Introduction to Statistical Analysis with SAS

David Gerbing

School of Business Administration

Portland State University

April 30, 2017

Table of Contents

Introduction to Statistical Analysis with SAS

A SAS code file consists of lines of SAS code. Most of the lines of code are part of either a `Data Step`, which transforms data into a format for analysis by one or more SAS statistical procedures, a `Proc`. All SAS statements end in semicolons, including comments, which begin with a `*`.

Create SAS Data Set from External Text File

External text data file. An efficient method for getting data into SAS is to first process the data through Excel. Enter the data into Excel and then either leave the data in that format, or save as a `csv` (comma separated value) text file, which delimits (separates) adjacent data values with a comma. As an example, consider the data file:

http://web.pdx.edu/~gerbing/data/employee.csv

Following is a listing of two rows of the `csv` formatted data file, `employee.csv`, as viewed in a text editor such as NotePad++ for Windows or TextWrangler for Mac.

```
Name,Years,Gender,Dept,Salary,Satisfaction,HealthPlan,Pre,Post
"Tian, Fang",9,F,ACCT,61084.02,med,2,82,92
"Osterman, Pascal",5,M,ACCT,39704.79,high,3,62,74
```

List multiple data values for each observation (car) in a single row (the wide form of a data table). Commas separate data values. Next convert this **`CSV`** text file into a SAS Data Set.

SAS Data Set. SAS statistical procedures only analyze data in the form of a SAS Data Set. The Data Step is the part of a SAS program that creates a SAS Data Set from data that is read into the SAS system. The following SAS program reads the data from an external text file, `employee.csv`, and then transforms the data into a SAS Data Set called `employee`. This external data file for this example is located in the Stats directory on the H: drive on a Windows computer.

SAS Code file Part I: The DATA Step for creating the SAS Data Set

```
title  'employee.sas';  * A SAS program for creating a SAS Data Step';
options linesize=110; * Maximum columns of output in Listing file ;

data employee; * Name of the newly created SAS Data Set from this Data Step ;
  infile "H:\Stats\employee.csv" dsd dlm="," firstobs=2; * Skip first line ;
  length Name$ 35  Gender$ 1  Dept$ 4 Satisfaction$ 4;  * Max length ;
  input Name$ Years Gender$ Dept$ Salary Satisfaction$ HealthPlan Pre Post ;
run;    * $ in input statement means non-numeric variable ;
```

`In file` statement. Specify the name of the text data file with the **`infile`** statement. Here read a `csv` file. Consistent with the format in which Excel saves a `csv` file, the **`DSD`** option causes SAS to (a) interpret two delimiters together to indicate a missing data value, and (b) remove quotes around character values. The **`DLM`** option specifies the character that delimits adjacent data values, here **`DLM=","`**. The **`firstobs`** option, for first observation, specifies the first line in the data file from which SAS begins reading data. If the first line of the data file lists the variable names, specify **`firstobs =2`**.

`Input` statement. The **`input`** statement reads the data row by row until reaching the End-of-File. The current row of data read from the data file is converted to internal SAS format and added as the next row of the SAS Data Set. The **`input`** statement also names the variables for a SAS analysis. For example,

```
       input Name$ Years Gender$ Dept$ Salary Satisfaction$ HealthPlan Pre Post;
```

names 9 variables and directs SAS to read the corresponding 9 data values from each row of the data file. The $ sign at the end of a variable name in the **`input`** statement is not part of the variable name, but is instead a SAS formatting instruction to read a non-numeric data value.

`Length` statement. Variables Gender, Dept and Satisfaction, plus the ID identifier Name, have values with non-numeric values. By default, each character variable has a maximum length of 8 characters. The **`length`** statement overrides this default, as in,

```
       length Name$ 20  Gender$ 1  Dept$ 4 Satisfaction 3;
```

which specifies the maximum of characters for each data value for the non-numeric variables.

`Proc import` and **`export`**. SAS also provides an add-on, purchased separately, that includes `proc import`, which more easily reads data from an external file. The procedure automatically creates the data step, and then runs the data step, which creates the SAS data set. Similar to the read function in R, `proc import` does not require to first specify the variable names if they are contained in the first row of the data file, it automatically assigns each variable to its type such as numeric and character, and can automatically recognize missing data.

```
       proc import datafile='file_name' out=sas_data_set;
```

This one statement replaces the data step in the SAS code file.

To read an MS Excel file, specify the file format with the `dbms` parameter.

```
       proc import datafile='file_name' dbms=xlsx out=sas_data_set;
```

A variety of different formats for a data analysis are available.

Parallel to the import procedure, there is an export procedure, which can write data files into a variety of formats, including MS Excel.

```
       proc export data=sas_data_set  outfile='filename.xlsx' dbms=xlsx replace;
```

SAS Proc Steps for Statistical Analysis

The previous listing of the first part of a SAS program created a SAS Data Set called *`employee`*. The remaining part of the SAS program initiates a variety of statistical procedures that analyze the data in *`employee`*. Each **`proc`** step provides a distinct statistical analysis of the specified SAS Data Set. A SAS program may consist of all or some of these **`proc`** steps, or include some of many others available. If running interactively, such as on MS Windows, then add a **`run;`** statement following each procedure to conduct the corresponding analysis.

Note that getting multiple SAS graphs from one analysis requires additional programming. For convenience, do one SAS graph per analysis, that is, separate code files. The data set can be created once and then saved, then directly read by each subsequent analysis.

SAS Code file Part II: multiple proc's for analyzing the data in the SAS Data Set
`* List the Data Set to make sure read correctly ;`

```
proc print data=employee (obs=10);  * Option – List only 1st ten observations
(rows) ;

filename gfile 'Type.pdf';
goptions device=pdfc gsfmode=replace gsfname=gfile;
proc gchart data=employee;
  vbar Dept;         * vertical bar chart of one variable;

filename gfile 'Histograms.pdf';
goptions device=pdfc gsfmode=replace gsfname=gfile;
proc univariate data=employee;   * Basic SUMMARY statistics;
  histogram Salary; * Graphics Option – Hi-resolution histograms;

proc corr data=employee;   * CORRELATION matrix (plus basic summary stats);
  var Years Salary Pre Post;  * Limit analysis to specified variables;

proc reg data=employee;   * REGRESSION one predictor, no options;
  model Salary = Years; * Salary is response variable with predictor Years;

* Hi-resolution Graphics scatterplot with regression line and conf interval;
symbol interpol=rlclm95 value=dot cv=red ci=blue co=green;
proc gplot data=employee; * Data Values red dots, reg line blue, confidence green;
  plot Years * Salary; * Scatterplot of Years and Salary;

* MULTIPLE REGRESSION - Salary as a function of Years, Pre and Post.
 Options for each predictor: Confidence levels for b’s (clb), Tolerance (tol).
 Options for each observation: Predicted values and residuals (r),  Conf Levels
   for Conditional Means (clm) plus Prediction Intervals (cli), Influence Stats;
proc reg data=employee;
  model Salary = Years Pre Post / clb clm cli r influence tol;
  id Name;  * Identifying variable that labels the observations.;
```

Proc Reg

```
   proc reg data = sas_data_set;
      model response variable = predictor variables / options;
      id variable;                                            OPTIONAL
      by variables;
      weight variable;
      output out=sas_data_set keyword=names...;
      plot y-variable*x-variable = symbol / options;
```

Options on Model Statement

`clb`: confidence intervals for the parameter estimates
`cli`: % confidence limits for an individual predicted value
`clm`: % confidence limits for the mean of the dependent variable
`r`: analysis of residuals
`influence`: influence analysis of each observation on the estimates and the predicted values.
`tol`: tolerance values for collinearity analysis of parameter estimates
`dw`: Durbin-Watson statistic to test if errors from a time series have autocorrelation

Some SAS Data Step Options

Data Transformations. In this example read variables `var1` and `var2` from the data file *`mydata.dat`* stored at the root level of the H: drive. Define a new transformed variable X as the sum of `var1` and `var2` for each row of data. The resulting SAS data set, `mydata`, contains three variables: `var1`, `var2`, and `x`.

```
   data mydata;
      infile "H:\mydata.dat";
      input var1 var2;
      x = var1 + var2;
```

Save a SAS Data set. Use the **libname** statement and a two-part SAS Data Set name to save the SAS Data Set to a file on the specified directory. The **libname** statement specifies the directory (library) into which the SAS Data Set is saved. Additional SAS programs can directly access the saved SAS Data Set without having to first construct it with a Data Step.

```
      libname sasdata "H:\Stats\SAS\Projects";
      data sasdata.mydata;  * To save, use two-part reference in place of mydata.
         infile "H:\myData.dat";
         input var1 var2;
```

To reference the above SAS Data Set, specify the full two-part name in the **proc** statement. These **proc** statements can be part of the same code file as the Data Step, or can be in separate SAS code files that no longer require a Data Step because the SAS Data Set has already been created and saved on the disk directory. An example follows.

```
   proc print data=sasdata.mydata;
```

BY Statement

A **by** variable is a categorical variable, usually with a relatively small number of values. SAS computes the analysis specified by the **proc** separately for each value of the **by** Variable. When a **by** statement appears, the procedure expects the input data set to be sorted according to the values of the **by** variable, such as with **proc sort**. The following example yields an analysis of summary statistics of the variables Height and Weight for *each* value of Gender, Male and Female. The default is an ascending sort, so the data are first sorted with all Females listed before all the Males.

```
   proc sort data=example;
      by Gender;
   proc univariate data=example;
      var Years Salary;
      by Gender;
   run;
```

SAS Example

Input for a SAS analysis consists of the SAS code file, a text file with a file type such as `.sas`, and the data file, a text file with a file type such as .csv or `.dat`. Usually keep the first part of the filename the same (or nearly so) for all files in a project, such as `myeg.sas` and `myeg.dat`. These `.sas` and `.dat` files are prerequisite for a SAS analysis.

The output of a SAS analysis includes a text file with a file type of `.log`, which contains accounting information regarding the analysis, such as a list of any errors that may have occurred. Default statistical output goes to a text file with a file type of `.lst`, though this output may be also directed to files with a variety of formats such as `.html` and `.rtf`. High-resolution graphics are output to graphics files of specified types such as `.gif` and `.pdf`.

The SAS code file in the example below is `myeg.sas`, so the output files are `myeg.log` and `myeg.lst`. The four files – `.sas`, `.dat`, `.log`, and `.lst` – are all text files, which can be read and modified by virtually any computer application such as a word processor, viewed best with a monospaced font.

SAS Program Input

```
title  'myeg sas:  A first sas program.';
options linesize=79;

data myeg;
  infile 'myeg.data';
  input gender$ weight height;

proc print data=myeg;

proc freq data=myeg;
  tables gender;

proc means data=myeg;
  var weight height;
```

SAS Data Input (space delimited)

```
M 145 61
F 162 71
F 101 61
M 220 65
M 187 72
```

SAS LST (listing) File Output

```
          myeg sas:  A first sas program.                              1

                       Obs    gender    weight    height

                        1       M         145       61
                        2       F         162       71
                        3       F         101       61
                        4       M         220       65
                        5       M         187       72

          myeg sas:  A first sas program.                              2

                              The FREQ Procedure

                                               Cumulative    Cumulative
          gender    Frequency     Percent     Frequency      Percent
          -----------------------------------------------------------
          F                2       40.00             2        40.00
          M                3       60.00             5       100.00

          myeg sas:  A first sas program.                              3

                              The MEANS Procedure

 Variable    N            Mean         Std Dev         Minimum         Maximum
 ----------------------------------------------------------------------------
 weight      5     163.0000000      44.7045859     101.0000000     220.0000000
 height      5      66.0000000       5.2915026      61.0000000      72.0000000
 ----------------------------------------------------------------------------
```

SAS Log File Output

```
gerbing:~/sas/myeg:92 > cat myeg.log
1                                                          The SAS System
15:58 Tuesday, May 24, 2016

NOTE: Copyright (c) 2002-2012 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software 9.4 (TS1M3)
      Licensed to PORTLAND STATE UNIVERSITY - SFA - T&R, Site 70153385.
NOTE: This session is executing on the Linux 3.10.0-327.10.1.el7.x86_64 (LIN X64)
platform.

NOTE: Additional host information:

Linux LIN X64 3.10.0-327.10.1.el7.x86_64 #1 SMP Tue Feb 16 17:03:50 UTC 2016
x86_64
CentOS Linux release 7.2.1511 (Core)

You are running SAS 9. Some SAS 8 files will be automatically converted
by the V9 engine; others are incompatible.  Please see
http://support.sas.com/rnd/migration/planning/platform/64bit.html

NOTE: SAS initialization used:
      real time           0.72 seconds
      cpu time            0.06 seconds

1          title  'myeg.sas:  a first SAS program.';
2          options linesize=79;
4          data myeg;
5            infile 'myeg.data';
6            input gender$ weight height;

NOTE: The infile 'myeg.data' is:
      Filename=/home/gerbing/sas/myeg/myeg.data,
      Owner Name=gerbing,Group Name=gerbing,
      Access Permission=-rw-r--r--,
      Last Modified=24May2016:08:56:18,
      File Size (bytes)=405

NOTE: 5 records were read from the infile 'myeg.data'.
      The minimum record length was 80.
      The maximum record length was 80.
NOTE: The data set WORK.MYEG has 5 observations and 3 variables.

2                               The SAS System      15:58 Tuesday, May 24, 2016

NOTE: DATA statement used (Total process time):
      real time           0.24 seconds
      cpu time            0.01 seconds


8          proc print data=myeg;

NOTE: There were 5 observations read from the data set WORK.MYEG.
NOTE: The PROCEDURE PRINT printed page 1.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.26 seconds
      cpu time            0.04 seconds


10         proc freq data=myeg;
11           tables gender weight height;
```

```
NOTE: There were 5 observations read from the data set WORK.MYEG.
NOTE: The PROCEDURE FREQ printed page 2.
NOTE: PROCEDURE FREQ used (Total process time):
      real time           0.09 seconds
      cpu time            0.01 seconds


13         proc means data=myeg;
14           var weight height;

NOTE: There were 5 observations read from the data set WORK.MYEG.
NOTE: The PROCEDURE MEANS printed page 3.
NOTE: PROCEDURE MEANS used (Total process time):
      real time           0.05 seconds
      cpu time            0.02 seconds


16         proc chart data=myeg;
17           vbar height;
NOTE: The PROCEDURE CHART printed page 4.
NOTE: PROCEDURE CHART used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds


NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           1.84 seconds
      cpu time            0.19 seconds
```

Directing SAS Graphics to a Graphics file

SAS can generate graphics output, such as from **`proc gplot`** or the **`histogram`** option in **`proc univariate`**. SAS on Windows automatically directs this graphics output to a separate window. Unix requires additional SAS statements to direct output to an external file. These statements also work for Windows if additional specification and formatting are desired.

For example, a device set to `pdfc` directs graphics output to a color pdf file. The following two statements direct SAS graphics to the specified file.

```
    filename gfile 'yourname.pdf';
    goptions device=pdfc gsfmode=replace gsfname=gfile;
```

Use `device=cgm` for vector graphics, which allows editing the graphic with a drawing application on your ow computer such as Adobe Illustrator. (To edit SAS graphics in Illustrator, first select the graphic then under the `Object` menu, choose `Clipping Mask` and then `Release`.) If the `filename` and `goptions` SAS lines are not included, graphic output will be directed to the default file `sasgraph.png`. If running SAS interactively, SAS prompts for a graphics device. Enter `pdfc` or `cgm` and the appropriate output is directed to the default file `sasgraph.pdf` or `sasgraph.cgm`.

Directing SAS Output to an HTML, RTF or PDF file

SAS output can be directed to files with a variety of formats other than the standard text file format of the `.lst` file. The mechanism that controls the SAS output system is `ODS` for Output Delivery System. SAS output can also be directed to files formatted as `.html`, `.pdf`, `.xlsx`, `.ppt`,

`.rtf` for a text file that a word processor such as MS Word can open, and `.epub3` for an ebook that opens up, for example in iBook on a Mac. To send SAS text output to a file in one or more of these formats, use one or any of the following `ODS` statements before the location in the SAS code file that generates the output.

```
    ods html5 file='file_name.html';
    ods pdf file='file_name.pdf;
    ods rtf file='file_name.rtf;
    ods excel file='file_name.xlsx';
    ods powerpoint file='file_name.pptx';
    ods epub3 file=''file_name.epub';
```

Access the following for extensive documentation of the many options, such as, for example, creating a gradient background for the PowerPoint slides.

http://support.sas.com/cdlsearch?qt=powerpoint&Find=Search&qp=url%3A%2Fdocumentation%2Fcdl%2Fen%2Fodsug%2F69832%2FHTML%2Fdefault

When creating an `.html` file, one possibility is to use an FTP client app to move the file to the local PC, and then view the output through a web browser. If you wish to post output directly to the web, as a PSU student, you can also log into the Odin computing system with an FTP client, such as the free FileZilla, or similar application. If Filezilla is not installed on the system you are using, and you do not have install permission, then either use another FTP client or use your own computer where you can download FileZilla.

Login to host `sftp://sftp.myfiles.pdx.edu`, and go to your Unix-Web folder, and then your `public_html` folder, which provides access to a web server. FTP any `html` output into that folder, and the output is available on the world wide web. The URL is

```
        web.pdx.edu/~username/name.html
```

After specifying one or more different output files in the specified formats, SAS also continues to create the standard `.lst` file, unless directed otherwise. Use

```
        ods listing close;
```

to stop all subsequent output to the SAS listing file `.lst` at the point in the SAS program file at which the statement occurs.

The output of ODS is organized by table, and many tables may be displayed. An advantage of ODS relies on the fact that SAS labels each table in an output with a specific name. ODS leverages this naming by limiting output to only tables selected by name. To find the names of the tables for a specific statistical procedure, web search

```
        "proc name" tables ods
```

where `name` is the name of the SAS procedure such as "`proc reg`". Or, add the following statement to a SAS code file:

```
        ods trace output;
```

which then instructs SAS to print a list of the tables from the output in the `.log` file.

Specify the selected tables by placing the following statement before the corresponding statistical procedure, listing all of the tables that should appear on the output:

```
      ods select table1 table2 table3;
```

Working with Unix

SAS on Unix

File Listing *Before* Running SAS

```
gerbing:~/sas/myeg:83 > ls -l
total 8
-rw-r--r--. 1 gerbing gerbing  405 May 24 08:56 myeg.data
-rw-r--r--. 1 gerbing gerbing 1375 May 24 15:53 myeg.sas
```

Running a SAS Code file

```
gerbing:~/sas/myeg:84 > sas myeg
```

(Default) File Listing *After* Running SAS

```
gerbing:~/sas/myeg:85 > ls -l
total 20
-rw-r--r--. 1 gerbing gerbing  405 May 24 08:56 myeg.data
-rw-rw-r--. 1 gerbing gerbing 3680 May 24 15:58 myeg.log
-rw-rw-r--. 1 gerbing gerbing 5814 May 24 15:58 myeg.lst
-rw-r--r--. 1 gerbing gerbing 1375 May 24 15:53 myeg.sas
```

Connecting to Unix

Secure interaction with a remote Unix system from a PC requires an application called an SSH client, where SSH stands for "secure shell". Access PSU research computing system with the `Host Name` of `circe.rc.pdx.edu`. *On-campus* access simply requires a logon with your user-id and password.

For *off-campus access* to the PSU computing network, two-factor authentication over a vpn (virtual private network) is required. Install the Cisco AnyConnect VPN Client, and then, on your smart phone, the Duo Mobile app. Connect remotely via the VPN client, which then pings your Duo Mobile app, from which you then approve the connection. Directions:

http://www.pdx.edu/oit/virtual-private-network-vpn
http://www.pdx.edu/oit/connecting-to-vpn-with-two-step-verification

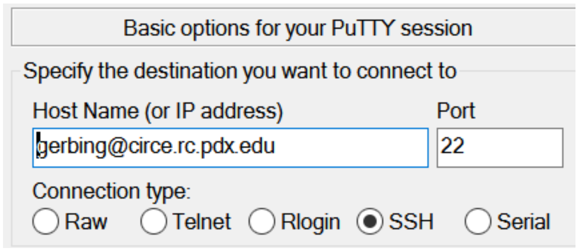
At its core, Macintosh is a Unix system, and already includes a command line app for remote connection called `Terminal`. Open the `Terminal` application and logon with

ssh username@circe.rc.pdx.edu

These settings can be saved under the `Shell` menu by choosing `New Remote Connection…` .

Many free SSH clients also exist for Windows, such as PuTTy. Web search "`ssh putty`" to obtain the site for downloading. PuTTy has the unusual property of having no menu line at the top of its window. Instead, to access different options, right-click at the top of its window. Also, instead of the usual `Edit` → `Copy` menu sequence, PuTTy automatically copies to the clipboard anything selected, ready for pasting into another application such as NotePad++ or MS Word.

To login with PuTTy just enter your user information and check the SSH box.


Basic options for your PuTTY session
Specify the destination you want to connect to

Host Name (or IP address): gerbing@circe.rc.pdx.edu
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

The default PuTTy display is white letters on a black screen. To change to a different visual configuration, right-click at the top of the PuTTY window and choose `Change Settings`. Then under `Window` choose `Colours`. The following screen color scheme may be more pleasing.

default background: 239, 239, 216
default foreground: 14, 15, 11
default bold foreground: 4, 4, 120
cursor color: 116, 167, 218

Once logged in, enter Unix commands such as to create and change directories, list and/or edit the contents of files, and to run SAS jobs.

FTP

Both input and output files easily can be moved back and forth between a PC or Mac and the Unix system with an FTP client. Also, easy to edit your SAS files on your local computer, usually with some text editor, and them upload. Once your graphics file is created, an FTP client application moves your graphics output file to your local computer to view and print. Go to filezilla-project.org to locate a free FTP client.

To login to an FTP client, for the Host enter: `sftp://circe.rc.pdx.edu`

Unix Commands

`ls`	list all files and directories (folders) within current directory
`ls -l file_name`	list files in current directory with specified name in a vertical list eg., `ls -l myeg.*` where `*` is a wildcard, so list all files that start with myeg
`cd directory_name`	move down one level to specified directory
`cd ..`	move up one directory level
`cd ~`	move to the home directory
`mkdir directory_name`	make directory
`rmdir directory_name`	remove directory
`cat file_name`	display file on screen, e.g., cat myeg.sas
`rm file_name`	remove file
`cp from to`	copy file
`mv from to`	physical move file, or rename file
`logout`	logout
`repeat last command`	up arrow

vim Editor [optional]

For purposes of running SAS jobs on Unix, all editing can be done on the local computer and files uploaded and downloaded easily with FTP, so knowledge of a Unix editor is not needed. To be serious, however, about editing text on a Unix system, as well as Windows or Mac, the recommended editor is `vim`, which is much faster to use for entering and editing text than, for example, MS Word. The `vim` editor is a free download for both Windows, Mac and Unix/Linux computers.

If working much on a Unix system, editing directly within Unix is preferred. To access `vim`, in response to the Unix prompt, enter

`vim` *filename*

If the file already exists, the file is opened `vim`. Otherwise creates a new file.

The key feature of `vim` is that it is modal. Entering vim is to be in `Normal` mode, ready for commands, but not, as with virtually other editors and word processors, ready to enter text. To enter text, first enter `Insert` mode, such as with the command `i`. Enter `ESC` to go back to `Normal` mode. The `ESC` is used much in `vim`, so much that most users re-map their `caps lock` key to `ESC` because `caps lock` is more accessible.

After that most important editing command, below is a subset of some `vim` commands. There are many more useful commands. To avoid the highlighting, enter: `:syntax off`.

Command	**Insert or Delete Text**
i	Insert before character
a	Insert after character
ESC	Leave insert mode
x	Delete character
dd	Delete current line
2dd	Delete 2 lines, etc.
Ctrl-y	Backward one screen
Ctrl-space	Next word

Command	**Cursor Movement**
Arrow Key	Specified direction
h j k l	Left, down, up, right
w	Forward a word
2w	Forward 2 words, etc
b	Back a word
0	Beginning of line
$	End of line
gg	Beginning of file
G	End of file

Command	**Saving File**
w	Write the file (save)
wq	Write and quit

q!	Quit and lose edits

Command	Misc
u	Undo

SAS on Microsoft Windows

On Windows SAS runs in a mouse-oriented GUI environment.

SAS Windows

Editor Window – your SAS instructions (SAS statements), saves as `.sas` file
Log Window – an accounting of your program, such as errors and processing times
Output Window – results displayed as text, saves a `.lst` file
Graphics Window – results displayed as hi-resolution color graphics

Activate a window by clicking on it, or choose the appropriate tab towards the bottom of the screen.

Running a SAS Program

To run the SAS statements in the Editor Window: make sure your data file, with a file type such as `.csv` or `.dat`, has been created. The **`infile`** statement in the Data Step must contain the *full* path name for the data file, such as,

```
      infile "H:\Stats\employee.csv" dsd dlm="," ;
```

From the menu choose `Run` > `Submit`, push the `Run` button on the toolbar, or push `F8`.

Submit just part of the program in the Editor Window by selecting that part with the mouse and Shift key before running. Otherwise the entire code file is submitted.

After running your SAS program, first check the Log window for errors. Then view the results of the analyses in the Output window and, if relevant, the Graphics window.

If previous analyses had been run, a useful command before a new run is `Edit` > `Clear Text` or `CTRL-E`, which clears the contents of the active window. Without this command, the output of the current analysis appends to the end of the previous analysis.

Saving & Importing SAS Files

The active window is saved with the usual `File` > `Save` command. The SAS code file, data file, log file and list file are all text files, so a `.sas` code file can be constructed outside of the SAS system with a text editor and pasted into the SAS Editor Window or read with `File` > `Open`. Similarly, the `.log` and `.lst` files can be read by any text editor or word processor, displayed with a monospaced font, reformatted, stored and printed as desired.

Save the Graphics Window of hi-resolution color graphics as an image with `File` > `Export as Image…` Then choose an image format such as GIF.

More SAS Examples

Documentation:
http://support.sas.com/documentation/cdl/en/procstat/63104/HTML/default/viewer.htm#univariate_toc.htm

Proc Univariate

```
   proc univariate data=sas_data_set;

      var variables;
      histogram variables / options;
      title 'your title' ;
      class variable;                                              OPTIONAL
      output out=SASdataset keyword=names...;                     |
```

Options on Histogram Statement

`endpoints`: display endpoint (cutpoint) labels, `midpoints` is default
`endpoints = low to high by width` create histogram with endpoints of class intervals (bins)
`kernel`: construct best-fitting smooth curve in general
`normal`: construct best-fitting normal curve
`cframe`: background color
`cfill`: color of histogram bars
`vscale=count`: vertical scale is count, default is percentage
`vaxis = low to high by width`: vertical axis scale

Examples

```
filename gfile 'testscores.pdf';
goptions device='pdfc' vorigin=3in vsize=6in hsize=7in gsfmode=replace
   gsfname='gfile';
proc univariate cibasic data=testscores;
  var raw;
* class section;
  histogram raw / nrows=3
                  midpoints = 10 to 40 by 1
                  vaxis = 0 to 6 by 1
                  vscale=count
                  height=3
                  frontref href = 25.2,28.8,32.4,36 chref=cx656060
                  hreflabels='70%' '80%' '90%' '100%'
                  kernel (color=black)
                  cframe=cxE6E6E6 cframeside=cxE6E6E6 cfill=steelblue;
  inset n = "n" (5.0) mean = "Mean" (8.2) median = "Median" (8.2) std = "Std Dev"
   (8.2) / pos=nw;
  label raw="Midterm Scores";
```

```
proc univariate data=disc normaltest;
  var salary;
  probplot salary / normal (mu=est sigma=est);
```

Proc Autoreg

Example

```
title  'ar.sas:  autoregression sas program.';
options linesize=90;

/* create the data set called ar */
data ar;
  infile 'ar.dat' dsd dlm=",";
  input t y;

/* print the data set to make sure read correctly */
proc print data=ar;

/* calculate basic summary statistics */
proc means data=ar;
  var t y;

/* "simple" regression */
proc reg data=ar;
  model y = t / clb dw;

/* autoregression */
proc autoreg data=ar;
  model y = t / nlag=2 dw=2 dwprob;
  output out=p p=yhat pm=trendhat;

proc print data=p;

filename gfile 'ar.pdf';
goptions device='pdfc' gsfmode=replace gsfname='gfile' rotate=landscape;
proc gplot data=p;
  symbol1 color=cx660000 value=dot interpol=none;
  symbol2 color=cx7997A7 value=star interpol=join width=8;
  plot y * t = 1
       yhat * t = 2 / overlay href=51.5 cframe=cxE6E6E6;
  label t='Time';
  title 'Forecasting Y';
```