Summer B Webinars in Psychometrics and Statistics

Introduction to SAS

Jue Wang, *Ph.D.*

August 03, 2020

Outline
I. The SAS Language
II. Interface of SAS/SAS Studio
III. Import SAS Datasets
IV. SAS Graphs

I. The SAS Language

It began in the late 60's and early 70's as a statistical package. SAS originally stood for Statistical Analysis System. However, unlike many competing statistical packages, SAS is also an extremely powerful, general-purpose programming language. SAS is a predominant software in the pharmaceutical industry and most Fortune 500 companies. In recent years, it has been enhanced to provide state-of-the-art data mining tools and programs for Web development and analysis.

SAS uses statements to write a series of instructions called a **SAS program**. The primary purpose of each SAS program is to process and analyze data through a collection of statements. The only way to learn a programming language is to practice (make errors, correct the errors, and make new errors). Once you understand and master all the "words" (SAS commands), you can write good "essays" (SAS program).

We will use SAS Studio supported by the SAS University Edition (Mac OS X). The full version – SAS Educational Analytics has more features than the University Edition. Only the interface differs a little.

II. Interface of SAS/SAS Studio

The main window of SAS Studio consists of a navigation pane on the left and a work area on the right.

a. **Explorer/Navigation window** – find your program and datasets here.

The navigation pane provides access to your server files and folder shortcuts, your tasks and snippets, the libraries that you have access to, and your file shortcuts. The Server Files and Folders section is displayed by default.

The work area is used to display your data, code, tasks, logs, and results.

b. **Code editor window** – type your SAS program here.

SAS Studio includes a color-coded, syntax-checking editor for editing new or existing SAS programs. You can also edit SOURCE entries in SAS catalogs. The editor includes a wide variety of features such as autocompletion, automatic formatting, and pop-up syntax help. With the code editor, you can write, run, and save SAS programs. You can also modify and save the code that is automatically generated when you run a task. The menu bar in the code editor window. Hover over and read each one.

c. **Log window** – contains messages by SAS keeping notes for all analysis and showing errors and warning.
d. **Results window (Results)** – display the output tables and figures

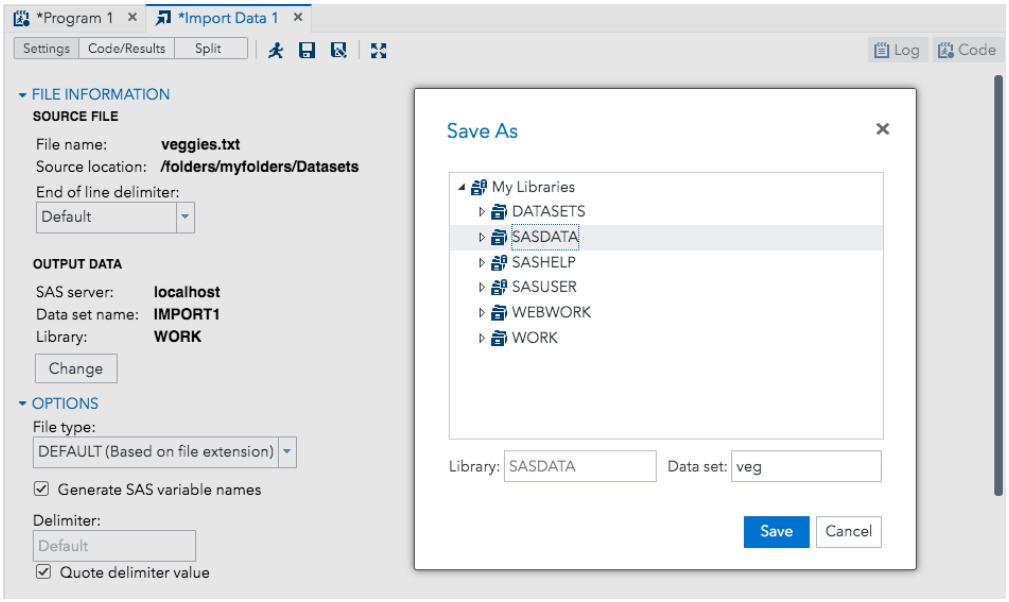
Make good use of the SAS Studio users' guide when you have questions:
https://documentation.sas.com/api/docsets/webeditorug/3.8/content/webeditorug.pdf?locale=en

III. Import/Export SAS Datasets

We will use *veggies.txt* in this example.

a. Using Import Data feature in SAS Studio

Next to the SAS program, we can see **Import Data**. This is a nice feature of SAS studio. Click on Import Data, drag (from My Folders on the left panel) and drop your data file, or click select file. Then click on settings, Change the Data set to veg.



Remember to check the OPTIONS – Generate SAS variable names. If the data file contains the variable names on the first line, check this box (leave it as default); otherwise, please de-select this box.

Then click on Run (click on little running man on the top panel). There are multiple ways to check if the dataset is loaded properly.

- See Log window: is there any error or warning?
- See Results window: there should be tables showing the attributes of the data file.
- Open the SAS dataset directly: My libraries → SASDATA → double click on VEG.

b. Use INFILE command to import a text file

A traditional way to import a txt datafile is to use INFILE command. In the Code Editor, we can start writing statements.

```
*Read in the dataset from a txt file and convert it to a SAS dataset*;
data veggies;
    infile "/folders/myfolders/Datasets/veggies.txt";
    input name $ code $ days number price;
    costperseed = price/number;
run;
```

Read data using a DATA step

- Reading dataset or changing variables in a dataset – we use DATA step.
- The first line (data veggies) defines the name of the SAS dataset.
- **INFILE** specifies the path where the file is located. If the data lines start from the second line (first line is the variable name), we add **FIRSTOBS=2** right after the path quote telling SAS to start reading from the 2nd line.
- **INPUT** specifies the variables. By default, SAS treats all variables as numeric. If reading character/string variables, put a $ after the variable name.
- **RUN** is required for all procedures/steps – telling SAS to execute the commands.

Change/create variable in a DATA step

- In the example, we create a new variable (cost per seed) as price divided by number.
- Various operations can be specified in a DATA step. For instance,
    - DROP statement: remove any variables from a data set;
    - KEEP statement: instead of dropping, it tells SAS which variables to keep;
    - See SAS Help document for a list of operators. Here is a snapshot.

Arithmetic Operators

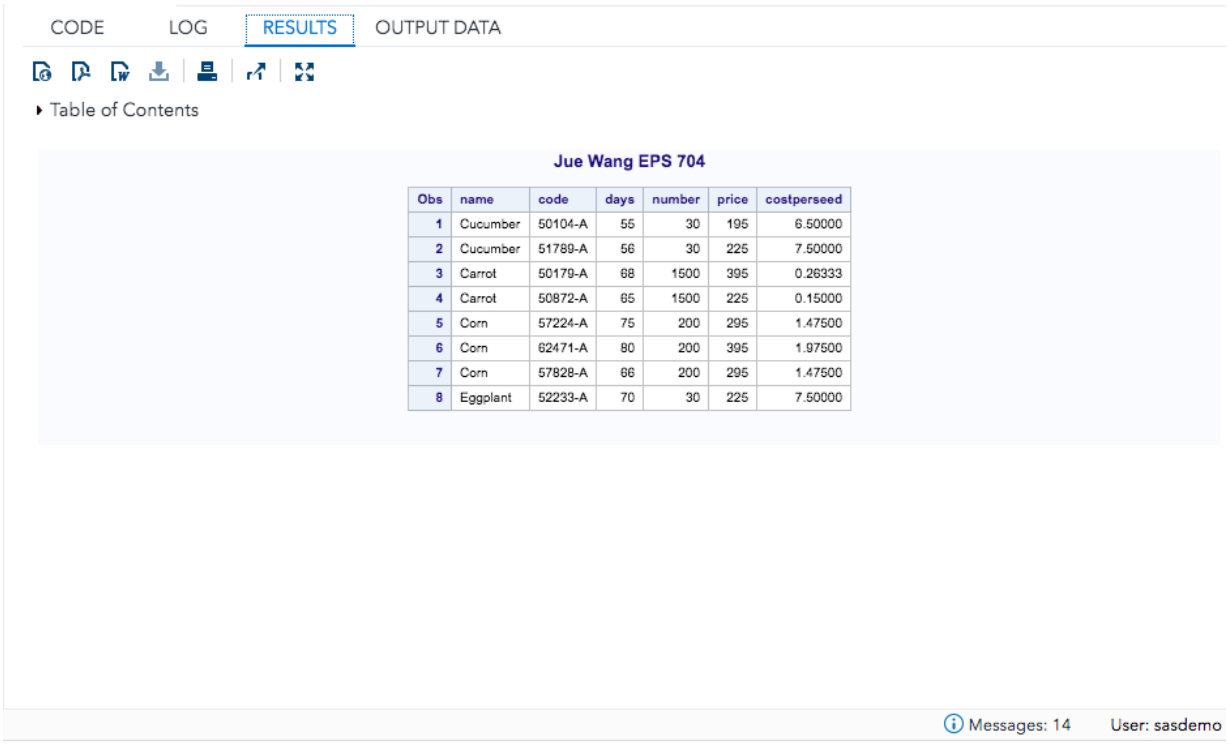
Symbol	Definition	Example	Result
**	exponentiation	a**3	raise A to the third power
*	multiplication [1]	2*y	multiply 2 by the value of Y
/	division	var/5	divide the value of VAR by 5
+	addition	num+3	add 3 to the value of NUM
-	subtraction	sale-discount	subtract the value of DISCOUNT from the value of SALE

1 The asterisk (*) is always necessary to indicate multiplication; 2Y and 2(Y) are not valid expressions.

c. Print data using a procedure called PROC PRINT. **PROC** stands for procedure.

```
proc print data=sasdata.veggies;
run;
```

The output table appears in the Results window. Highlight/Select the codes and hit run (running man). Remember always check the data either in the library, OUTPUT DATA window (in SAS studio) or using PROC PRINT to display in an output table (RESULTS window) to make sure they were read in correctly!



Obs	name	code	days	number	price	costperseed
1	Cucumber	50104-A	55	30	195	6.50000
2	Cucumber	51789-A	56	30	225	7.50000
3	Carrot	50179-A	68	1500	395	0.26333
4	Carrot	50872-A	65	1500	225	0.15000
5	Corn	57224-A	75	200	295	1.47500
6	Corn	62471-A	80	200	395	1.97500
7	Corn	57828-A	66	200	295	1.47500
8	Eggplant	52233-A	70	30	225	7.50000

Notes:

- **Semicolons conclude every complete statement in SAS!**
- The colors are helpful for detecting coding errors.
    - Comments begin with a * and are color coded in green.
    - DATA or PROC calls (define keywords) are in dark blue
    - Statement in the quotes are purple.
    - Numeric constants are teal.

IV. SAS Graphs – PROC SGPLOT

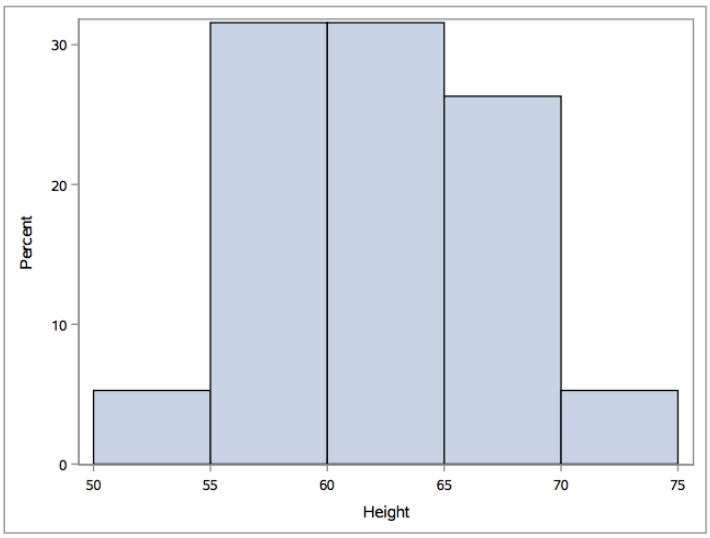
a. Histogram – Frequency distribution of continuous variable (Height)

- PROC SGPLOT line: identify the dataset.
- HISTOGRAM statement: simply specify the variable name.

Code Editor:

```
*Simple Histogram for Height*;
proc sgplot data=sashelp.class;
  histogram height;
run;
```

Results Window:

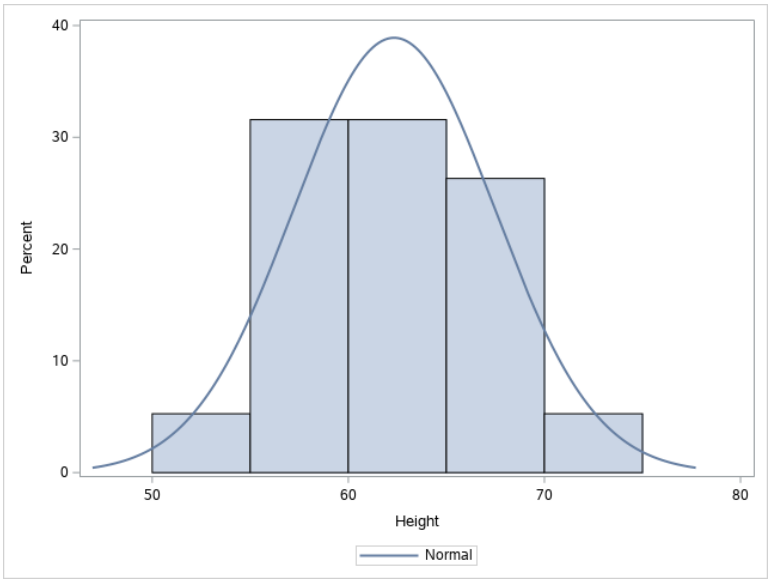


We can also add a normal density curve using DENSITY statement.

Code:

```
*Histogram for Height with normal density curve*;
proc sgplot data=sashelp.class;
  histogram height;
  density height;
run;
```

Results:



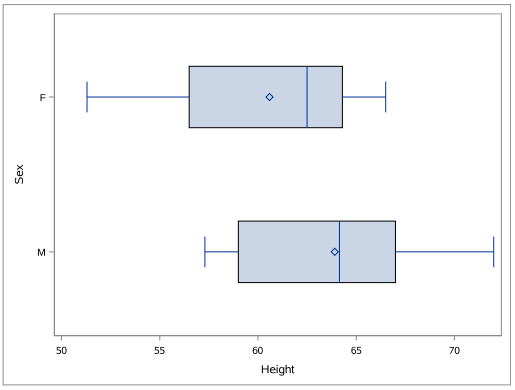
b. Boxplot – Frequency distribution of continuous variable (Height by Gender)

Code:

```
*Boxplot -- Height by gender*;
proc sgplot data=sashelp.class;
  hbox height / category=sex;
run;
```

- HBOX statement: specify the continuous variable
- CATEGORY option: identify the groups or categories

Results:

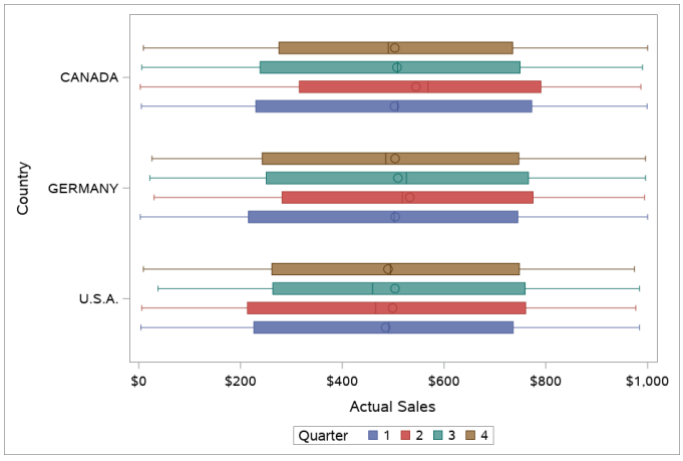


- GROUP option: add another categorical variable.

Let's look at a different dataset (sashelp.prdsale) showing furniture sales data. Display actual sale by country in each quarter. Code:

```
*Boxplot -- actual sale by country per quarter*;
proc sgplot data=sashelp.prdsale;
  hbox actual / category=country group=quarter;
run;
```

Results:

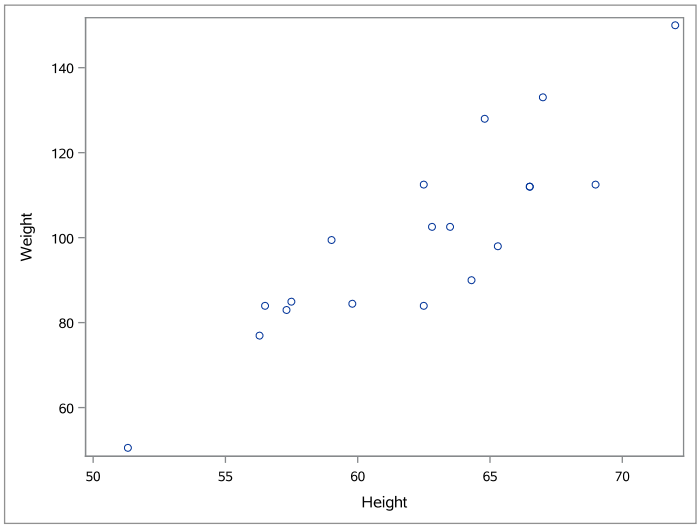


c. Scatterplot – Relationship between two continuous variables

a. Using SCATTER statement: specify variables on the x-axis and y-axis separately. Code:

```
*Scatterplot -- Height by Weight*;
proc sgplot data=sashelp.class;
  scatter x=height y=weight;
run;
```

Results:



b. MARKERATTRS Option for either REG or SCATTER statement.

Marker options specify the appearance of the markers in the plot. Usually we specify color, size, and symbols.

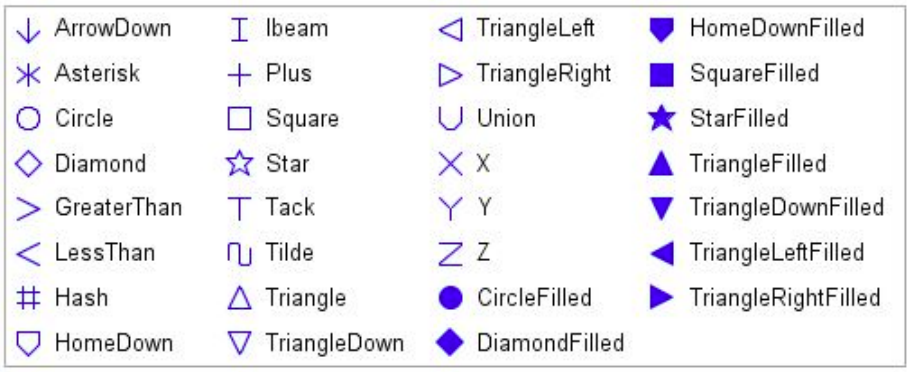
Code:

```
*Scatterplot with user defined markers -- Height by Weight*;
proc sgplot data=sashelp.class;
  scatter x=height y=weight / markerattrs=(color=red size=10 symbol=trianglefilled);
run;
```

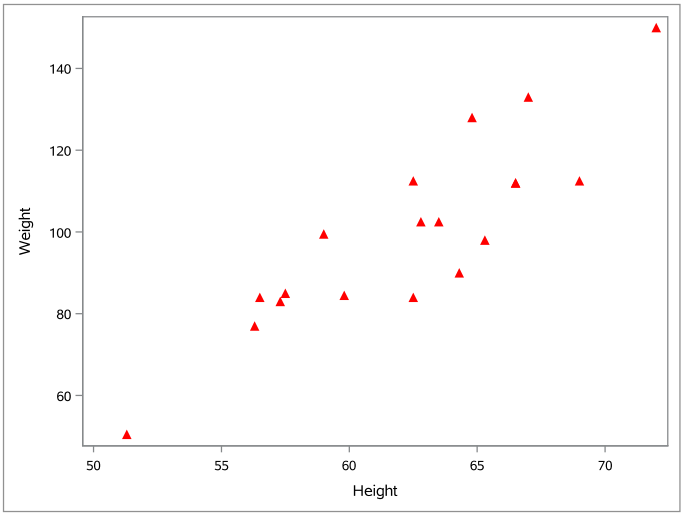
- **Color**: Common colors are all available. Here is a complete list of colors supported by SAS Graph: https://support.sas.com/content/dam/SAS/support/en/books/pro-template-made-easy-a-guide-for-sas-users/62007_Appendix.pdf.

- **Size**: specifies the size of the markers.

- **Symbols**: Here are the symbol markers supported by the PROC SGPLOT.



Results:



c. Display Height-Weight relationship by gender using GROUP option. Code:

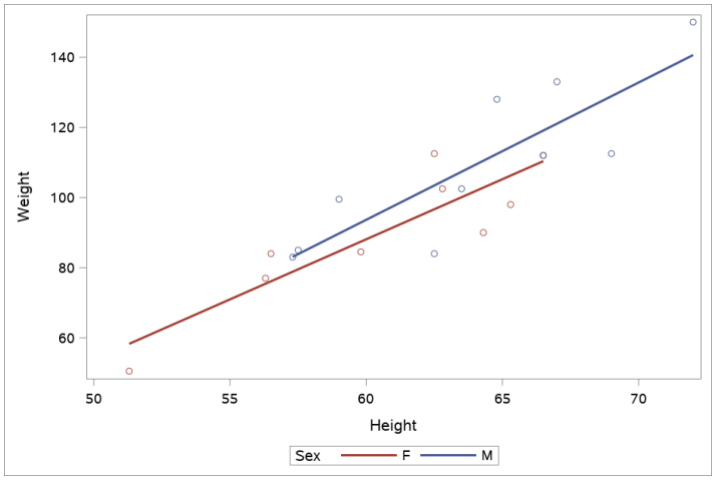
```
*Scatterplot -- Height by Weight separated by gender*;
proc sgplot data=sashelp.class;
  scatter x=height y=weight / group=sex;
run;
```

Check out the graph by yourself.

d. Using REG statement: specify dependent (y) and independent variable (x), and they will be displayed on the x-axis and y-axis correspondingly. Provide separate lines for female and male using GROUP option. Code:

```
*Scatterplot with regression line -- Height by Weight*;
proc sgplot data=sashelp.class;
   reg y=weight x=height / group=sex;
run;
```

Results:



3.5 Line plot – Changes over time

Let's look at another data set (sashelp.stocks) – Performance of Three Stocks from 1996 to 2005 The data set contains 699 observations.

Selected Variables	Type	Description
Stock	Character	IBM, Intel, and Microsoft
Date	Number	Date
Close	Number	Closing price on a specific date
High	Number	Highest price on a specific date
Low	Number	Lowest price on a specific date

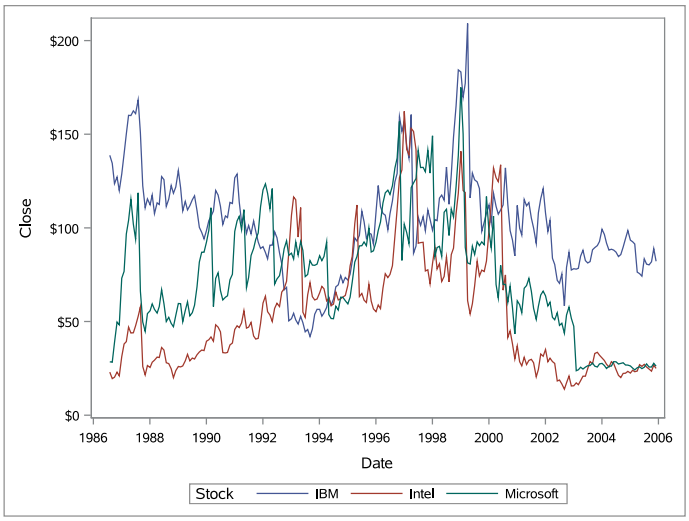
Let's see the closing price between 1996 and 2006 for the three stocks.

Example I -- Code:

```
*Line plot -- Three stocks over years*;
proc sgplot data=sashelp.stocks;
   series x=date y=close / group=stock;
run;
```

- SERIES statement: specify the x-axis and y-axis. Usually time variable is on the x-axis.
- GROUP option: define groups or categories for separate lines.

Results:

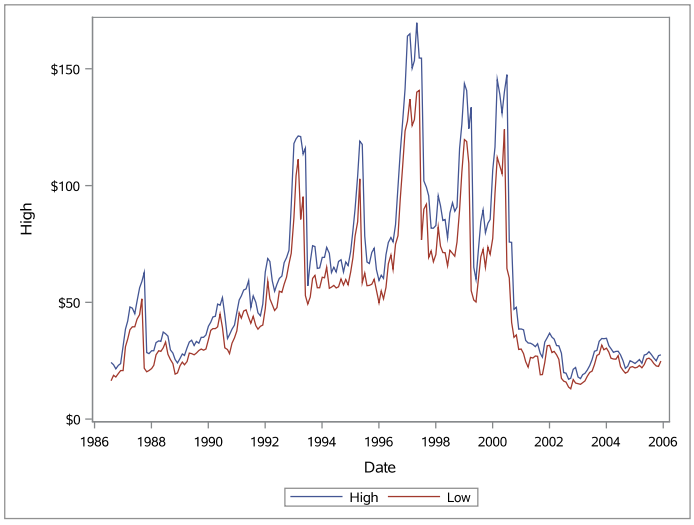


Example II -- Code:

```
*Line plot -- Multiple SERIES statements*;
proc sgplot data=sashelp.stocks;
   where stock in ('Intel');
   series x=date y=high;
   series x=date y=low;
run;
```

- Use multiple SERIES statements to draw separate lines (same x-axis).

Results:



There are many statements in SGPLOT procedure. Within each statement, there are also many options available which creates lots of flexibilities for creating graphical displays in SAS. Play with it! For more information, please check out SAS documentation: https://documentation.sas.com/?docsetId=grstatproc&docsetTarget=n0yjdd910dh59zn1toodgupaj4v9.htm&docsetVersion=9.4&locale=en