Module B8: Analyzing Data in SAS

Table of Contents

Now that data access, validation, and manipulation are behind you, you are ready to address the peak of the programming process: analyzing and reporting on the data. Analyzing your data can mean a lot of different things. It could be basic summarization to examine what happened in the past, or it could be complex data mining or machine learning algorithms to predict what might happen in the future. We are going to concentrate on reporting, summarizing, and analyzing using Statistical procedures.

Note: For some of the examples below I will be using The Little SAS Book, 6th Edition, some come from Learning SAS by Example By Ron Cody, and the majority come from SAS Institute Programming 1. For the examples from the SAS Institute the library used is called **pg1** (for Programming 1) rather than **mylib** (that I use for the other examples).

PROC TTEST

As you would expect from its name, the TTEST procedure, which is part of SAS/STAT software, computes t tests. You use t-tests when you want to compare means. Suppose, for example, that a statistics instructor selected a random sample of students to have extra tutoring. She could test whether the true mean of their scores was above a specific level (called a one sample test), she could compare the scores of students who had tutoring to those who did not (a two independent sample test), and she could compare the scores of students before and after tutoring (a paired test). PROC TTEST performs all these types of tests.

One sample comparisons

To compute a t test for a single mean, you list that variable in a VAR statement. SAS will test whether the mean is significantly different from H0, a specified null value. The default value of H0 is zero. You can specify a different value using the H0= option.

```
PROC TTEST H0=n <options>
      VAR col-name;
RUN;
```

Two independent sample comparisons

To compare two independent groups, you use a CLASS and a VAR statement. In the CLASS statement, you list the variable that distinguishes the two groups. In the VAR statement, you list the response variable. Note: for this type of comparison (using the syntax below) the data would need to be in a format where one variable is the response variable (numeric measurement) and another variable is the designation of which group/population the observation is from (character).

```
PROC TTEST H0=n <options>
        CLASS col-name;
        VAR col-name;
RUN;
```

Paired comparisons

When the variables you are comparing are paired, you use a PAIRED statement. The simplest form of this statement lists the two variables to be compared, separated by an asterisk. Note: the format of this type of analysis would be one variable of pre-scores and one variable of post-scores.

```
PROC TTEST <options>
        PAIRED col-name-1 * col-name-2;
RUN;
```

Options

ALPHA = n	specifies the level for the confidence limits. The value of n must be between 0 (100% confidence) and 1 (0% confidence). The default is 0.05 (95% confidence limits)
CI = type	specifies the type of confidence interval for the standard deviation. If you don't specify this option, then by default the value of type is EQUAL which produces and equal-tailed confidence interval. Other possible values are UMPU for an interval based on the uniformly most powerful unbiased test, and NONE to request no confidence interval for the standard deviation.
H0 = n	requests a test of the hypothesis H0 = n. The default value is 0.
NOBYVAR	moves the names of the variables from the title to the output table
SIDES = type	specifies whether the p-value and confidence interval are one or two-tailed. Possible values of type are 2 (the default) for two-tailed, L (for lower one-sided test) or U (for an upper one-sided test).

The TTEST procedure uses ODS Graphics to produce several plots that help you visualize your data including histograms, box plots, and Q-Q plots. Many plots are generated by default, but you can control which plots are created using the PLOTS option on the PROC TTEST statement. Here is the general form of the PROC TTEST statement with plot options:

```
PROC TTEST plots= (plot request-list);
        …
RUN;
```

Plot requests

The plots available to you depend on the type of comparison you request. Here are plots you can request for one sample, two sample, and paired t tests:

ALL	requests all appropriate plots.
BOXPLOT	creates box plots
HISTOGRAM	creates histograms overlaid with normal and kernel density curves
INTERVALPLOT	creates plots of confidence interval of means
NONE	suppresses all plots
QQPLOT	creates a normal quantile-quantile (Q-Q) plot
SUMMARYPLOT	creates one plot that includes both histograms and box plots.
AGREEMENTPLOT	creates agreement plots (for paired t tests only)
PROFILESPLOT	creates a profiles plot (for paired t tests only)

Excluding automatic plots

By default, the QQPLOT and SUMMARYPLOT are generated automatically for one sample, two sample and paired t tests. For paired t tests, the AGREEMENTPLOT and PRORFILESPLOT are also generated by default. If you choose specific plots in the plot-list, the default plots will still be created unless you add the ONLY global option:

```
PROC TTEST plots(only)= (plot request-list);
        …
RUN;
```

Example: The data in the **Olympic50Swim.txt** (found in Box folder) file gives the finishing times for semifinal and final races of the women's 50 meter freestyle swim. Each swimmer's initials are followed by their final time and semifinal time in seconds. Each line of data contains times for four swimmers.

The following program reads the raw data (**Olypics50swim.txt**) and uses a paired t test to test the mean difference between the semifinal and final times:

```
DATA mylib.swim;
    INFILE "/home/ehepfer/DSA 8030 - Fall 2020/Olympic50Swim.txt";
    INPUT Swimmer $ FinalTime SemiFinalTime @@;
RUN;

PROC TTEST DATA=mylib.swim;
    Title "50m Freestyle Semifinal vs. Final Results";
    PAIRED SemiFinalTime * FinalTime;
RUN;
```

50m Freestyle Semifinal vs. Final Results

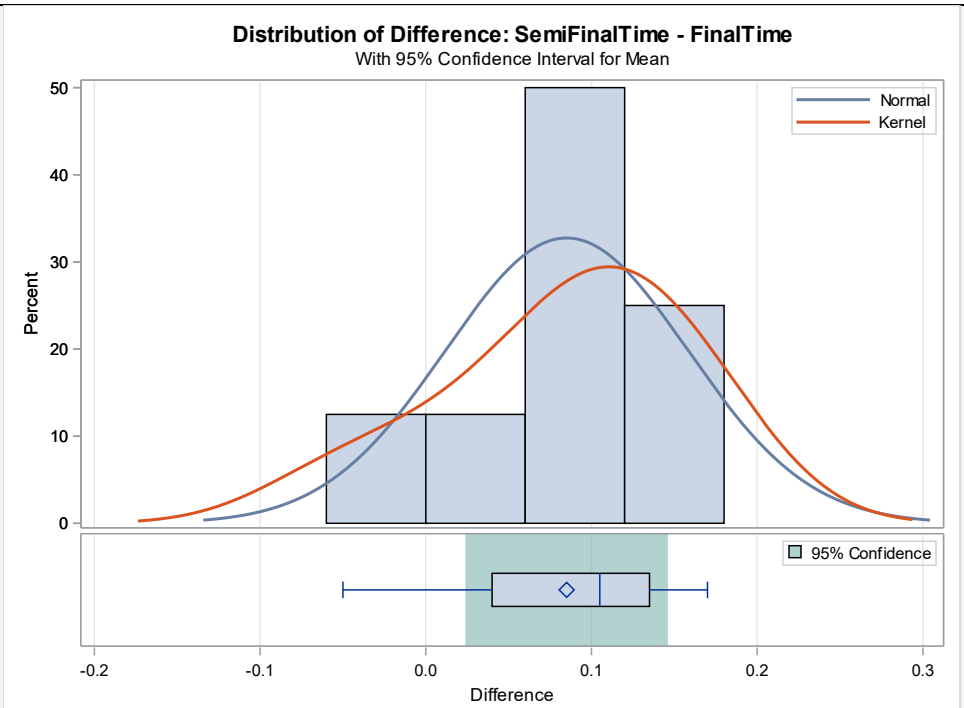
The TTEST Procedure

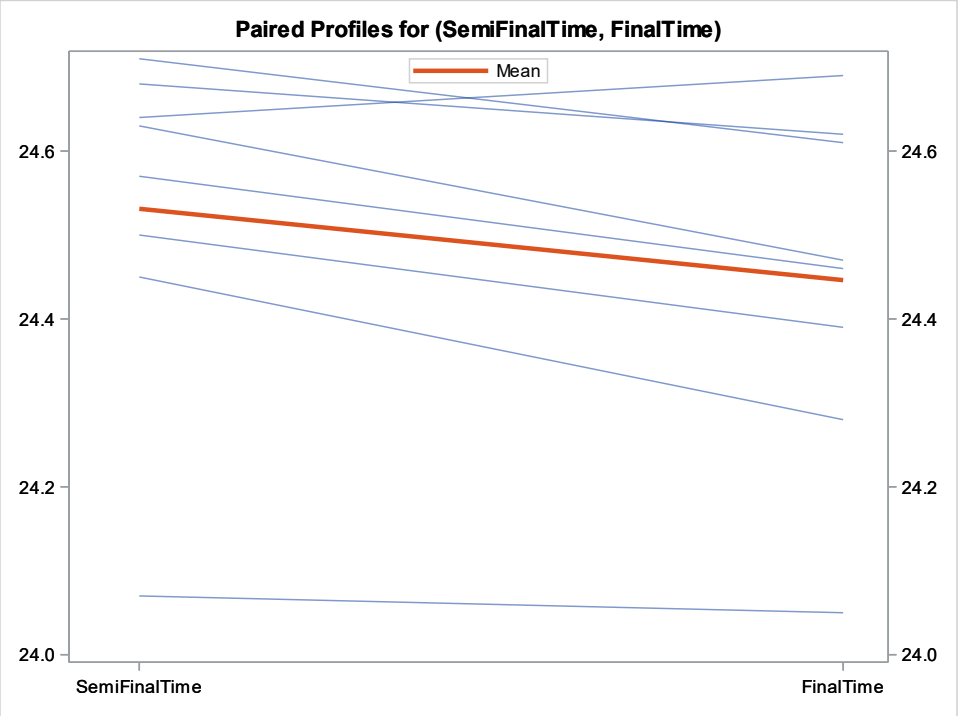
Difference: SemiFinalTime - FinalTime

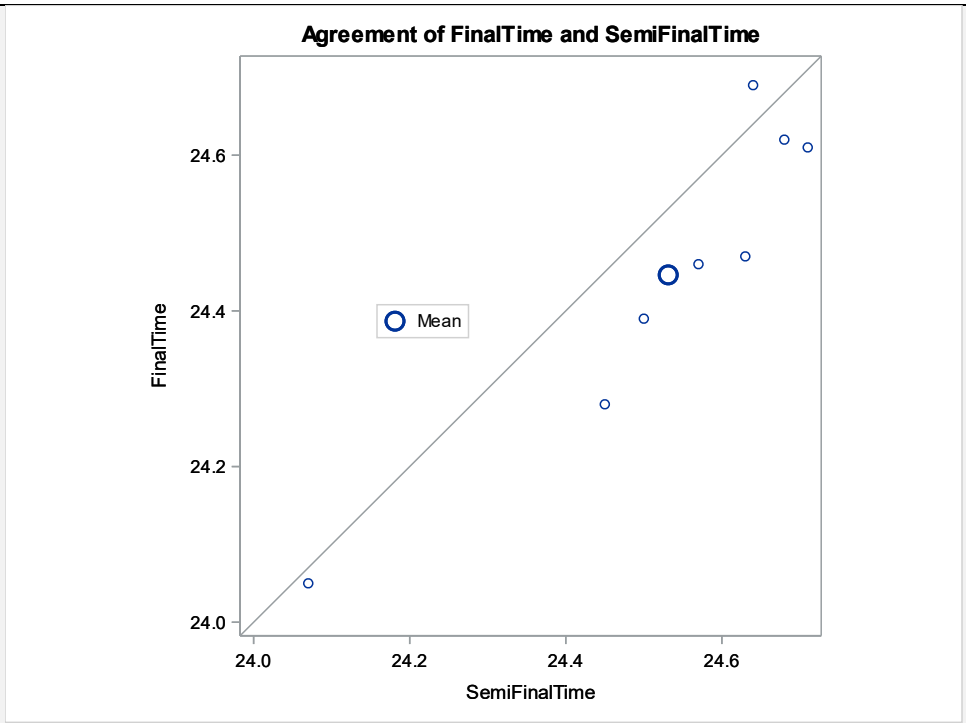
N	Mean	Std Dev	Std Err	Minimum	Maximum
8	0.0850	0.0731	0.0258	-0.0500	0.1700

Mean	95% CL Mean		Std Dev	95% CL Std Dev	
0.0850	0.0239	0.1461	0.0731	0.0483	0.1488

DF	t Value	Pr > |t|
7	3.29	0.0133


Distribution of Difference: SemiFinalTime - FinalTime
With 95% Confidence Interval for Mean
Normal
Kernel
95% Confidence
Percent
50
40
30
20
10
0
-0.2
-0.1
0.0
0.1
0.2
0.3
Difference


Paired Profiles for (SemiFinalTime, FinalTime)
Mean
24.6
24.4
24.2
24.0
SemiFinalTime
FinalTime


Agreement of FinalTime and SemiFinalTime
FinalTime
SemiFinalTime
24.0
24.2
24.4
24.6
Mean

PROC CORR

The CORR procedure computes correlations of variables. A correlation coefficient is a value between -1 and 1 that measures the linear relationship between variables. If two variables have no linear relationship the correlation coefficient (r) is 0. The close the absolute value of r is to 1 the stronger the linear relationship between the variables.

```
PROC CORR;
      VAR col-name(s);
      WITH col-name(s);
RUN;
```

By default, proc CORR computes correlations between all possible pairs of numeric variables.

VAR statement – variables in the VAR statement appear across the top of the table of correlations

WITH statement - variables listed in the WITH statement appear down the side of the table. (If no WITH statement is provided the procedure will use the variables in the VAR statement across the top and down the side.

Proc CORR automatically computes correlation using Pearson product-moment. There are options to have the Spearman, Hoeffding, or Kendall coefficients used instead.

This Example has a video

Reference: The Little SAS Book, 6th Edition – Section 9.8
Example: Each student in a statistics class recorded three values: test score, the number of hours spent watching videos in the week prior to the test, and the number of hours spent exercising during the same week. Here are the raw data:

Notice each line contains data for five students. The following program reads data from a file called **Exercise.dat** and the uses PROC CORR to compute the correlations:

```
56 6 2   78 7 4   84 5 5   73 4 0   90 3 4
44 9 0   76 5 1   87 3 3   92 2 7   75 8 3
85 1 6   67 4 2   90 5 5   84 6 5   74 5 2
64 4 1   73 0 5   78 5 2   69 6 1   56 7 1
87 8 4   73 8 3  100 0 6   54 8 0   81 5 4
78 5 2   69 4 1   64 7 1   73 7 3   65 6 2
```

```
DATA mylib.class;
    INFILE '/home/ehepfer/DSA 8030 - Fall 2020/Little SAS Book Examples/Exercise.dat';
    INPUT Score Videos Exercise @@;
RUN;

*Perform correlation analysis;

PROC CORR DATA=mylib.class Pearson SPEARMAN;
    VAR Videos Exercise;
    WITH Score;

    TITLE'Correlations and Test Scores';
    TITLE'With Hours of Videos and Exercise';
RUN;
```

With Hours of Videos and Exercise

The CORR Procedure

1 With Variables:	Score
2 Variables:	Videos Exercise

Simple Statistics

Variable	N	Mean	Std Dev	Median	Minimum	Maximum
Score	30	74.63333	12.58484	74.50000	44.00000	100.00000
Videos	30	5.10000	2.33932	5.00000	0	9.00000
Exercise	30	2.83333	1.94906	2.50000	0	7.00000

Pearson Correlation Coefficients, N = 30
Prob > |r| under H0: Rho=0

	Videos	Exercise
Score	-0.55390 0.0015	0.79733 <.0001

Spearman Correlation Coefficients, N = 30
Prob > |r| under H0: Rho=0

	Videos	Exercise
Score	-0.46801 0.0091	0.80556 <.0001

In this example, both hours of videos and hours of exercise are statistically significantly correlated with test score, but exercise is positively correlated while videos is negatively correlated. This means students who watched more videos tended to have lower scores, while the students who spent more time exercising tended to have higher scores.

PROC REG

Reference: The Little SAS Book, 6th Edition – Section 9.10

The REG procedure fits linear regression models by least squares and is one of many SAS procedures that perform regression analysis. PROC REG is part of SAS/STAT, which is licensed separately from Base SAS. We will show an example of a simple regression analysis using continuous numeric variables with only one regressor variable. However, PROC REG is capable of analyzing models with many regressor variables using a variety of model-selection methods, including stepwise regression, forward selection, and backward elimination. Other procedures in SAS/STAT will perform non-linear and logistic regression. In SAS/ETS, you will find procedures for time series analysis

The REG procedure has only two required statements. It must start with the PROC REG statement and have a MODEL statement specifying the analysis model.

```
PROC REG;
      MODEL dependent = independent;
RUN;
```

This Example has a video

Reference: The Little SAS Book, 6th Edition – Section 9.10

Example: At your young neighbor's T-ball game (that's where the players hit the ball from the top of a tee instead of having the ball pitched to them), he said to you, "You can tell how far they'll hit the ball by how tall they are." To give him a little practical lesson in statistics, you decide to test his hypothesis. You gather data from 30 players, measuring their height in inches and their longest of three hits in feet. The following are the data. Notice that data (**Baseball.dat)** for several players are listed on one line:

```
50 110  49 135  48 129  53 150  48 124 50 143  51 126  45 107
53 146  50 154  47 136  52 144  47 124 50 133  50 128  50 118
48 135  47 129  45 126  48 118  45 121 53 142  46 122  47 119
51 134  49 130  46 132  51 144  50 132 50 131
```

The following program reads the data and performs the regression analysis. In the MODEL statement, Distance is the dependent variable, and Height is the independent variable.

```
DATA mylib.hits;
    INFILE '/home/ehepfer/DSA 8030 - Fall 2020/Little SAS Book Examples/Baseball.dat';
    INPUT Height Distance @@;
RUN;

*Perform regression analysis;

PROC REG DATA=mylib.hits PLOTS(ONLY) = (DIAGNOSTICS FITPLOT);
    MODEL Distance = Height;
    TITLE 'Results of Regression Analysis';
RUN;
```

Results of Regression Analysis

The REG Procedure
Model: MODEL1
Dependent Variable: Distance

Number of Observations Read	30
Number of Observations Used	30

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	1365.50831	1365.50831	16.86	0.0003
Error	28	2268.35836	81.01280		
Corrected Total	29	3633.86667			

Root MSE	9.00071	R-Square	0.3758
Dependent Mean	130.73333	Adj R-Sq	0.3535
Coeff Var	6.88479		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-11.00859	34.56363	-0.32	0.7525
Height	1	2.89466	0.70506	4.11	0.0003

PROC SGPLOT

Reference: Little SAS Book 6th Edition – Section 8.2-8.5

Bar Charts

Bar charts show the distribution of a categorical variable where the length of each bar is proportional to the number of observations in that category. To create a chart with vertical bars, use a VBAR statement with this general form:

```
PROC SGPLOT;
    VBAR col-name </options>;
RUN;
```

For horizontal bars, replace the keyword VBAR with HBAR.

Possible Options include

ALPHA=n	specifies the level for the confidence limits. The value of n must be between 0 (100% confidence) and 1 (0% confidence). The default is 0.05 (95% confidence limits).
BARWIDTH = n	set the width of bars. Values range from 0.1 to 1 with a default of 0.8.
DATALABEL = variable-name	displays a label for each bar. If you specify a variable name, then values of that variable will be used. Otherwise, SAS will calculate appropriate values.
DISCRETEOFFSET = n	offests bars from midpoints which is useful for overlaying bar charts. The value must be between -0.5 (left) and +0.5 (right). The default is 0 (no offset).
LIMITSTAT = statistic	specifies the type of limit lines to be shown. Possible values are CLM, STDDEV (standard deviation), or STDERR (standard error). This option cannot be used with the GROUP=option. You must specify a RESPONSE= option and STAT=MEAN.
MISSING	includes a bar for missing values
GROUP = variable-name	specifies a variable used to group the data.
GROUPDISPLAY = type	specifies how to display grouped bars, either STACK (the default) or CLUSTER.
RESPONSE = variable-name	specifies a numeric variable to be summarized.
STAT = statistic	specifies a statistic, either FREQ, MEAN, or SUM. FREQ is the default if there is no response variable. SUM is the default when you specify a response variable.

TRANSPARENCY = n	specifies the degree of transparency for the bars. The value of n must be between 0 (the default) and 1, with 1 being completely transparent and 0 being completely opaque.

This Example has a video

Reference: Little SAS Book 6th Edition – Section 8.2

Example: A chocolate manufacturer is considering whether to add four new varieties of chocolate to its line of products. The company asked volunteers to taste the new flavors. The data (found in **Choc.dat**) contain each person's age group (A for adult, C for child) followed by their favorite flavor (80%Cacao, Earl Grey, Ginger, or Pear).

```
A Pear
A 80%Cacao
A EarlGrey
C 80%Cacao
A Ginger
C Pear
C 80%Cacao
C Pear
C Pear
A EarlGrey
A 80%Cacao
C 80%Cacao
A Ginger
A Pear
C EarlGrey
C 80%Cacao
A 80%Cacao
A EarlGrey
A 80%Cacao
C Pear
C Pear
A 80%Cacao
C Pear
C 80%Cacao
```

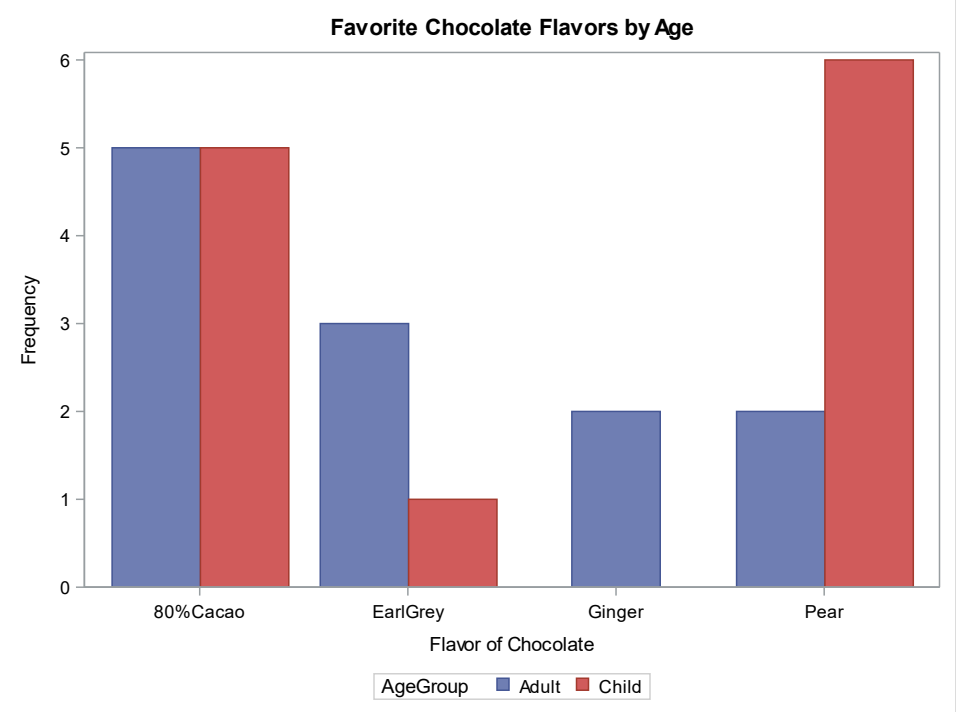
SAS code:

```
DATA chocolate;
    INFILE "/home/ehepfer/DSA 8030 - Fall 2020/Choc.dat";
    INPUT AgeGroup $ FavoriteFlavor $;
Run;

/*create temporary format*/
PROC FORMAT;
    VALUE $AgeGp 'A' = 'Adult' 'C' = 'Child';
Run;

/* Bar chart for favorite flavor*/
PROC SGPLOT data=chocolate;
    VBAR FavoriteFlavor/ GROUP = AgeGroup GROUPDISPLAY = CLUSTER;
    FORMAT AgeGroup $AgeGp.;
    LABEL FavoriteFlavor = 'Flavor of Chocolate';
    TITLE 'Favorite Chocolate Flavors by Age';
Run;
```

Output:



The chart has clustered bars showing the number of respondents in each age group who chose each flavor. The LABEL statement replaced the name of the variable Favorite Flavor with the words "Flavor of Chocolate" in the X axis label. The FORMAT statement replaced the data values (A and C) with more descriptive values (Adult and Child) in the legend.

Histograms and Density Curves

To show the distribution of continuous data, you can use histograms (or box plots which are next). In a histogram, the data are divided into discrete intervals called bins. Each bin is represented by a rectangle which makes histograms look similar to bar charts. However, bar charts typically have a gap between the bars while histograms do not. To create a histogram, use a HISTOGRAM statement with this basic form:

```
PROC SGPLOT;
      HISTOGRAM col-name </options>;
RUN;
```

Possible options include:

BINSTART = n	specifies the midpoint for the first bin
BINWIDTH = n	specifies the bin width (in units of the horizontal axis). SAS determines the number of bins. This option is ignored if you specify the NBINS= option.
NBINS = n	specifies the number of bins. SAS determines the bin width
SCALE = scaling-type	specifies the scale for the vertical axis, either PERCENT (the default), COUNT, or PROPORTION
SHOWBINS	places tick marks at the midpoints of the bins. By default, tick marks are placed at regular intervals based on minimum and maximum values.
TRANSPARENCY = n	specifies the degree of transparency for the histogram. The value of n must be between 0 (the default) and 1, with 1 being completely transparent and 0 being completely opaque.

DENSITY CURVES You can also plot density curves for your data. The basic form of a DENSITY statement is

```
PROC SGPLOT;
      DENSITY col-name </options>;
RUN;
```

Possible options:

TYPE=distribution type	specifies the type of distribution curve, either NORMAL (by default) or KERNEL

TRANSPARENCY = n	specifies the degree of transparency for the density curve. The value of n must be between 0 (the default) and 1, with 1 being completely transparent and 0 being completely opaque.

The HISTOGRAM and DENSITY statements can be used together, but not with other types of graphs. Keep in mind that when you overlay graphs, the order of statements is important because the second graph will be drawn on top of the first and could hide it.

This Example has a video

Reference: Little SAS Book 6th Edition – Section 8.3

Example: A fourth graph class has a competition to see who can read the most books in one month. For each student, the teacher records the student's name and the number of books read. Notice that each line of data (**Reading.txt)** includes six students.

```
Belia 4 Anthony 9 Joe 10 Chirs 6 BEth 5 Daniel 2
David 7 Emily 7 Josh 7 Will 9 Olivia 7 Matt 8
Maddy 8 Sam 13 Jessica 6 Jose 6 Mia 12 Elliott 8
Tyler 15 Lauren 10 Cate 14 AVa 11 Mary 9 Eric 10
Megan 13 Michael 9 John 18 Alex 5 Cody 11 Amy 4
```

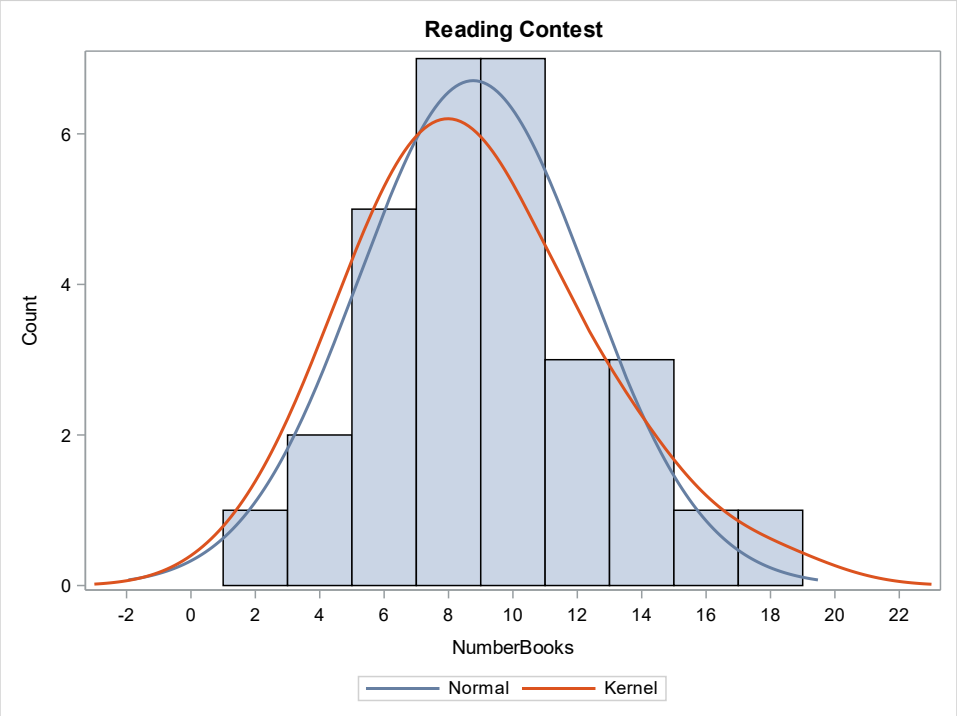
SAS Code:

```
DATA contest;
    infile "/home/ehepfer/DSA 8030 - Fall 2020/Reading.txt";
    input Name $ NumberBooks @@;
RUN;

/* Create histogram and density curve */

PROC SGPLOT data = contest;
    HISTOGRAM NumberBooks / BINWIDTH = 2 SHOWBINS SCALE = COUNT;
    DENSITY NumberBooks;
    DENSITY NumberBooks / TYPE = Kernel;
    Title 'Reading Contest';
RUN;
```

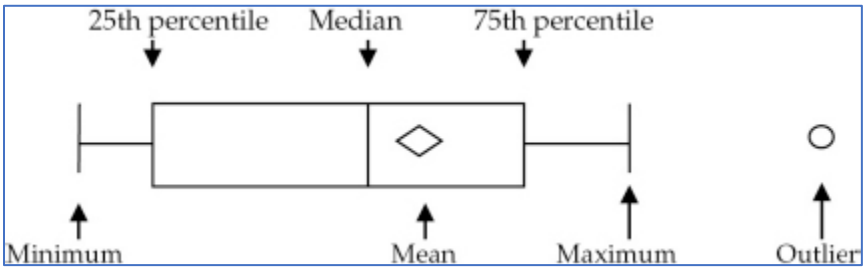
Output:



Boxplots

Like histograms, box plots show the distribution of continuous data. This type of graph is also called a box-and-whisker plot (or box plot) because of the way it looks. Every part of a box plot tells you something about the distribution of your data.

The ends of the box indicate the 25th and 75th percentiles. The line inside the box indicates the 50th percentile (the median), and the marker indicates the mean. By default, the whiskers cannot be longer than 1.5 times the length of the box. Any points beyond the whiskers are considered outliers and are marked with circles. If you specify the EXTREME option, then the whiskers will extend the entire range.



To create a horizontal box plot, use a HBOX statement like this:

```
PROC SGPLOT;
      HBOX col-name </options>;
RUN;
```

For vertical box plots, replace the keyword HBOX with VBOX.

Possible options include

CATEGORY = variable-name	specifies a categorical variable. One box plot will be created for each of this variable.
EXTREME	specifies that the whiskers should extend to the true minimum and maximum vales so outliers will not be identified.
GROUP = variable-name	specifies a second categorical variable. One box plot will be created for each value of this variable within the categorical variable.
MISSING	includes a box for missing values for the group or category variable
TRANSPARENCY = n	specifies the degree of transparency for the box plot. The value of n must be between 0 (the default) and 1, with 1 being completely transparent and 0 being completely opaque.

The VBOX and HBOX statements cannot be used with statement that create other types of plots.

This Example has a video

Reference: Little SAS Book 6th Edition – Section 8.4

Example: A small town sponsors an annual bicycle criterium. That's a race where bicyclists go round and round a loop. The racers compete in three divisions: Youth, Adult, and Masters. The data (**Criterium.txt)** include each bicyclist's division and the number of laps they completed in one hour. Notice that each line of data contains results for five competitors.

```
Adult 44 Adult 33 Youth 33 Masters 38 Adult 40
Masters 32 Youth 32 Youth 38 Youth 33 Adult 47
Masters 37 Masters 46 Youth 34 Adult 42 Youth 24
Masters 33 Adult 44 Youth 35 Adult 49 Adult 38
Adult 39 Adult 42 Adult 32 Youth 42 Youth 31
Masters 33 Adult 33 Masters 32 Youth 37 Masters 40
```

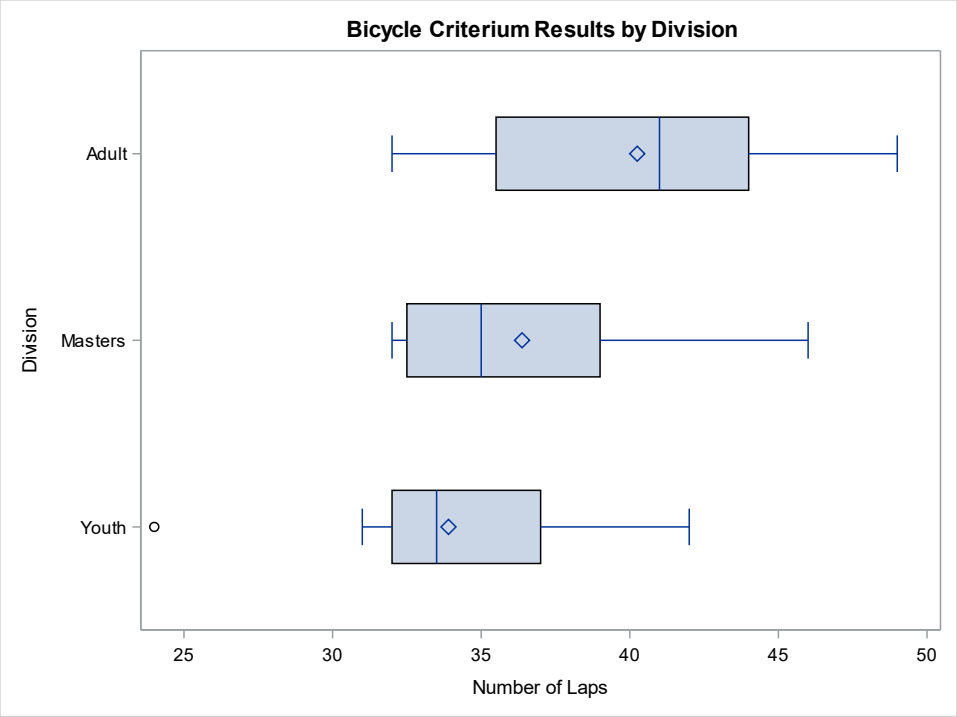
SAS Code:

```
DATA bikerace;
    infile "/home/ehepfer/DSA 8030 - Fall 2020/Criterium.txt";
    input Division $ NumberLaps @@;
RUN;

/* Create box plot */

PROC SGPLOT data =bikerace;
    HBOX NumberLaps / CATEGORY = Division;
    TITLE 'Bicycle Criterium Results by Division';
    LABEL NumberLaps = 'Number of Laps';
RUN;
```

Output:



Scatter Plots

Scatter plots are an effective way to show the relationship between two continuous variables. For experimental data, the independent variable is traditionally assigned to the horizontal axis while the dependent variable is assigned to the vertical axis. To create scatter plots, use the SCATTER statement like this:

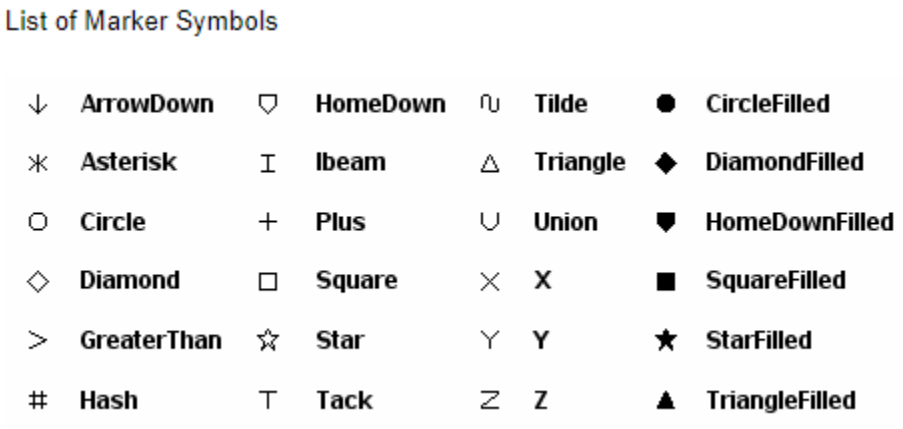
```
PROC SGPLOT;
    SCATTER X=horizontal-variable Y=vertical-variable </options>;
RUN;
```

Possible options include:

DATA LABEL = variable-name	displays a label for each data point. If you specify a variable name, the vales of that variable will be used as labels. If you do not specify a variable name, then the value of the Y variable will be used.
GROUP = variable-name	specifies a variable to be used for grouping data.
NOMISSINGGOUP	specifies that observations with missing values for the group variable should not be included
TANSPARENCY = n	specifies the degree of transparency for the markers. The value of n must be between 0 (the default) and 1, with 1 being completely transparent and 0 being completely opaque.

You can change the colors and marker symbols by using a STYLEATTRS statement. To change symbols use the DATASYMBOLS options. To change colors use the DATACONTRASTCOLORS option.

Below is a picture of the possible marker symbols:



This Example has a video

Reference: Little SAS Book 6th Edition – Section 8.5

Example: To illustrate the use of scatter plots, here are data (**Birds.txt)** about birds. For each species there are four variables: name, type (S for songbirds or R for raptors), length in cm (from tip of beak to tip of tail), and wingspan in cm.

```
Robin,S,28,41
Bald Eagle,R,102,244
Barn Owl,R,50,110
Osprey,R,66,180
Cardinal,S,23,31
Godfinch,S,11,19
Golden Eagle,R,100,234
Crow,S,52,100
Magpie,S,60,60
Elf Owl,R,15,27
Condor,R,140,300
```

Produce a scatter plot using Wingspan as x and Length as y grouped by type.

SAS Code:

```
DATA wings;
    infile "/home/ehepfer/DSA 8030 - Fall 2020/Birds.txt" dlm=',';
    input @1 Name $ Type $ Length Wingspan;
RUN;

/* Create temporary format for S and R */
PROC FORMAT;
    value $birdtype
        'S' = 'Songbirds'
        'R' = 'Raptors';
RUN;

PROC SGPLOT data=wings;
    scatter x = Wingspan Y = Length / Group = Type;
    format type $birdtype.;
    STYLEATTRS datasymbols= (CircleFilled Square) datacontrastcolors=(Orange Purple);
    title 'Comparison of Wingspan vs. Length';
RUN;
```

Output: