EXST 700x

Lab #2: Basic Data Organization and PROC Statements

Tips

1. Previous lab assignments are posted online if you need to refer to them.

2. It is a good idea to save your SAS code often while you are working to avoid accidental loss. You can save it to your desktop or the temp directory, but the lab machines have too many users for us to guarantee the integrity of anything saved on the local hard drive, and you may not be able to get the same machine next week. It would be safest to bring a flash drive with you to save your work or save your work to TigerBytesII

3. You only need to save your SAS program and data where applicable. It is not necessary to save output or log since you can get them every time you run SAS program.

4. Read the details discussed in the lab assignments. It will help you to understand SAS.

5. You can use computers in either lab room 11 or lab room 44 if the lab is not occupied by a class. Room 44 has fewer scheduled classes and is usually open. If you want to install SAS onto your own computer, see the staff in room 161.

SAS input and output

For this and future assignments it is recommended that you create a directory specifically for the assignment. I will create a directory called “EXST700X Assignment 02” on a flash drive. If you do not have a flash drive you can create a subdirectory in your “My Documents” directory which is probably called “C:\Users*YourLogonID*\Documents”. If you save a file without specifying a directory it will go into default initial folder used by SAS, a directory called “C:\Users*YourLogonID*\Documents\My SAS Files\9.3”.

Most of the data sets that we will use for the EXST 7005 lab are small enough so that they can be placed directly into the SAS program in the program editor, as we did last week. However, many real world practical uses of SAS require using datasets much too large to include in the program. In this assignment access some external data sets.

Today’s example needs an external file called “grades.txt” that contains exactly the same data as was originally in the program. To use the external data set we simply remove the data from the program and use an “INFILE” statement to indicate to the program where it may find the data. Give the name of the data file with the full path in quotes;
INFILE ‘F:\EXST700X Assignment 02\GRADE.txt’;.

The *CURRENT FOLDER*: SAS keeps track of what is called the current folder. When you initially open the current folder is usually “C:\Users\YourUserID\Documents\My SAS Files\9.3”. If you double click on a SAS file to open it, the directory where that file is stored becomes the current folder. If files to be input and files to output are to be placed in the same current folder, it is not necessary to provide the full path. The SAS code given as an example in this assignment is available online with the data included as “example 02 example.TXT”. The current folder is shown on the right side of the lower border of the SAS window. The current folder can be changed by clicking on the name of the current folder and using

windows to redirect the folder. If you have created a folder for today's exercise then direct SAS to this directory and make sure the data is stored in that folder.

ODS output files: The Output Delivery System (ODS) can create some output files of various types. We have already seen the ODS control output in our usual housekeeping statements.

```
dm 'log;clear;output;clear';
options nodate nocenter pageno=1 ls=78 ps=55;
OPTIONS FORMCHAR="|----|+|---+=|-/\<>*";
ODS listing; ods graphics off; *ods noresults;

title1 "EXST 700X Example 02";
title2 "James P Geaghan, SECTION 0";

***EXST 700X**********;
*** Example 02      ***;
*** James P Geaghan ***;
*** SECTION 0       ***;
**********************;
```

Notice that I have embellished the statements above with a box made of comment statements, and that I have deactivated the **`ods noresults;`** statement by turning it inte a comment. Otherwise it would suppress the "RESULTS VIEWER" output of the windows program.

Other statements that may be included in this section that control program behavior and input or output. Three that you may find useful are the ODS statements;

```
ODS HTML style=minimal body='EXAMPLE02.html';
ODS RTF style=minimal body='EXAMPLE02.rtf';
ODS PDF style=minimal body='EXAMPLE02.PDF';
```

These statements will create an HTML output file, RTF output file and a PDF output file, respectively. The **`style=minimal`** simplifies the HTML output by suppressing font changes and background shading. These files will be written to the "current folder" unless full directory path information is provided within the quotes of the "**body**" option.

The HTML output is the same as the output SAS writes to the "RESULTS VIEWER". However, the output included in the file starts at whatever point in the program the statements are placed. That is, only output created after the insertion point of the ODS statement is included in the output file. The output for each ODS ceases with the statements;

```
ods html close;
ods rtf close;
ods pdf close;
```

Permanent SAS datasets: When a data set is created in SAS with a simple data step such as "`DATA grades;`" it goes into a temporary SAS library called the "WORK" library. When the program ends the WORK library is cleared and lost to future use. These dataset can be referred to by single data set name statements like "`CLASS`" or as two-level name statements where the two levels are separate by a period, such as "`WORK.SAS`".

It is also possible to create a permanent SAS LIBRARY that will persist when SAS is exited, and can be used for future analysis. To do this, a SASLIB statement must be placed early in the program before the DATA step. For example, the following statement,

```
LibName SASLib 'F:\EXST700X Assignment 02';
```

would create a SAS library called **`SASLib`** in the directory "**`F:\EXST700X Assignment 02`**". To create a permanent dataset called **`GRADE`** in a DATA step, a "two level" dataset name would be used.

```
data SASLib.GRADE;
```

After running this statement you will find a SAS dataset called "**grade.sas7bdat**" in the directory specified in the SAS **libname** statement.

There is also a default directory for permanent storage called SASUSER, so SASUSER could be substituted for TEMPDIR without a prior SASLIB statement. An alternative would be to use the statement;

```
data sasuser.GRADE;
```

PROC SORT: The SORT procedure is used to alter a dataset such that all the observations are ordered according to values of one or more variables.

It is possible to leave the original data set unaltered and to sort the data into a newly created data set. To do this the SORT statement will be followed by an OUT option that gives the name of the new SAS dataset. The OUT option must part of the PROC SORT statement, before the semicolon that ends the SORT statement (though it does not have to be on the same line).

If you don't use an OUT option the original dataset will be replaced with the sorted version. You can make the new dataset either a temporary one or a permanent one by choosing the library's name before the dataset. By default, "WORK" is a temporary SAS library, and "SASUSERS" is a permanent SAS library.

A BY statement always follows a SORT statement. It specifies the variable by which the dataset is to be sorted. The default sorting order is ascending. In this case, we are going to sort data according to sections.

```
proc sort data=SASLib.GRADE; by SECTION;
run;
proc print data=SASLib.GRADE; by SECTION;
   Title3 'Data listing from PROC PRINT';
   Title4 'With a by statement';
run;
```

Many applications require sorting by more than one variable. This is easily done by listing the variables you want to sort by in the "BY" statement from most important to least important. By default SAS sorts from lowest to highest. If you want to sort some variable from high to low, you can put the word "DESCENDING" in the BY statement before the variable's name. For example, if you use the following SAS code, the dataset will be sorted first in order of the SECTION variable in an ascending order and then, within each SECTION, by the EXAM variable in descending order. The following statement will save the sorted dataset in the permanent **SASLib** library in a file called "SORTEDGRADE".

```
proc sort data=SASLib.GRADE out=SASLib.SORTEDGRADE;
   by SECTION descending EXAM;
run;
```

An important point to notice: if you are sorting by a character variable, capital letters will be sorted before lower case letters. Therefore, observations starting with capital letters will show up ahead of those starting with lower case letters. So please remember to either consistently capitalize all of the first letters of every observation or to capitalize none.

PROC PRINT: The PRINT procedure is used to print data in tabular form. It is frequently used with other options. The SORT procedure discussed above does not produce any output, so sorting is usually followed by a print procedure of some other application. Other statements can be used together with the PROC PRINT statement such as:

1) VAR statement: – controls the variables to be printed and the order in which they are printed. For example, if you want to list only NAME and EXAM and not the other variables, you could use the following code:

```
proc print data=SORTEDGRADE; by SECTION;
   Title3 'Data listing from PROC PRINT - only 2 variables';
   Title4 'Sorted by two variables (second one descending)';
   var EXAM name;
run;
```

2) BY statement: – groups the observations by values of one or more certain variables. However, the dataset must be sorted before using this statement. So applications usually follow a PROC SORT procedure.

3) And of course the ever popular TITLE statements previously discussed.

PROC UNIVARIATE

PROC UNIVARIATE is a very commonly used procedure for data exploration. It produces statistics describing the distribution of a single variable. It should always be followed by a VAR statement to specify which variable is to be analyzed. Otherwise, SAS will compute and exhibit statistics of all the numeric variables in the dataset.

```
proc univariate data=SASLib.GRADE plot;
   var Exam;
   Title3 'PROC UNIVARIATE output';
run;
```

PROC MEANS

If only a few major statistics are needed for a variable, such as the mean and the standard deviation, and not all the summary statistics, then PROC MEANS may be a better choice than PROC UNIVARIATE. By default the mean, the total number of observation (without missing values), the standard deviation, the minimum and the maximum are reported. However, specifying specific statistics in the procedure can produce a different set of statistics. Some of the key words to call optional statistics are: CSS, RANGE, CV, STDDEV, STDERR, MAX, SUM, MEAN, MIN, MODE, USS, N, VAR, MEDIAN, Q3, Q1, QRANGE.

A few important statements are commonly used with PROC MEANS:

1) VAR: it can specify the variables you want SAS to analyze if you don't want statistics for all numeric variables.

2) CLASS: not to be confused with the variable "CLASS" in this example, the CLASS statement will make SAS report statistics for different groups (i.e. classes or categories). It does not require the data to be sorted beforehand. For example:

```
proc means data=SASLib.GRADE;
   class SECTION;
   var EXAM LABAVG;
   Title3 'PROC MEANS output';
run;
```

PROC PLOT

The scatter plot of one variable plotted against another is a useful tool for the initial evaluation of data. It can be used to evaluate the range of values for both variables, to determine of there are any very unusual, and possibly erroneous values, and to do an initial evaluation of possible trends of one variable to increase or decrease as the other increases.

```
proc plot data=SASLib.GRADE;
   plot EXAM * LABAVG = name;
   Title3 'PROC PLOT presentation';
run;
```

Assignment 2

The dataset: Information on available used cars.

COUNTRY	MILEAGE	PRICE	MPG	YEAR
JAPAN	80	5800	34	2004
USA	110	2800	23	2001
USA	75	5300	20	2004
JAPAN	92	6800	30	2005
GERMANY	87	6500	22	2001
USA	90	3000	25	2002
JAPAN	67	9000	33	2006
JAPAN	76	8500	32	2006
GERMANY	120	4200	20	1997
USA	89	3400	19	2000
USA	80	5500	24	2004
JAPAN	120	1500	28	1994
JAPAN	98	4000	30	2000
USA	85	3300	23	2001
GERMANY	115	4500	22	1999
USA	78	5200	21	2003
USA	90	3600	20	2002
JAPAN	115	4000	29	1998
JAPAN	85	7000	33	2005

Suppose you are going to select a used car from the above list. Create the following output and answer any questions posed using SAS. For each question, please show relevant SAS output and your log. Try to organize your responses for clarity.

1) You will want to include in your program the usual housekeeping statements with options, comments and titles similar to those in ASSIGNMENT 01. (2 points)

Include appropriate title statements. (1 point)

Create a data step to enter the data set above. The data step will include an INPUT statement and a DATA statement. To these statements add the following LABEL statement: (1 point)

```
label mileage = '(thousands of miles)';
```

2. Print data for all the cars, but only for the variables **`price`** and **`mileage`**. (1 point)
3. Print all the variables in three groups, grouped by countries. ***Hint: sort first***. (1 point)
4. Sort the original dataset in an ascending order by both to **`price`** and **`mileage`** and print this new dataset. (1 point)
5. Sort the original dataset in a descending order according to year, and save it in the permanent library “sasuser”. Then print this new dataset. (1 point)
6. Use PROC UNIVARIATE to calculate mean and standard deviation for **`price`**. (1 point)
7. Using PROC MEANS to find the minimum and maximum values for **`MPG`**. (1 point)