2 Fit a linear regression model on training data and assess against testing data.

a) Fit a linear regression model on training data.

```
/* Notice we are using new data sets that need to be read into the
  environment */
proc import out = train
  datafile = 'C:/Users/boston_train.csv'
  dbms = csv replace;
  getnames = yes;
run;
proc import out = test
  datafile = 'C:/Users/boston_test.csv'
  dbms = csv replace;
  getnames = yes;
run;

proc reg data = train outest=RegOut;
  predY: model Target = _0-_12;
quit;

                             The REG Procedure
                               Model: predY
                        Dependent Variable: Target

                  Number of Observations Read         354
                  Number of Observations Used         354
```

```
                           Analysis of Variance

                                     Sum of           Mean
Source                    DF        Squares         Square    F Value    Pr > F

Model                     13          22145     1703.47137      68.48    <.0001
Error                    340     8458.20364       24.87707
Corrected Total          353          30603

          Root MSE              4.98769    R-Square     0.7236
          Dependent Mean       22.48249    Adj R-Sq     0.7131
          Coeff Var            22.18479

                           Parameter Estimates

                        Parameter       Standard
  Variable     DF        Estimate          Error    t Value    Pr > |t|

  Intercept     1        36.10820        6.50497       5.55      <.0001
  _0            1        -0.08563        0.04277      -2.00      0.0461
  _1            1         0.04603        0.01715       2.68      0.0076
  _2            1         0.03641        0.07601       0.48      0.6322
  _3            1         3.24796        1.07414       3.02      0.0027
  _4            1       -14.87294        4.63609      -3.21      0.0015
  _5            1         3.57687        0.53699       6.66      <.0001
  _6            1        -0.00870        0.01685      -0.52      0.6059
  _7            1        -1.36890        0.25296      -5.41      <.0001
  _8            1         0.31312        0.08237       3.80      0.0002
  _9            1        -0.01288        0.00460      -2.80      0.0054
  _10           1        -0.97690        0.17100      -5.71      <.0001
  _11           1         0.01133        0.00336       3.37      0.0008
  _12           1        -0.52672        0.06256      -8.42      <.0001
```

b) Assess the model against the testing data.

```
/* Predicton on testing data */
proc score data = test score=RegOut type=parms predict out = Pred;
    var _0-_12;
run;

/* Compute the squared differences between predicted and target */
data Pred;
    set Pred;
    sq_error = (predY - Target)**2;
run;

/* Compute the mean of the squared differences (mean squared error) as an
   assessment of the model */
proc means data = Pred mean;
```

```
  var sq_error;
run;

                             The MEANS Procedure

                         Analysis Variable : sq_error

                                        Mean
                                ------------
                                  17.7713080
                                ------------
```

REG Procedure | SCORE Procedure | MEANS Procedure

6.3 Fit a decision tree model on training data and assess against testing data.

a) Fit a decision tree classification model.

i) Fit a decision tree classification model on training data and determine variable importance

```
/* Notice we are using new data sets that need to be read into the
   environment */
proc import out = train
    datafile = 'C:/Users/breastcancer_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/breastcancer_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

/* HPSPLIT procedure is used to fit a decision tree model */
proc hpsplit data = train seed = 29;
    target Target;
    input _0-_29;
    /* Export information about variable importance */
    output importance=import;
    /* Export the model code so this can be used to score testing data */
    code file='hpbreastcancer.sas';
run;

/* Output of this model gives assessment against training data
     and variable importance */

                            The HPSPLIT Procedure

                           Performance Information
```

Execution Mode	Single-Machine
Number of Threads	4

Data Access Information

Data	Engine	Role	Path
WORK.TRAIN	V9	Input	On Client

Model Information

Split Criterion Used	Entropy
Pruning Method	Cost-Complexity
Subtree Evaluation Criterion	Cost-Complexity
Number of Branches	2
Maximum Tree Depth Requested	10
Maximum Tree Depth Achieved	6
Tree Depth	3
Number of Leaves Before Pruning	15
Number of Leaves After Pruning	6
Model Event Level	1

Number of Observations Read	398
Number of Observations Used	398

The HPSPLIT Procedure

Model-Based Confusion Matrix

Actual	Predicted 1	0	Error Rate
1	242	1	0.0041
0	10	145	0.0645

Model-Based Fit Statistics for Selected Tree

N Leaves	ASE	Mis-class	Sensitivity	Specificity	Entropy	Gini	RSS
6	0.0229	0.0276	0.9959	0.9355	0.1297	0.0457	18.2063

Model-Based Fit Statistics for Selected Tree

AUC

```
                          0.9852

                    Variable Importance

                         Training
          Variable     Relative     Importance      Count

          _23           1.0000       11.2865           1
          _27           0.4072        4.5962           1
          _1            0.3487        3.9356           2
          _6            0.2355        2.6581           1
```

ii. Assess the model against the testing data.

```
/* Score the test data using the model code */
data scored;
    set test;
    %include 'hpbreastcancer.sas';
run;

/* Use prediction probabilities to generate predictions, and compare these
     to the true responses */
/* If the prediction probability is less than 0.5, classify this as a 0
   and otherwise classify as a 1.  This isn't the best method -- a better
   method would be randomly assigning a 0 or 1 when a probability of 0.5
   occurrs, but this insures that results are consistent */
data scored;
    set scored;
    if (P_Target1 < 0.5) then prediction = 0;
    else prediction = 1;
    if (Target = prediction) then Result = "Correct";
  else Result = "Wrong";
run;

/* Determine how many were correctly classified */
proc freq data = scored;
  tables Result;
run;

                    The FREQ Procedure

                                          Cumulative    Cumulative
       Result     Frequency     Percent     Frequency      Percent
       ------------------------------------------------------------
       Correct         157       91.81           157        91.81
       Wrong            14        8.19           171       100.00
```

HPSPLIT Procedure | %include & if-then/else statements | FREQ Procedure

b) Fit a decision tree regression model.

i) Fit a decision tree regression model on training data and determine variable importance.

```
/* Notice we are re-using data sets but it is good to re-read the
   original versions back into the environment */
proc import out = train
    datafile = 'C:/Users/boston_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/boston_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

/* HPSPLIT procedure is used to fit a decision tree model */
proc hpsplit data = train seed = 29;
    target Target / level = int;
    input _0-_12;
    /* Export information about variable importance */
    output importance=import;
    /* Export the model code so this can be used to score testing data */
    code file='hpboston.sas';
run;

/* Output of this model gives assessment against training data
   and variable importance */
                           The HPSPLIT Procedure

                         Performance Information

                    Execution Mode        Single-Machine
                    Number of Threads     4

                        Data Access Information

                Data           Engine     Role      Path

                WORK.TRAIN     V9         Input     On Client

                            Model Information

          Split Criterion Used                              Variance
          Pruning Method                             Cost-Complexity
          Subtree Evaluation Criterion               Cost-Complexity
          Number of Branches                                       2
          Maximum Tree Depth Requested                            10
```

```
          Maximum Tree Depth Achieved                           10
          Tree Depth                                            10
          Number of Leaves Before Pruning                      188
          Number of Leaves After Pruning                       101

            Number of Observations Read                     354
            Number of Observations Used                     354



                           The HPSPLIT Procedure

               Model-Based Fit Statistics for Selected Tree

                         N
                    Leaves           ASE           RSS

                       101        0.9750         345.2

                            Variable Importance

                                  Training
               Variable    Relative     Importance      Count

               _5            1.0000          132.8         13
               _12           0.6026        79.9998         16
               _7            0.3968        52.6772          9
               _4            0.2663        35.3541         12
               _0            0.2324        30.8579          7
               _9            0.1574        20.8933          8
               _6            0.1202        15.9544         12
               _10           0.1063        14.1112          4
               _11           0.0855        11.3541          8
               _2            0.0713         9.4695          5
               _8            0.0696         9.2408          3
               _1            0.0583         7.7437          3
```

ii. Assess the model against the testing data.

```
/* Score the test data using the model code */
data scored;
    set test;
    %include 'hpboston.sas';
run;

/* Compute the squared differences between predicted and target */
data scored;
    set scored;
    sq_error = (P_Target - Target)**2;
run;
```

```
/* Compute the mean of the squared differences (mean squared error) as an
     assessment of the model */
proc means data = scored mean;
  var sq_error;
run;

                            The MEANS Procedure

                       Analysis Variable : sq_error

                                      Mean
                               ------------
                                 24.6222895
                               ------------
```

HPSPLIT Procedure | %include statement | MEANS Procedure

6.4 Fit a random forest model on training data and assess against testing data.

a) Fit a random forest classification model.

i) Fit a random forest classification model on training data and determine variable importance.

```
/* Notice we are re-using data sets but it is good to re-read the
   original version back into the environment */
proc import out = train
    datafile = 'C:/Users/breastcancer_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/breastcancer_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

/* Output includes information about variable importance */
proc hpforest data = train;
    input _0 - _29 / level = interval;
    target Target / level = nominal;
    save file = 'hpbreastcancer2.bin';
run;

                          The HPFOREST Procedure

                          Performance Information
```

Execution Mode	Single-Machine
Number of Threads	4

Data Access Information

Data	Engine	Role	Path
WORK.TRAIN	V9	Input	On Client

Model Information

Parameter	Value	
Variables to Try	5	(Default)
Maximum Trees	100	(Default)
Inbag Fraction	0.6	(Default)
Prune Fraction	0	(Default)
Prune Threshold	0.1	(Default)
Leaf Fraction	0.00001	(Default)
Leaf Size Setting	1	(Default)
Leaf Size Used	1	
Category Bins	30	(Default)
Interval Bins	100	
Minimum Category Size	5	(Default)
Node Size	100000	(Default)
Maximum Depth	20	(Default)
Alpha	1	(Default)
Exhaustive	5000	(Default)
Rows of Sequence to Skip	5	(Default)
Split Criterion	.	Gini
Preselection Method	.	BinnedSearch
Missing Value Handling	.	Valid value

Number of Observations

Type	N
Number of Observations Read	398
Number of Observations Used	398

Baseline Fit Statistics

Statistic	Value
Average Square Error	0.238
Misclassification Rate	0.389
Log Loss	0.669

Fit Statistics

Number of Trees	Number of Leaves	Average Square Error (Train)	Average Square Error (OOB)	Misclassification Rate (Train)
1	16	0.03015	0.0750	0.03015
2	35	0.01947	0.0739	0.04523
3	53	0.01284	0.0724	0.00754
4	66	0.01225	0.0658	0.01005
5	80	0.01156	0.0700	0.00754
6	92	0.01124	0.0712	0.00754
7	106	0.00938	0.0633	0.00251
8	122	0.00879	0.0623	0.00000
9	139	0.00887	0.0611	0.00000
10	157	0.00867	0.0611	0.00000
11	171	0.00889	0.0589	0.00251
12	188	0.00874	0.0557	0.00000
13	203	0.00847	0.0551	0.00000
14	223	0.00841	0.0552	0.00000
15	241	0.00804	0.0537	0.00251
16	253	0.00795	0.0496	0.00251
17	268	0.00827	0.0489	0.00503
18	283	0.00813	0.0485	0.00251
19	300	0.00793	0.0471	0.00251
20	315	0.00783	0.0471	0.00251
21	329	0.00763	0.0465	0.00251
22	345	0.00747	0.0453	0.00000
23	361	0.00740	0.0448	0.00000
24	375	0.00744	0.0442	0.00000
25	392	0.00749	0.0449	0.00251
26	406	0.00764	0.0448	0.00251
27	420	0.00750	0.0440	0.00251
28	437	0.00764	0.0438	0.00000
29	451	0.00776	0.0431	0.00000
30	466	0.00774	0.0426	0.00000
31	484	0.00778	0.0432	0.00251
32	502	0.00759	0.0426	0.00000
33	518	0.00749	0.0420	0.00251
34	535	0.00747	0.0418	0.00000
35	550	0.00742	0.0415	0.00000
36	562	0.00746	0.0411	0.00000
37	578	0.00741	0.0411	0.00000
38	594	0.00731	0.0404	0.00000
39	609	0.00717	0.0407	0.00000
40	623	0.00720	0.0404	0.00000
41	642	0.00712	0.0405	0.00000
42	661	0.00702	0.0399	0.00000
43	679	0.00687	0.0397	0.00000
44	692	0.00677	0.0396	0.00000

45	710	0.00665	0.0392	0.00000
46	731	0.00652	0.0391	0.00000
47	741	0.00654	0.0387	0.00000
48	754	0.00661	0.0392	0.00000
49	769	0.00656	0.0393	0.00000
50	780	0.00657	0.0395	0.00000
51	795	0.00658	0.0395	0.00000
52	812	0.00657	0.0399	0.00000
53	829	0.00653	0.0399	0.00000
54	843	0.00662	0.0402	0.00000
55	856	0.00662	0.0403	0.00000
56	869	0.00663	0.0401	0.00000
57	883	0.00655	0.0396	0.00000
58	898	0.00653	0.0397	0.00000
59	914	0.00653	0.0394	0.00000
60	929	0.00661	0.0397	0.00000
61	946	0.00658	0.0396	0.00000
62	959	0.00655	0.0393	0.00000
63	975	0.00657	0.0394	0.00000
64	988	0.00660	0.0393	0.00000
65	1008	0.00662	0.0396	0.00000
66	1020	0.00671	0.0397	0.00000
67	1036	0.00675	0.0401	0.00000
68	1054	0.00672	0.0397	0.00000
69	1072	0.00678	0.0401	0.00000
70	1088	0.00686	0.0405	0.00000
71	1103	0.00692	0.0407	0.00000
72	1122	0.00692	0.0410	0.00000
73	1137	0.00695	0.0411	0.00000
74	1156	0.00682	0.0406	0.00000
75	1171	0.00678	0.0406	0.00000
76	1188	0.00668	0.0403	0.00000
77	1202	0.00665	0.0402	0.00000
78	1215	0.00661	0.0402	0.00000
79	1229	0.00661	0.0400	0.00000
80	1247	0.00658	0.0399	0.00000
81	1263	0.00657	0.0395	0.00000
82	1276	0.00659	0.0394	0.00000
83	1292	0.00659	0.0393	0.00000
84	1305	0.00652	0.0388	0.00000
85	1322	0.00649	0.0387	0.00000
86	1342	0.00644	0.0386	0.00000
87	1359	0.00647	0.0387	0.00000
88	1373	0.00655	0.0388	0.00000
89	1389	0.00655	0.0389	0.00000
90	1404	0.00652	0.0385	0.00000
91	1418	0.00658	0.0386	0.00000
92	1432	0.00652	0.0383	0.00000
93	1447	0.00649	0.0381	0.00000
94	1460	0.00654	0.0382	0.00000

95	1481	0.00657	0.0386	0.00000
96	1495	0.00650	0.0383	0.00000
97	1509	0.00646	0.0381	0.00000
98	1522	0.00651	0.0382	0.00000
99	1537	0.00649	0.0382	0.00000
100	1554	0.00647	0.0382	0.00000

Fit Statistics

Misclassification Rate (OOB)	Log Loss (Train)	Log Loss (OOB)
0.0750	0.6942	1.727
0.0895	0.1558	1.545
0.0952	0.0429	1.358
0.0893	0.0453	1.059
0.0877	0.0447	1.139
0.0871	0.0457	1.054
0.0803	0.0417	0.860
0.0821	0.0414	0.800
0.0842	0.0424	0.742
0.0787	0.0429	0.743
0.0734	0.0445	0.739
0.0732	0.0447	0.626
0.0732	0.0443	0.574
0.0781	0.0447	0.574
0.0756	0.0436	0.571
0.0729	0.0433	0.457
0.0678	0.0439	0.404
0.0603	0.0436	0.404
0.0628	0.0430	0.349
0.0628	0.0429	0.349
0.0628	0.0425	0.348
0.0628	0.0420	0.294
0.0653	0.0418	0.294
0.0628	0.0416	0.292
0.0628	0.0420	0.294
0.0628	0.0423	0.243
0.0603	0.0418	0.241
0.0603	0.0429	0.241
0.0578	0.0433	0.239
0.0578	0.0436	0.239
0.0628	0.0437	0.241
0.0578	0.0435	0.240
0.0553	0.0430	0.238
0.0553	0.0431	0.237
0.0553	0.0432	0.237
0.0528	0.0430	0.236
0.0528	0.0431	0.236

0.0528	0.0428	0.185
0.0553	0.0427	0.186
0.0528	0.0426	0.185
0.0553	0.0424	0.186
0.0553	0.0422	0.184
0.0553	0.0418	0.184
0.0553	0.0415	0.184
0.0578	0.0410	0.183
0.0578	0.0410	0.183
0.0528	0.0411	0.182
0.0578	0.0412	0.182
0.0553	0.0412	0.183
0.0553	0.0415	0.183
0.0528	0.0414	0.183
0.0578	0.0417	0.184
0.0578	0.0415	0.184
0.0578	0.0420	0.186
0.0578	0.0420	0.186
0.0528	0.0421	0.186
0.0528	0.0418	0.185
0.0528	0.0418	0.185
0.0528	0.0417	0.184
0.0553	0.0418	0.184
0.0528	0.0417	0.184
0.0553	0.0415	0.184
0.0578	0.0416	0.184
0.0578	0.0416	0.184
0.0578	0.0418	0.184
0.0578	0.0421	0.185
0.0603	0.0422	0.186
0.0578	0.0421	0.185
0.0553	0.0425	0.186
0.0578	0.0428	0.187
0.0578	0.0430	0.188
0.0578	0.0432	0.189
0.0603	0.0431	0.189
0.0603	0.0427	0.188
0.0578	0.0425	0.188
0.0553	0.0423	0.187
0.0578	0.0423	0.187
0.0578	0.0422	0.187
0.0578	0.0421	0.187
0.0553	0.0421	0.186
0.0578	0.0420	0.185
0.0553	0.0420	0.185
0.0553	0.0419	0.184
0.0553	0.0417	0.183
0.0528	0.0416	0.183
0.0553	0.0414	0.183
0.0528	0.0415	0.183

0.0528	0.0416	0.184
0.0503	0.0417	0.184
0.0477	0.0416	0.183
0.0503	0.0417	0.183
0.0503	0.0415	0.183
0.0528	0.0414	0.134
0.0503	0.0417	0.134
0.0528	0.0419	0.135
0.0503	0.0416	0.135
0.0477	0.0415	0.134
0.0477	0.0416	0.134
0.0477	0.0415	0.134
0.0452	0.0416	0.135

Loss Reduction Variable Importance

Variable	Number of Rules	Gini	OOB Gini	Margin	OOB Margin
_7	69	0.057751	0.05100	0.115502	0.10851
_27	116	0.057536	0.04812	0.115072	0.10648
_22	66	0.053462	0.04054	0.106925	0.09267
_23	92	0.049798	0.03969	0.099596	0.08961
_20	84	0.045727	0.03686	0.091453	0.08190
_2	43	0.030053	0.02561	0.060105	0.05721
_0	44	0.026259	0.01873	0.052518	0.04483
_13	47	0.018831	0.01425	0.037662	0.03329
_6	55	0.021984	0.01321	0.043968	0.03523
_3	16	0.010751	0.01275	0.021502	0.02310
_26	84	0.017139	0.00693	0.034279	0.02387
_21	73	0.009979	0.00400	0.019958	0.01367
_10	31	0.007944	0.00273	0.015889	0.01089
_12	31	0.007102	0.00217	0.014204	0.00929
_17	31	0.002941	0.00049	0.005882	0.00286
_5	12	0.001882	-0.00010	0.003764	0.00152
_16	17	0.001134	-0.00055	0.002268	0.00089
_11	23	0.001679	-0.00057	0.003358	0.00096
_8	22	0.001543	-0.00077	0.003086	0.00052
_18	22	0.001787	-0.00105	0.003573	0.00081
_9	23	0.001656	-0.00105	0.003312	0.00063
_4	22	0.002237	-0.00114	0.004475	0.00147
_1	58	0.008366	-0.00147	0.016732	0.00648
_24	80	0.010527	-0.00149	0.021054	0.00906
_25	55	0.005040	-0.00151	0.010081	0.00449
_28	70	0.008423	-0.00168	0.016846	0.00617
_15	16	0.001345	-0.00203	0.002690	-0.00059
_14	29	0.001679	-0.00282	0.003357	-0.00110
_19	49	0.003804	-0.00413	0.007609	-0.00028
_29	74	0.005801	-0.00418	0.011603	0.00225

ii) Assess the model against the testing data.

```
/* Prediction on testing data */
ods select none;
proc hp4score data = test seed = 29;
    score file = 'hpbreastcancer2.bin' out = scored;
run;
ods select all;

/* Determine how many were correctly classified */
data scored;
    set scored;
    if (I_Target = Target) then Result = "Correct";
    else Result = "Wrong";
run;

proc freq data = scored;
  tables Result;
run;

                            The FREQ Procedure

                                            Cumulative    Cumulative
       Result     Frequency     Percent     Frequency      Percent
       ------------------------------------------------------------
       Correct         166       97.08           166        97.08
       Wrong             5        2.92           171       100.00
```

HPFOREST Procedure | HP4SCORE Procedure | FREQ Procedure

b) Fit a random forest regression model.

i) Fit a random forest regression model on training data and determine variable importance.

```
/* Notice we are re-using data sets but it is good to re-read the original
   versions back into the environment */
proc import out = train
    datafile = 'C:/Users/boston_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/boston_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

proc hpforest data = train;
    input _0-_12 / level = interval;
    target Target / level = interval;
    save file = 'hpboston2.bin';
run;
```

The HPFOREST Procedure

Performance Information

Execution Mode	Single-Machine
Number of Threads	4

Data Access Information

Data	Engine	Role	Path
WORK.TRAIN	V9	Input	On Client

Model Information

Parameter	Value	
Variables to Try	4	(Default)
Maximum Trees	100	(Default)
Inbag Fraction	0.6	(Default)
Prune Fraction	0	(Default)
Prune Threshold	0.1	(Default)
Leaf Fraction	0.00001	(Default)
Leaf Size Setting	1	(Default)
Leaf Size Used	1	
Category Bins	30	(Default)
Interval Bins	100	
Minimum Category Size	5	(Default)
Node Size	100000	(Default)
Maximum Depth	20	(Default)
Alpha	1	(Default)
Exhaustive	5000	(Default)
Rows of Sequence to Skip	5	(Default)
Split Criterion	.	Variance
Preselection Method	.	BinnedSearch
Missing Value Handling	.	Valid value

Number of Observations

Type	N
Number of Observations Read	354
Number of Observations Used	354

Baseline Fit Statistics

Statistic	Value
Average Square Error	86.450

Fit Statistics

Number of Trees	Number of Leaves	Average Square Error (Train)	Average Square Error (OOB)
1	187	19.2696	47.7098
2	375	11.0807	42.1586
3	576	6.4925	30.3264
4	771	4.7796	24.0582
5	959	4.5157	23.4562
6	1155	4.9109	22.6314
7	1347	4.1582	23.0371
8	1547	3.7434	21.2459
9	1748	3.4531	21.2847
10	1946	3.1072	20.2029
11	2136	3.2121	19.2038
12	2332	3.1237	18.8287
13	2525	3.3520	18.5620
14	2725	3.3682	18.2171
15	2923	3.2042	18.1505
16	3116	3.0559	17.8622
17	3313	3.0496	17.9996
18	3507	2.9018	17.6530
19	3701	2.7845	17.7495
20	3901	2.9128	17.7119
21	4094	2.9422	17.0785
22	4285	2.8756	16.2443
23	4480	2.8453	16.4825
24	4672	2.8007	16.3980
25	4866	2.8716	16.3951
26	5064	2.7869	15.7487
27	5256	2.6759	15.3203
28	5447	2.6251	14.9469
29	5638	2.6434	15.0160
30	5827	2.5882	14.7212
31	6019	2.5907	14.9240
32	6213	2.5328	14.9864
33	6406	2.4964	14.8766
34	6601	2.4117	14.6833
35	6794	2.3887	14.7204
36	6992	2.3944	14.7274
37	7180	2.4368	14.9644
38	7362	2.4589	14.7638
39	7555	2.4956	14.6984
40	7755	2.4616	14.5620
41	7957	2.4374	14.4358
42	8151	2.4279	14.3331

43	8326	2.3984	14.3855
44	8509	2.3441	14.2211
45	8699	2.3285	14.0444
46	8888	2.3773	14.1862
47	9091	2.3820	13.9894
48	9293	2.3723	13.7633
49	9487	2.3369	13.5058
50	9677	2.3450	13.4064
51	9873	2.3123	13.2669
52	10064	2.2963	13.2131
53	10256	2.3327	13.3625
54	10449	2.3690	13.5171
55	10642	2.3361	13.4808
56	10840	2.2878	13.3526
57	11038	2.3348	13.3863
58	11229	2.3017	13.3146
59	11420	2.2946	13.2027
60	11614	2.2704	13.0569
61	11800	2.2582	13.0369
62	11994	2.2237	13.0228
63	12188	2.1841	12.8869
64	12391	2.1758	12.8971
65	12588	2.1407	12.6641
66	12787	2.1258	12.5012
67	12976	2.1301	12.5143
68	13172	2.1087	12.3520
69	13371	2.0942	12.2695
70	13564	2.0966	12.2662
71	13747	2.0913	12.2450
72	13946	2.0750	12.2277
73	14128	2.0646	12.1783
74	14312	2.0417	12.0684
75	14514	2.0736	12.0957
76	14709	2.0818	12.1248
77	14903	2.0602	12.1210
78	15104	2.0610	12.1531
79	15307	2.0625	12.1542
80	15499	2.0450	12.1263
81	15690	2.0633	12.2513
82	15881	2.0662	12.3139
83	16066	2.0781	12.4064
84	16264	2.0635	12.3756
85	16456	2.0853	12.4839
86	16644	2.0671	12.4459
87	16843	2.0761	12.4871
88	17038	2.0730	12.4799
89	17224	2.0802	12.5118
90	17426	2.1014	12.5369
91	17625	2.0904	12.5344
92	17816	2.1211	12.6684

```
                   93          18012      2.1473     12.7202
                   94          18207      2.1337     12.7711
                   95          18399      2.1340     12.8146
                   96          18590      2.1635     12.8990
                   97          18784      2.1495     12.9017
                   98          18971      2.1447     12.9297
                   99          19161      2.1445     12.9491
                  100          19358      2.1487     12.9533

                 Loss Reduction Variable Importance

               Number                    OOB        Absolute   OOB Absolute
  Variable   of Rules         MSE        MSE           Error          Error

  _5             1612    25.20881   21.74008        1.641774       1.234735
  _12            4407    27.11737   21.45478        1.839931       1.088602
  _2              902     7.59158    5.57044        0.491575       0.265974
  _4             1079     4.64446    1.72722        0.448335       0.194131
  _10            1028     4.25296    1.66205        0.308733       0.104484
  _0              390     2.47937    0.82684        0.184771       0.087478
  _9             1301     2.79671    0.58671        0.285697       0.061757
  _1              139     0.07821   -0.05230        0.018239      -0.005778
  _3              187     0.75658   -0.07944        0.033998      -0.014106
  _7             2293     6.36749   -0.21261        0.567863       0.104445
  _8              834     0.56810   -0.35514        0.096400      -0.024171
  _11            3539     2.87942   -0.56051        0.477124      -0.003090
  _6             1547     1.66719   -0.56374        0.266348      -0.012889
```

ii) Assess the model against the testing data.

```
/* Prediction on testing data */
ods select none;
proc hp4score data = test seed = 29;
    score file = 'hpboston2.bin' out = scored;
run;
ods select all;

/* Compute the squared differences between predicted and target */
data scored;
    set scored;
    sq_error = (P_Target - Target)**2;
run;

/* Compute the mean of the squared differences (mean squared error) as an
   assessment of the model */
proc means data = scored mean;
  var sq_error;
run;
                             The MEANS Procedure
```

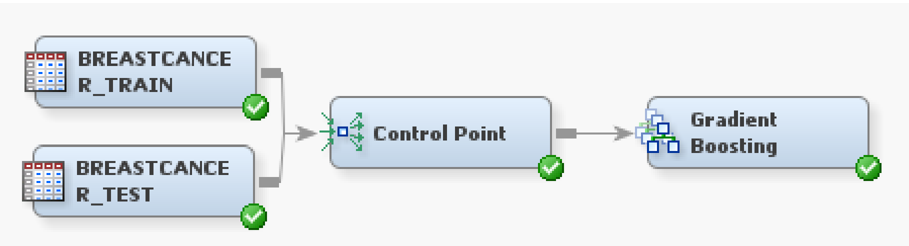
```
                        Analysis Variable : sq_error

                                        Mean
                                ------------
                                   8.7840698
                                ------------
```

HPFOREST Procedure | HP4SCORE Procedure | MEANS Procedure

6.5 Fit a gradient boosting model on training data and assess against testing data.

a) Fit a gradient boosting classification model.

Currently, there is not a gradient boosting procedure available in Base SAS Therefore, the best method to create a gradient boosting model currently is using SAS Enterprise Miner. Create the following diagram in SAS Enterprise Miner:

Output: 

For the Gradient Boosting node, set Seed = 29, set Shrinkage = 0.01, and set Train Proportion = 100.

This diagram results in the following variable importance and misclassification against training & testing data:

Output:

Variable Name	Importance
_23	1
_27	0.988671
_7	0.382448
_13	0.294633
_22	0.178301
_1	0.113222
_24	0.068714
_20	0.044286
_19	0.03198
_21	0
_10	0
_14	0
_16	0
_18	0
_2	0
_15	0
_29	0
_3	0
_0	0
_17	0
_25	0
_11	0
_12	0
_28	0
_6	0
_26	0
_4	0
_5	0
_8	0
_9	0

Output:

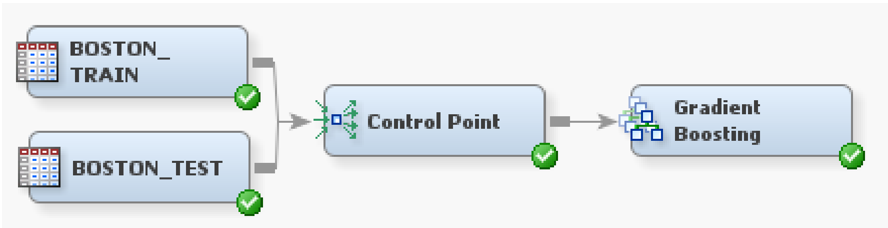
Statistics Label	Train	Test
Sum of Frequencies	398	171
Sum of Case Weights Times Freq	796	342
Misclassification Rate	0.035176	0.040936
Maximum Absolute Error	0.717472	0.743284
Sum of Squared Errors	99.49027	40.19162
Average Squared Error	0.124988	0.117519
Root Average Squared Error	0.353536	0.342811
Divisor for ASE	796	342
Total Degrees of Freedom	398	.

Output:

```
Classification Table

Data Role=TRAIN Target Variable=Target Target Label=' '

                       Target      Outcome    Frequency     Total
Target    Outcome    Percentage   Percentage    Count     Percentage

  0          0        100.000      90.968        141       35.4271
  0          1          5.447       9.032         14        3.5176
  1          1         94.553     100.000        243       61.0553
```

b) Fit a gradient boosting regression model.

Again, there is not a gradient boosting procedure available in Base SAS, currently. Create the following diagram in SAS Enterprise Miner:

Output: 

For the Gradient Boosting node, set Seed = 29, set Shrinkage = 0.01, and set Train Proportion = 100.

This diagram results in the following variable importance and root mean squared error against training & testing data:

Output:

Variable Name	Importance
_12	1
_5	0.953865
_0	0.074612
_4	0
_1	0
_11	0
_3	0
_2	0
_10	0
_8	0
_7	0
_6	0
_9	0

Output:

Statistics Label	Train	Test
Sum of Frequencies	354	152
Sum of Case Weights Times Freq	354	152
Maximum Absolute Error	27.01107	21.83158
Sum of Squared Errors	17560.66	7109.598
Average Squared Error	49.60639	46.77367
Root Average Squared Error	7.04318	6.839128
Divisor for ASE	354	152
Total Degrees of Freedom	354	.

6.6 Fit an extreme gradient boosting model on taining data and assess against testing data.

a) Fit an extreme gradient boosting classification model.

Fit an extreme gradient boosting classification model on training data and assess the model against the testing data.

For more information on the R code used below, please see the R Tutorial.

```
proc iml;
  submit / R;
    train = read.csv('C:/Users/breastcancer_train.csv')
    test = read.csv('C:/Users/breastcancer_test.csv')

    library(xgboost)
      set.seed(29)

    xgbMod <- xgboost(data.matrix(subset(train, select = -c(Target))),
                      data.matrix(train$Target), max_depth = 3, nrounds = 2,
                      objective = "binary:logistic", n_estimators = 2500,
                      shrinkage = .01)
    # Prediction on testing data
    predictions <- predict(xgbMod, data.matrix(subset(test, select =
                                                  -c(Target))))
    pred.response <- ifelse(predictions < 0.5, 0, 1)

    # Determine how many were correctly classified
    Results <- ifelse(test$Target == pred.response, "Correct", "Wrong")
    table(Results)
  endsubmit;
quit;

[1] train-error:0.037688
[2] train-error:0.020101
Results
Correct   Wrong
    165       6
```

Fit an extreme gradient boosting regression model on training data and assess the model against the testing data.

```
proc iml;
  submit / R;
    train = read.csv('C:/Users/boston_train.csv')
    test = read.csv('C:/Users/boston_test.csv')

    library(xgboost)
      set.seed(29)
```

```
    xgbMod <- xgboost(data.matrix(subset(train, select = -c(Target))),
                      data.matrix(train$Target / 50), max_depth = 3,
                      nrounds = 2, n_estimators = 2500, shrinkage = .01)

    # Predict the target in the testing data, remembering to
    # multiply by 50
    prediction = data.frame(matrix(ncol = 0, nrow = nrow(test)))
    prediction$target_hat <- predict(xgbMod,
                                     data.matrix(subset(test,
                                                 select = - c(Target))))*50

    # Compute the squared difference between predicted tip and actual tip
    prediction$sq_diff <- (prediction$target_hat - test$Target)**2

    # Compute the mean of the squared differences (mean squared error)
    # as an assessment of the model
    mean_sq_error <- mean(prediction$sq_diff)
    print(mean_sq_error)
  endsubmit;
quit;
```

```
[1] train-rmse:0.146609
[2] train-rmse:0.114851
[1] 36.13079
```

IML Procedure

6.7 Fit a support vector model on training data and assess against testing data.

a) Fit a support vector classification model.

i) Fit a support vector classification model on training data.

```
/* Notice we are re-using data sets but it is good to re-read the original
   versions back into the environment */
proc import out = train
    datafile = 'C:/Users/breastcancer_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/breastcancer_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

/* Fit a support vector classification model */
proc hpsvm data = train;
  input _0-_29 / level = interval;
```

```
    target Target / level = nominal;
    code file='hpbreastcancer3.sas';
run;

                            The HPSVM Procedure

                          Performance Information

                    Execution Mode       Single-Machine
                    Number of Threads    4

                         Data Access Information

                Data          Engine    Role     Path

                WORK.TRAIN    V9        Input    On Client

                            Model Information

                 Task Type                  C_CLAS
                 Optimization Technique     Interior Point
                 Scale                      YES
                 Kernel Function            Linear
                 Penalty Method             C
                 Penalty Parameter          1
                 Maximum Iterations         25
                 Tolerance                  1e-06

                Number of Observations Read             398
                Number of Observations Used             398

                             Training Results

             Inner Product of Weights                35.2508001
             Bias                                     -6.375275
             Total Slack (Constraint Violations)     34.3511008
             Norm of Longest Vector                  3.79226578
             Number of Support Vectors                       71
             Number of Support Vectors on Margin             63
             Maximum F                               11.4630802
             Minimum F                               -4.7061491
             Number of Effects                               30
             Columns in Data Matrix                          30

                             Iteration History

               Iteration    Complementarity     Feasibility

                       1       1002265.3132    88067.240896
```

```
                      2       1411.2168312    80.210592636
                      3       210.36307705    8.0210592E-7
                      4       5.5675772656    1.2652961E-8
                      5       0.8865572275    1.544403E-10
                      6       0.2947605635    3.866263E-11
                      7       0.1606295757    1.766043E-11
                      8       0.0981078445    8.719581E-12
                      9       0.0603316585    4.770961E-12
                     10       0.0258720492      1.4998E-12
                     11       0.0171466879    5.151435E-13
                     12       0.0090859249    1.514344E-13
                     13       0.0023785349    3.508305E-14
                     14       0.0001072635    3.552714E-15
                     15        4.813479E-7    5.617035E-15

                        Classification Matrix

                                Training Prediction
               Observed            1           0       Total

               1                 243           0         243
               0                   7         148         155
               Total             250         148         398

                            Fit Statistics

                        Statistic      Training

                        Accuracy         0.9824
                        Error            0.0176
                        Sensitivity      1.0000
                        Specificity      0.9548
```

HPSVM Procedure

ii) Assess the model against the testing data.

```
/* Prediction on testing data */
data scored;
    set test;
    %include 'hpbreastcancer3.sas';
run;

/* Determine how many were correctly classified */
data scored;
    set scored;
    if (I_Target = Target) then Result = "Correct";
    else Result = "Wrong";
run;
```

```
proc freq data = scored;
  tables Result;
run;

                            The FREQ Procedure

                                                Cumulative    Cumulative
       Result     Frequency     Percent     Frequency      Percent
       ------------------------------------------------------------
       Correct         167       97.66           167        97.66
       Wrong             4        2.34           171       100.00
```

%include & if-then/else statements | FREQ Procedure

b) Fit a support vector regression model.

Not available in this current release.

6.8 Fit a neural network model on training data and assess against testing data.

a) Fit a neural network classification model.

i) Fit a neural network classification model on training data.

```
/* Notice we are using new data sets that need to be read into the
   environment */
proc import out = train
  datafile = 'C:/Users/digits_train.csv'
  dbms = csv replace;
  getnames = yes;
run;
proc import out = test
  datafile = 'C:/Users/digits_test.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* In order to use the NEURAL Procedure we first need to create a data
   mining database (DMDB) that reflects the original data */
proc dmdb batch data = train
    out = dmtrain
    dmdbcat = digits;
  var _0 - _63;
  class Target;
  target Target;
run;
proc dmdb batch data = test
    out = dmtest
    dmdbcat = digits;
```

```
  var _0 - _63;
  class Target;
  target Target;
run;

/* Now we can fit the neural network model */
/* Neural network produces a lot of output which is why here
   "nloptions noprint" is specified */
proc neural data = train dmdbcat = digits random = 29;
  nloptions noprint;
  input _0 - _63 / level = interval;
  target Target / level = nominal;
  archi MLP hidden=100;
  train maxiter = 200;
  score out = out outfit = fit;
  score data = test out = gridout;
run;
```

DMDB Procedure | NEURAL Procedure

ii) Assess the model against the testing data.

```
/* Prediction on testing data */
data scored;
    set gridout;
    rename I_Target = Prediction;
run;

/* This produces a confusion matrix */
proc freq data = scored;
    tables Target*Prediction / nopercent norow nocol;
run;

                            The FREQ Procedure

                       Table of Target by Prediction

      Target     Prediction(Into: Target)

      Frequency|0       |1       |2       |3       |4       |  Total
      ---------+--------+--------+--------+--------+--------+
             0 |     58 |      0 |      0 |      0 |      0 |     58
      ---------+--------+--------+--------+--------+--------+
             1 |      1 |     56 |      0 |      0 |      0 |     58
      ---------+--------+--------+--------+--------+--------+
             2 |      0 |      0 |     58 |      0 |      0 |     58
      ---------+--------+--------+--------+--------+--------+
             3 |      0 |      0 |      0 |     58 |      0 |     59
      ---------+--------+--------+--------+--------+--------+
             4 |      0 |      0 |      0 |      0 |     51 |     54
      ---------+--------+--------+--------+--------+--------+
```

```
        5 |       0 |       0 |       0 |       0 |       0 |      59
---------+--------+--------+--------+--------+--------+
        6 |       0 |       0 |       0 |       0 |       0 |      41
---------+--------+--------+--------+--------+--------+
        7 |       0 |       0 |       0 |       0 |       0 |      51
---------+--------+--------+--------+--------+--------+
        8 |       0 |       4 |       0 |       0 |       0 |      45
---------+--------+--------+--------+--------+--------+
        9 |       0 |       0 |       0 |       0 |       0 |      57
---------+--------+--------+--------+--------+--------+
Total          59       60       58       58       51      540
(Continued)

                 Table of Target by Prediction

Target     Prediction(Into: Target)

Frequency|5        |6        |7        |8        |9        |  Total
---------+--------+--------+--------+--------+--------+
        0 |       0 |       0 |       0 |       0 |       0 |      58
---------+--------+--------+--------+--------+--------+
        1 |       0 |       1 |       0 |       0 |       0 |      58
---------+--------+--------+--------+--------+--------+
        2 |       0 |       0 |       0 |       0 |       0 |      58
---------+--------+--------+--------+--------+--------+
        3 |       1 |       0 |       0 |       0 |       0 |      59
---------+--------+--------+--------+--------+--------+
        4 |       1 |       1 |       0 |       1 |       0 |      54
---------+--------+--------+--------+--------+--------+
        5 |      58 |       0 |       0 |       0 |       1 |      59
---------+--------+--------+--------+--------+--------+
        6 |       0 |      41 |       0 |       0 |       0 |      41
---------+--------+--------+--------+--------+--------+
        7 |       1 |       0 |      50 |       0 |       0 |      51
---------+--------+--------+--------+--------+--------+
        8 |       0 |       0 |       0 |      39 |       2 |      45
---------+--------+--------+--------+--------+--------+
        9 |       2 |       0 |       0 |       2 |      53 |      57
---------+--------+--------+--------+--------+--------+
Total          63       43       50       42       56      540
```

FREQ Procedure

b) Fit a neural network regression model.

i) Fit a neural network regression model on training data.

```
/* Notice we are re-using data sets but it is good to re-read the original
   versions back into the environment */
proc import out = train
  datafile = 'C:/Users/boston_train.csv'
```

```
  dbms = csv replace;
  getnames = yes;
run;
proc import out = test
  datafile = 'C:/Users/boston_test.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* In order to use the NEURAL Procedure we first need to create a data
   mining database (DMDB) that reflects the original data */
proc dmdb batch data = train
    out = dmtrain
    dmdbcat = boston;
  var _0 - _12 Target;
  target Target;
run;
proc dmdb batch data = test
    out = dmtest
    dmdbcat = boston;
  var _0 - _12 Target;
  target Target;
run;

/* Now we can fit the neural network model */
/* Neural network produces a lot of output which is why here
   "nloptions noprint" is specified */
proc neural data = train dmdbcat = boston random = 29;
  nloptions noprint;
  archi MLP hidden=100;
  input _0 - _12 / level = interval;
  target Target / level = interval;
  train maxiter = 250;
  score data = test outfit = netfit out = gridout;
run;
```

DMDB Procedure | NEURAL Procedure

ii) Assess the model against the testing data.

```
/* Prediction on testing data */
data scored(keep = sq_error P_Target Target);
    set gridout;
    sq_error = (P_Target - Target)**2;
run;

/* Determine mean squared error */
proc means data = scored mean;
var sq_error;
run;
```

```
                            The MEANS Procedure

                       Analysis Variable : sq_error

                                   Mean
                           ------------
                             16.1149499
                           ------------
```

MEANS Procedure

---

7 Unsupervised Machine Learning

7.1 KMeans Clustering

```
proc import out = iris
    datafile = 'C:/Users/iris.csv'
    dbms = csv replace;
    getnames = yes;
run;

data iris;
    length Species $ 20;
    set iris;
    if (Target = 0) then Species = "Setosa";
    if (Target = 1) then Species = "Versicolor";
    if (Target = 2) then Species = "Virginica";
run;

proc fastclus data=iris maxclusters=3 out=kmeans random = 29 noprint;
    var PetalLength PetalWidth SepalLength SepalWidth;
run;

proc freq data = kmeans;
    tables Species*Cluster / nopercent nocol norow;
run;
                            The FREQ Procedure

                        Table of Species by CLUSTER

              Species     CLUSTER(Cluster)

              Frequency  |       1|       2|       3|  Total
              -----------+--------+--------+--------+
              Setosa     |      0 |     50 |      0 |     50
              -----------+--------+--------+--------+
```

```
              Versicolor |      0 |      0 |     50 |     50
              -----------+--------+--------+--------+
              Virginica  |     33 |      0 |     17 |     50
              -----------+--------+--------+--------+
              Total            33       50       67      150
```

FASTCLUS Procedure | FREQ Procedure

7.2 Spectral Clustering

For more information on the R code used below, please see the R Tutorial.

```
proc iml;
  submit / R;
    iris = read.csv('C:/Users/iris.csv')
      iris$Species = ifelse(iris$Target == 0, "Setosa",
                            ifelse(iris$Target == 1, "Versicolor",
                                   "Virginica"))
      features <- as.matrix(subset(iris, select = c(PetalLength,
                                                    PetalWidth, SepalLength,
                                                    SepalWidth)))
    library(kernlab)
    set.seed(29)
    spectral <- specc(features, centers = 3, iterations = 10,
                      nystrom.red = TRUE)
    labels <- as.data.frame(spectral)
    table(iris$Species, labels$spectral)
  endsubmit;
quit;


              1  2  3
  Setosa     50  0  0
  Versicolor  0 47  3
  Virginica   0  3 47
```

IML Procedure

7.3 Ward Hierarchical Clustering

```
proc import out = iris
  datafile = 'C:/Users/iris.csv'
  dbms = csv replace;
  getnames = yes;
run;

data iris;
  length Species $ 20;
  set iris;
    if (Target = 0) then Species = "Setosa";
    if (Target = 1) then Species = "Versicolor";
```

```
    if (Target = 2) then Species = "Virginica";
run;

proc cluster data = iris method = ward print=15 ccc pseudo noprint;
   var petal: sepal:;
   copy species;
run;

proc tree noprint ncl=3 out=out;
   copy petal: sepal: species;
run;

proc freq data = out;
  tables Species*Cluster;
run;
```

CLUSTER Procedure | TREE Procedure | FREQ Procedure

7.4 DBSCAN

For more information on the R code used below, please see the R Tutorial.

```
proc iml;
  submit / R;
    iris = read.csv('C:/Users/iris.csv')
      iris$Species = ifelse(iris$Target == 0, "Setosa",
                            ifelse(iris$Target == 1, "Versicolor",
                                   "Virginica"))
      features <- as.matrix(subset(iris, select = c(PetalLength,
                                                    PetalWidth, SepalLength,
                                                    SepalWidth)))
    library(dbscan)
      set.seed(29)
    dbscan <- dbscan(features, eps = 0.5)
    labels <- dbscan$cluster
    table(iris$Species, labels)
  endsubmit;
quit;

            labels
              0  1  2
  Setosa      1 49  0
  Versicolor  6  0 44
  Virginica  10  0 40
```

IML Procedure

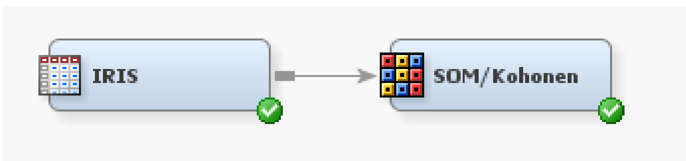
7.5 Self-organizing map

Currently, there is not a self-organizing map procedure available in Base SAS. Therefore, the best method to create a self-organizing map currently is using SAS Enterprise Miner. First, you need to read in the Iris data set, setting the Species/Target variable to be dropped before investigation.

Output:

Name	Role	Level	Report	Order	Drop
PetalLength	Input	Interval	No		No
PetalWidth	Input	Interval	No		No
SepalLength	Input	Interval	No		No
SepalWidth	Input	Interval	No		No
Target	Target	Interval	No		Yes

Then create the following diagram in SAS Enterprise Miner:

Output: 

For the SOM/Kohonen node set the following options:

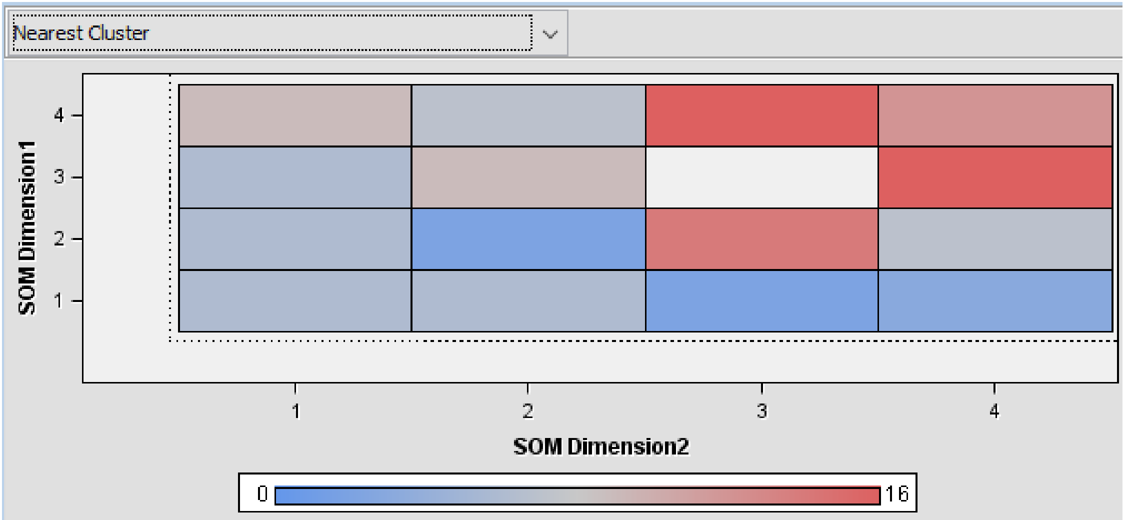
1. Choose the Kohonen SOM method.
2. Set row and column to both be 4.
3. Under the "Kohonen" options section, set "Use Defaults" to "No", and open the Kohonen Options window by clicking the ... box.
4. Set the following options in the popup window:

Kohonen Options

Output:

Property	Value
Learning Rate	0.05
Initial Rate	0.05
Final Rate	0.01
Number of Steps	1000
Convergence Criterion	1.0E-4
Max Iterations	100
Max Steps	1200

This model produces the following output which is similar to the output of R and Python:

Output: 

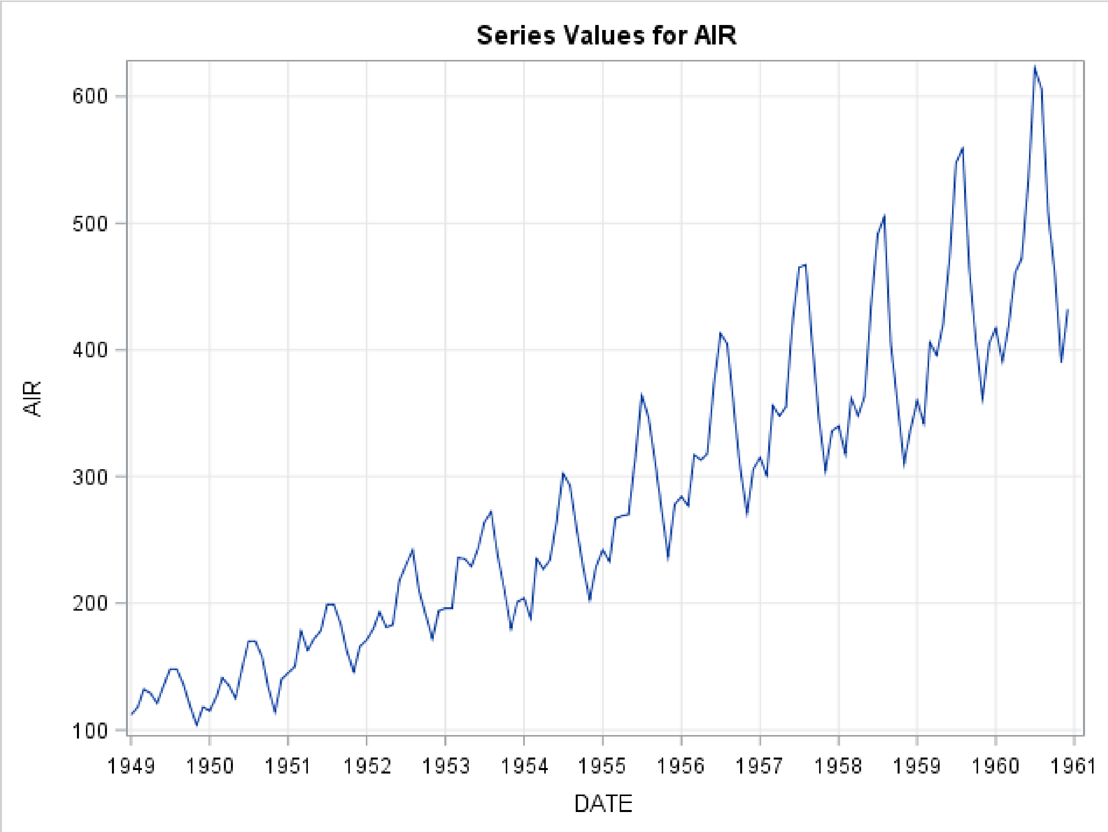
---

8 Forecasting

8.1 Fit an ARIMA model to a timeseries.

a) Plot the timeseries.

```
proc import out = air
  datafile = 'C:/Users/air.csv'
  dbms = csv replace;
  getnames = yes;
run;

proc timeseries data = air plot = series;
    id date interval = month;
    var air;
run;
```

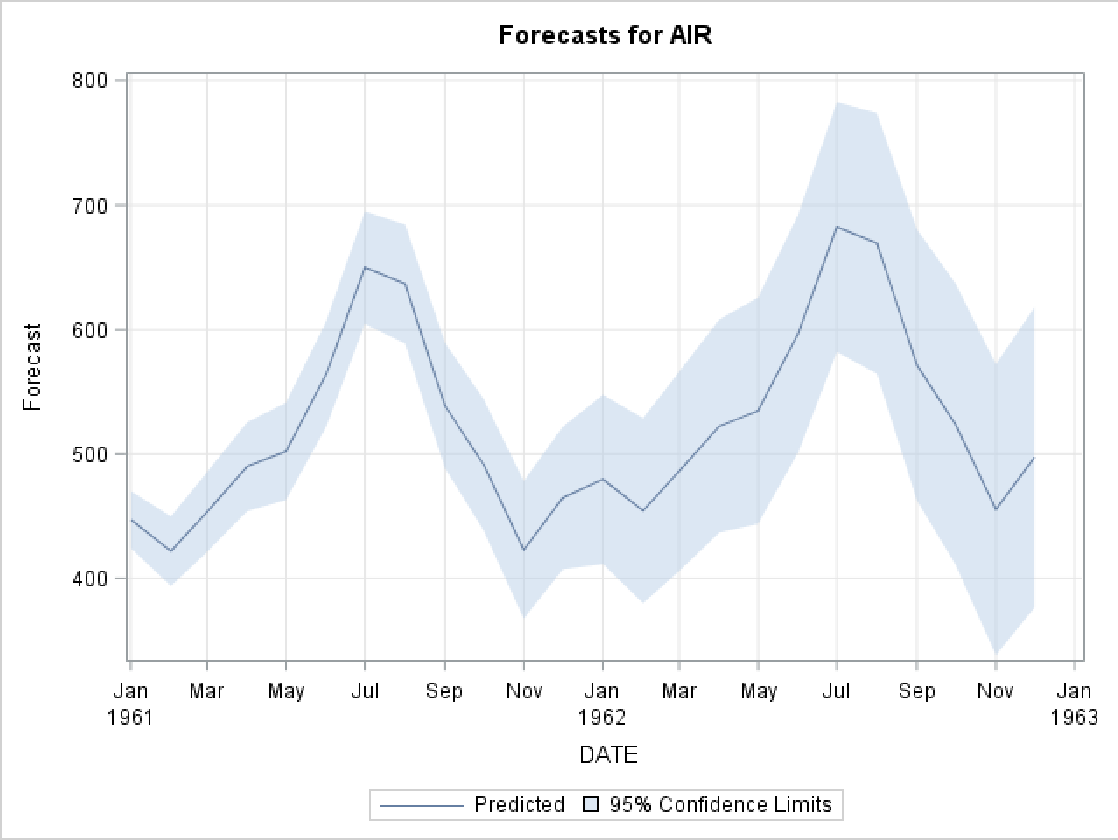
Output: 

TIMESERIES Procedure

b) Fit an ARIMA model and predict 2 years (24 months).

The output of this code has been limited for space reasons.

```
proc arima data = air;
    identify var = air(1,12) noprint;
    estimate q=(1)(12) noint method=ml noprint;
    forecast id=date interval=month out=forecast;
run;

/* SAS automatically predicts 2 years out and plots the predictions */
```

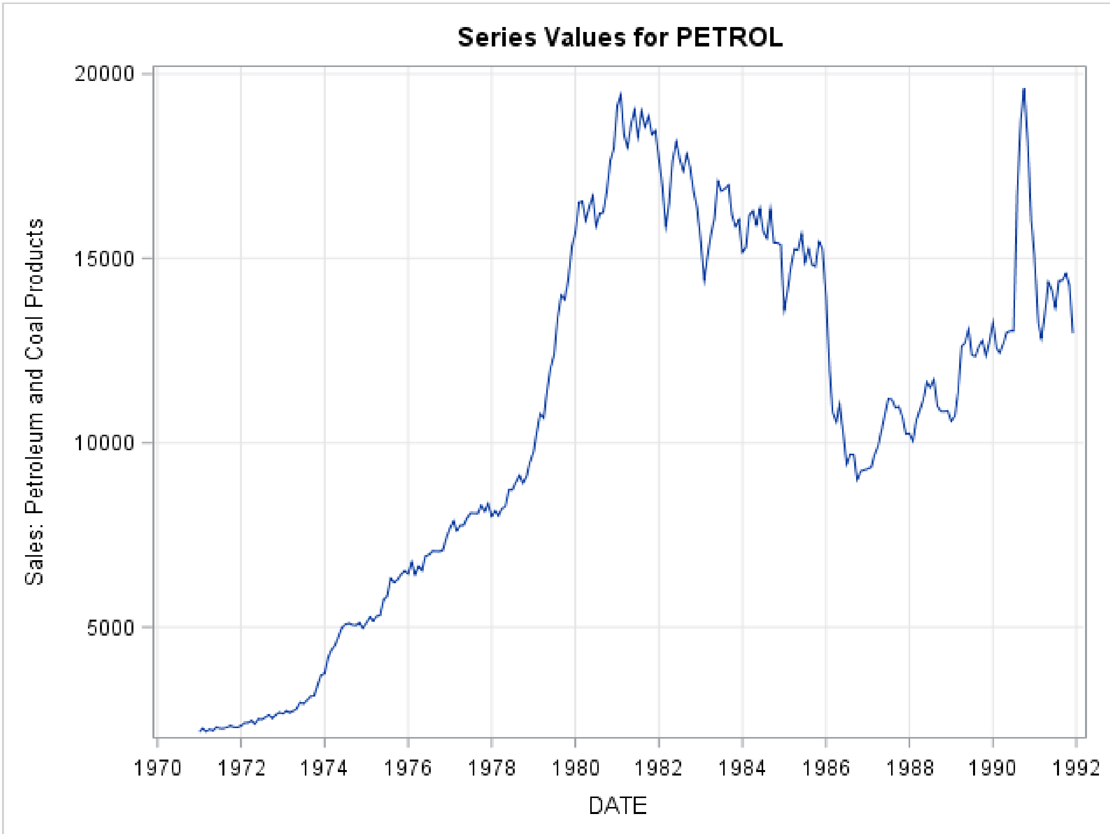
Output: 

ARIMA Procedure

8.2 Fit a Simple Exponential Smoothing model to a timeseries.

a) Plot the timeseries.

```
proc import out = usecon
  datafile = 'C:/Users/usecon.csv'
  dbms = csv replace;
  getnames = yes;
run;
proc timeseries data = usecon plot = series;
    id date interval = month;
    var petrol;
run;
                         The TIMESERIES Procedure

                              Input Data Set

                  Name                           WORK.USECON
                  Label
                  Time ID Variable                      DATE
                  Time Interval                        MONTH
                  Length of Seasonal Cycle                12
```

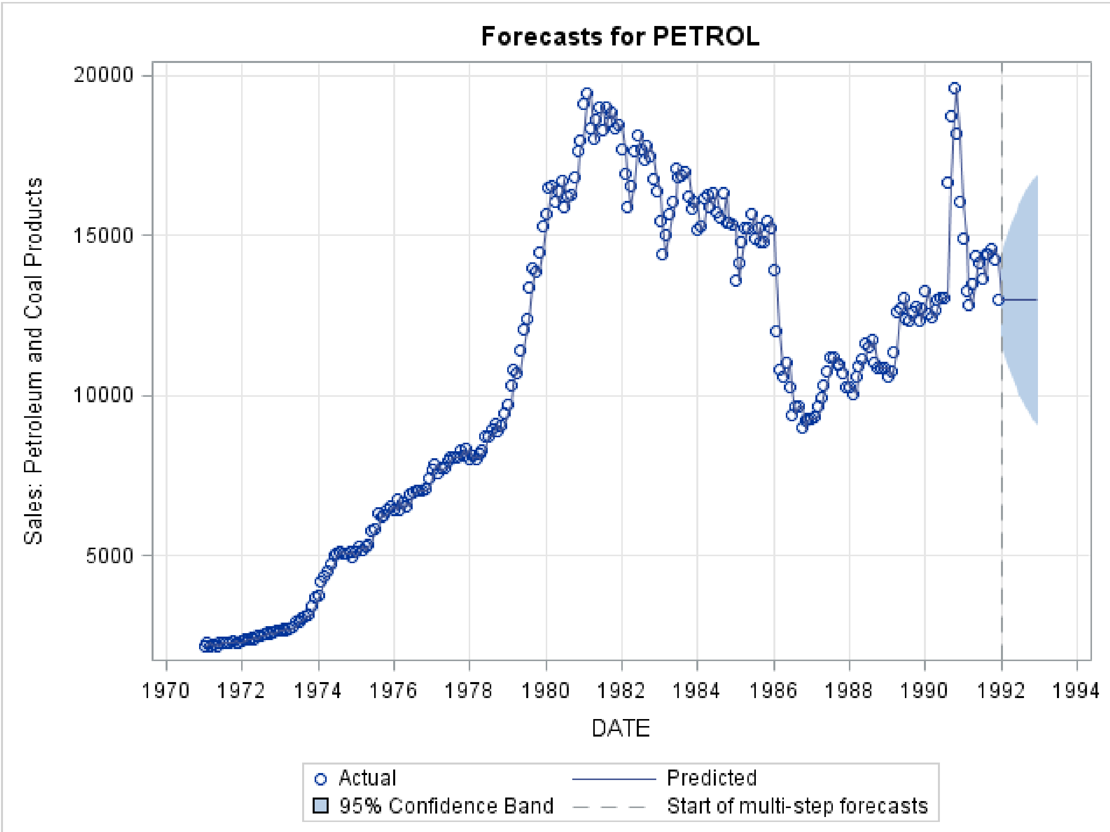
Output: 

TIMESERIES Procedure

b) Fit a Simple Exponential Smoothing model, predict 2 years (24 months) out and plot predictions.

```
proc esm data = usecon out = forecast lead = 24 plot = forecasts;
    id date interval = month;
    forecast petrol / model = simple;
run;

                             The ESM Procedure

                              Input Data Set

                  Name                             WORK.USECON
                  Label
                  Time ID Variable                        DATE
                  Time Interval                          MONTH
                  Length of Seasonal Cycle                  12
                  Forecast Horizon                          24

                           Variable Information

                  Name                                PETROL
                  Label
```

```
                    First                             JAN1971
                    Last                              DEC1991
                    Number of Observations Read           252
```

Output: 

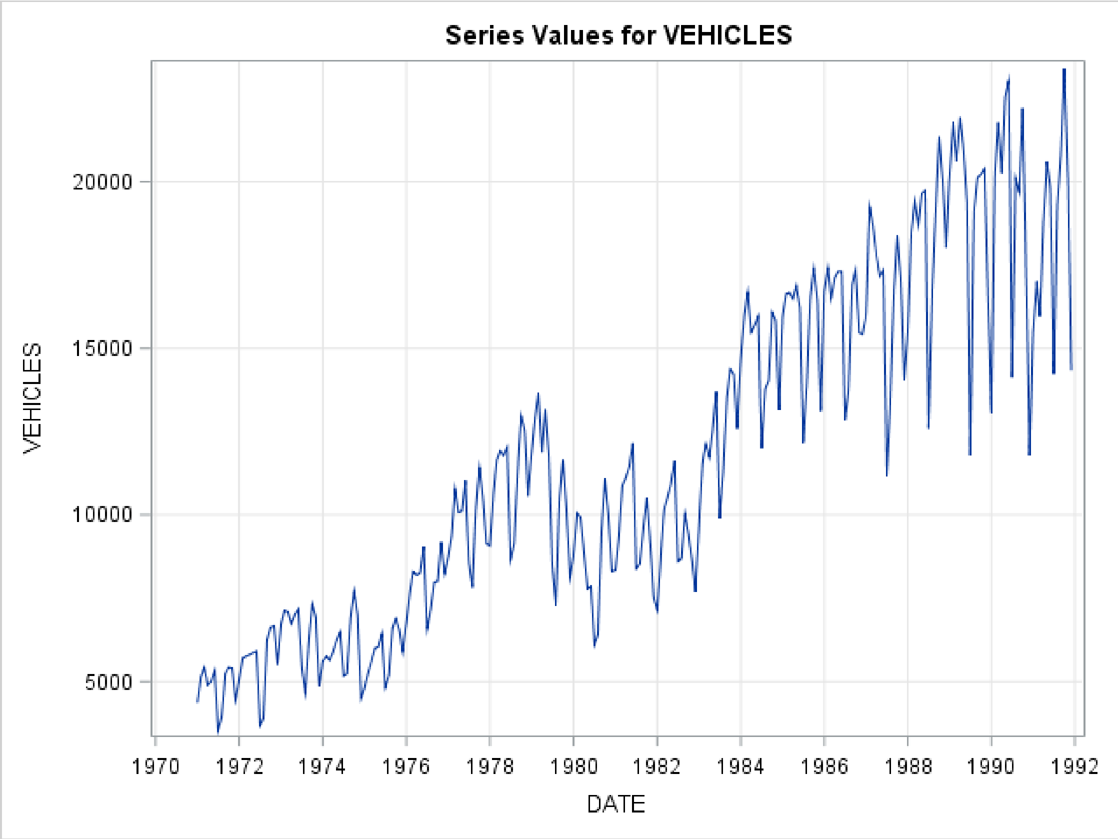
ESM Procedure

8.3 Fit a Holt-Winters model to a timeseries.

a) Plot the timeseries.

```
proc timeseries data = usecon plot = series;
    id date interval = month;
    var vehicles;
run;
```

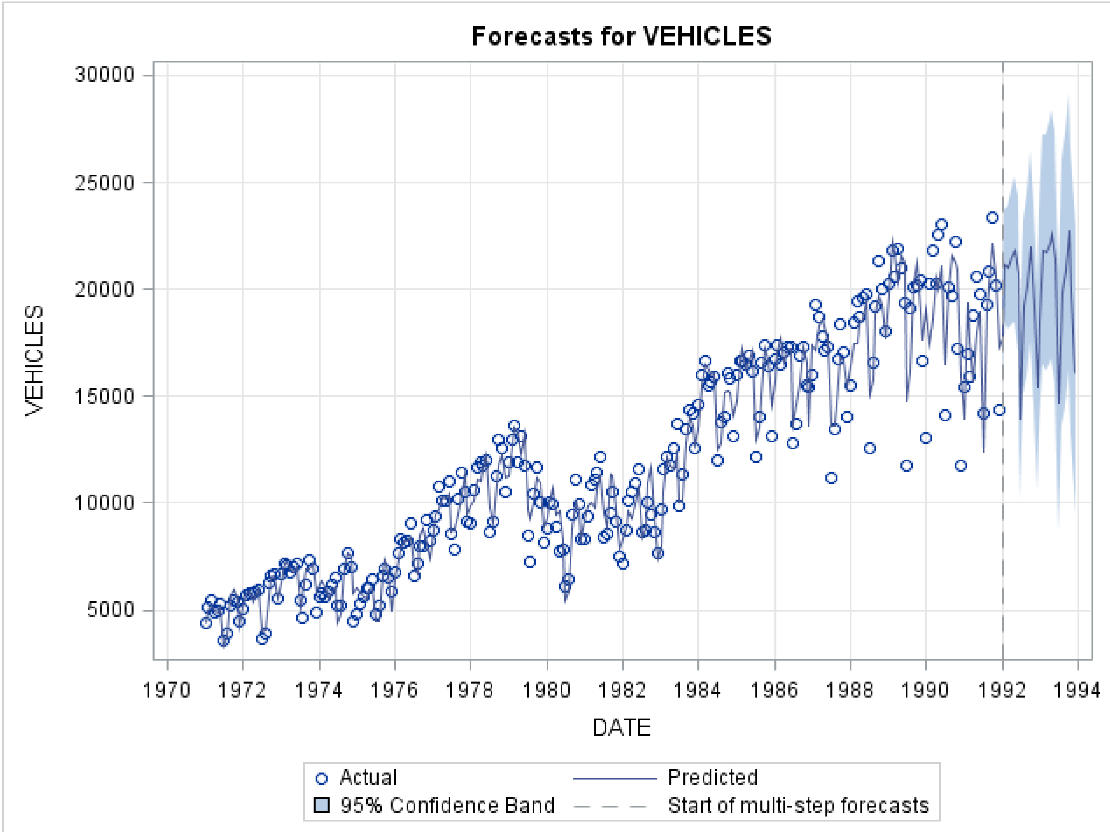
```
                         The TIMESERIES Procedure

                              Input Data Set

                  Name                          WORK.USECON
                  Label
                  Time ID Variable                     DATE
                  Time Interval                       MONTH
                  Length of Seasonal Cycle               12
```

Output: 

TIMESERIES Procedure

b) Fit a Holt-Winters additive model, predict 2 years (24 months) out and plot predictions.

```
proc esm data = usecon out = forecast lead = 24 plot = forecasts;
    id date interval = month;
    forecast vehicles / model = addwinters;
run;
```

Output: 

ESM Procedure

9 Model Evaluation & Selection

9.1 Evaluate the accuracy of regression models.

a) Evaluation on training data.

```
/* Notice we are re-using data sets but it is good to re-read the original
   version back into the environment */
proc import out = train
  datafile = 'C:/Users/boston_train.csv'
  dbms = csv replace;
  getnames = yes;
run;
proc import out = test
  datafile = 'C:/Users/boston_test.csv'
  dbms = csv replace;
  getnames = yes;
run;

/* Random Forest Regression Model */
ods select none;
```

```
proc hpforest data = train ;
    input _0-_12 / level = interval;
    target Target / level = interval;
    save file = 'rfMod.bin';
run;
ods select all;

/* Evaluation on training data  */
ods select none;
proc hp4score data = train;
    score file = 'rfMod.bin' out = scored_train;
run;
ods select all;

/* Determine coefficient of determination score */
proc iml;
  use scored_train;
    read all var _ALL_ into data;
  close scored_train;
  tip = data[,1];
  pred_rf = data[,2];
  r2_rf = 1 - ( (sum((tip - pred_rf)##2)) / (sum((tip - mean(tip))##2)) );
  print(r2_rf);
quit;

                                   r2_rf

                                 0.9781718
```

b) Evaluation on testing data.

```
/* Random Forest Regression Model (rfMod) */

/* Evaluation on testing data  */
ods select none;
proc hp4score data = test;
    score file = 'rfMod.bin' out = scored_test;
run;
ods select all;

/* Determine coefficient of determination score */
proc iml;
  use scored_test;
    read all var _ALL_ into data;
  close scored_test;
  tip = data[,1];
  pred_rf = data[,2];
  r2_rf = 1 - ( (sum((tip - pred_rf)##2)) / (sum((tip - mean(tip))##2)) );
  print(r2_rf);
quit;
```

```
r2_rf

0.8926887
```

The formula used here for the coefficient score is based off the Python skearn formula for r2_score.

HPFOREST Procedure | HP4SCORE Procedure | IML Procedure

9.2 Evaluate the accuracy of classification models.

a) Evaluation on training data.

```
proc import out = train
    datafile = 'C:/Users/digits_train.csv'
    dbms = csv replace;
    getnames = yes;
run;
proc import out = test
    datafile = 'C:/Users/digits_test.csv'
    dbms = csv replace;
    getnames = yes;
run;

/* Random Forest Classification Model */
ods select none;
proc hpforest data = train;
    input _0-_63 / level = interval;
    target Target / level = nominal;
    save file = 'rfMod.bin';
run;

/* Evaluation on training data */
proc hp4score data = train;
    score file = 'rfMod.bin' out = scored;
run;
ods select all;

data scored(keep = Target I_Target correct);
    set scored;
    correct = (I_Target = Target);
run;

/* Determine accuracy score */
proc iml;
    use scored;
      read all var _ALL_ into data;
    close scored;

    accuracy_forest = (1/nrow(data)) * sum(data[,2]);
```

```
    print(accuracy_forest);
quit;

                              accuracy_forest

                                            1
```

b) Evaluation on testing data.

```
/* Random Forest Classification Model (rfMod) */

/* Evaluation on testing data */
ods select none;
proc hp4score data = test;
    score file = 'rfMod.bin' out = scored;
run;
ods select all;

data scored(keep = Target I_Target correct);
    set scored;
    correct = (I_Target = Target);
run;

/* Determine accuracy score */
proc iml;
    use scored;
      read all var _ALL_ into data;
    close scored;

    accuracy_forest = (1/nrow(data)) * sum(data[,2]);

    print(accuracy_forest);
quit;

                              accuracy_forest

                                    0.9685185
```

The formula used here for the accuracy score is based off the Python skearn formula for accuracy_score.

HPFOREST Procedure | HP4SCORE Procedure | IML Procedure

9.3 Evaluation with cross validation.

a) KFold

The HPFOREST output for the 5 models has been removed from the output in this tutorial for the sake of space.

```
proc import out = breastcancer
    datafile = 'C:/Users/breastcancer.csv'
    dbms = csv replace;
    getnames = yes;
run;

data folds;
    set breastcancer;
    *randomly assign observation to one of K groups;
    call streaminit(29);
    rand=ceil(5*rand('UNIFORM'));
    output;
run;

%macro hp_KFolds();

data train1 test1 train2 test2 train3 test3
     train4 test4 train5 test5;
    set folds;
    %do i = 1 %to 5;
        %do j = 1 %to 5;
            if (rand = &j) then do;
                if (&i ^= &j) then output train&i;
                else output test&i;
            end;
        %end;
    %end;
    drop rand;
run;

%do i = 1 %to 5;

ods select none;
proc hpforest data = train&i;
    input _0-_29 / level = interval;
    target Target / level = nominal;
    save file = 'hpbreastcancer&i.bin';
run;

proc hp4score data = test&i;
    score file = 'hpbreastcancer&i.bin' out = scored_&i;
run;
ods select all;

data scored_&i;
    set scored_&i;
    correct = (I_Target = Target);
run;
```

```
proc freq data = scored_&i noprint;
  tables correct / out=FreqCount&i;
run;

%end;

%mend;

%hp_KFolds()

data FreqCount;
    set FreqCount1 FreqCount2 FreqCount3 FreqCount4 FreqCount5;
    if (correct = 1);
run;

proc means data = FreqCount mean std;
  var PERCENT;
run;
                            The MEANS Procedure

          Analysis Variable : PERCENT Percent of Total Frequency

                               Mean         Std Dev
                        ----------------------------
                          96.0918078       1.8699234
                        ----------------------------
```

HPFOREST Procedure | HP4SCORE Procedure | FREQ Procedure | MEANS Procedure

b) ShuffleSplit

The HPFOREST output for the 5 models has been removed from the output in this tutorial for the sake of space.

```
proc import out = breastcancer
    datafile = 'C:/Users/breastcancer.csv'
    dbms = csv replace;
    getnames = yes;
run;

proc surveyselect data = breastcancer out = cv seed = 29 samprate = 0.7
                                        outall reps = 5;
run;

data train1 train2 train3 train4 train5 test1 test2 test3 test4 test5;
     set cv;
     if (replicate = 1) then do;
        if (selected = 1) then output train1;
        else output test1;
```

```
   end;
   if (replicate = 2) then do;
        if (selected = 1) then output train2;
        else output test2;
   end;
   if (replicate = 3) then do;
        if (selected = 1) then output train3;
        else output test3;
     end;
   if (replicate = 4) then do;
        if (selected = 1) then output train4;
        else output test4;
   end;
   if (replicate = 5) then do;
        if (selected = 1) then output train5;
        else output test5;
     end;
run;

%macro hp_replicate();

%do i = 1 %to 5;

ods select none;
proc hpforest data = train&i;
    input _0-_29 / level = interval;
    target Target / level = nominal;
    save file = 'hpbreastcancer&i.bin';
run;

proc hp4score data = test&i;
    score file = 'hpbreastcancer&i.bin' out = scored_&i;
run;
ods select all;

data scored_&i;
    set scored_&i;
    correct = (I_Target = Target);
run;

proc freq data = scored_&i noprint;
  tables correct / out=FreqCount&i;
run;

%end;

%mend;

%hp_replicate()
```

```
data FreqCount;
    set FreqCount1 FreqCount2 FreqCount3 FreqCount4 FreqCount5;
    if (correct = 1);
run;

proc means data = FreqCount mean std;
  var PERCENT;
run;
                        The SURVEYSELECT Procedure

                Selection Method    Simple Random Sampling

                   Input Data Set           BREASTCANCER
                   Random Number Seed                 29
                   Sampling Rate                     0.7
                   Sample Size                       399
                   Selection Probability         0.70123
                   Sampling Weight                     0
                   Number of Replicates                5
                   Total Sample Size                1995
                   Output Data Set                    CV



                            The MEANS Procedure

          Analysis Variable : PERCENT Percent of Total Frequency

                             Mean         Std Dev
                       ----------------------------
                         95.7647059       0.6443795
                       ----------------------------
```

SURVEYSELECT Procedure | HPFOREST Procedure | HP4SCORE Procedure | FREQ Procedure | MEANS Procedure

---

10 Text Analytics

---

11 Deep Learning

Appendix

1 Built-in SAS Data Types

- CHAR The SAS implementation of a string as a fixed-length character string of length *n*.
- DOUBLE A decimal point number implemented as a 64-bit double precision, floating-point number.

2 SAS Procedures

ARIMA

CLUSTER

COMPARE

CONTENTS

CORR

DMDB

FCMP

ESM

EXPORT

FASTCLUS

FREQ

GENMOD

HP4SCORE

HPFOREST

HPSPLIT

HPSVM

IML

IMPORT

MEANS

NEURAL

PRINCOMP

PRINT

PLM

REG

SCORE

SGPLOT

- histogram
- inset
- reg
- scatter
- vbox

SGSCATTER

SORT

SQL

SURVEYSELECT

TIMESERIES

TREE

3 SAS DATA step

Statements:

%include

if-then/else

infile

input

merge

output

set

where

Alphabetical Index

Data Frame

A two-dimensional tabular structure with labeled axes (rows and columns), where data observations are represented by rows and data variables are represented by columns.

Dictionary

An associative array which is indexed by keys which map to values. Therefore, a dictionary is an unordered set of key:value pairs where each key is unique. In SAS, a dictionary can be implemented using a hash table. Please see the following example.

```
/* Results will be displayed in the log */
data class_dict;
declare hash mydict();
mydict.defineKey("Name");
mydict.defineData("Age");
mydict.defineDone();
do while (not eof);
  set sashelp.class end = eof;
  rc = mydict.add();
  output;
end;
Name = 'James';
rc = mydict.find();
put rc= Name= Age=;
```

Output:
```
rc=0 Name=James Age=12
rc=160038 Name=James Age=12
```

Series

A series is a one-dimension data frame, which is also called an array in SAS. Please see the following example.

```
array my_array{4} a1-a4 (1 3 5 9);
```

For more information on SAS packages and functions, along with helpful examples, please see SAS.