SAS Reference

This document describes some basic tips for using SAS.

There are two main **steps** in a SAS "program": data steps and procedure steps. Each step is composed of SAS **statements** that are lines of code. **It is important to remember that statements always end with a semicolon in SAS.** Since a data step must occur before any procedure steps, first we will look at data entry.

For future reference, code that refers to fixed SAS statements will be in all capitals and variable names will be lowercase. The blue sections in Courier font contain example code that can be copied into the SAS editor and run. The output will be a printout of the data set.

Data Steps

Data steps are used to enter and name data. There are multiple ways to enter data but we will only tackle two common ways: **instream** data entry and using **external** files.

```
DATA kids;
INPUT name $ males females;
DATALINES;
Smith 1 1
Jones 2 1
Johnson 0 3
Cooper 2 2
;
RUN;
PROC PRINT;
RUN;
```

The DATA line is used to name the data set for future reference, so here the set created would be referred to as "kids" in future steps.

The INPUT line is used to tell SAS what variables will be in the data set, the format of the variables (e.g. numeric or alphanumeric), and the order the variables will be entered. A dollar sign after a variable indicates that the variable is alphanumeric. The lack of a symbol following a variable indicates that the variable is numeric.

The RUN statement is needed so SAS knows to execute the section of code above the run. Without the RUN statement, SAS will not do anything when the code is submitted.

The DATALINES statement notifies SAS that the next section will contain the raw data. SAS expects (without further commands) that the data will be entered with spaces in between the variables and each observation is on its own line. If we don't have spaces in between the data, we can modify the program to show a different format, e.g. in which fixed columns for variables are noted.

The PROC PRINT statement in the above example is part of a procedure step used to print out the data and is separate from the data step. It is useful for checking input and calculation statements executed in the data step.

Now here is an example of an input statement where the columns for the variable are fixed in the data set:

```
DATA kids;
INPUT name $ 1-7 males 8 females 9;
DATALINES;
Smith 11
Jones 21
Johnson03
Cooper 22
;
RUN;
PROC PRINT;
RUN;
```

The numbers after the variables are the columns SAS will expect to find that variable. For example SAS will expect the number of male children to be in column 8.

Data steps can also be used to generate new variables from the raw data. For example we can again modify the program to calculate the total number of kids in the household.

```
DATA kids;
INPUT name $ males females;
kidtot = males + females;
DATALINES;
Smith 1 1
Jones 2 1
Johnson 0 3
Cooper 2 2
;
RUN;
PROC PRINT;
RUN;
```

For each observation SAS will create the variable kidtot. For the line (SAS calls each data line an observation) that has 'Cooper' for the name variable, kidtot will be 4.

SAS allows for a variety of other logical statements to be used within the data step, as a quick example, we can use IF-THEN-ELSE statements to make an indicator variable for determining if a family has male children or not:

```
DATA kids;
INPUT name $ males females;
IF males = 0 THEN hasmale = 0;
ELSE hasmale = 1;
DATALINES;
Smith 1 1
Jones 2 1
Johnson 0 3
Cooper 2 2
;
RUN;
PROC PRINT;
RUN;
```

Three other useful statements in a DATA step are SET, WHERE, and DROP statements.

SET statements are useful when you have a data set already created but you want to work with only part of the data or you want to add more observations. The statement itself places an existing data set in a new one. For example if we wanted to add an observation to the "kids" data set and we already have "kids" created we can do the following:

```
DATA newline;
INPUT name $ males females;
DATALINES;
Legg 0 0
;
DATA newkids;
SET kids;
SET newline;
RUN;
PROC PRINT;
RUN;
```

WHERE statements are used to limit what observations are included in the new dataset. For example if we only wanted the "newkids" data set to have families with no male children we could do this:

```
DATA newkids;
SET kids;
WHERE males=0;
RUN;
PROC PRINT;
RUN;
```

WHERE statements can also be used in PROC steps as we shall see later in PROC SURVEYMEANS.

DROP statements are used to remove variables from the data set. Suppose we want to strip the family name from the data set, we could do:

```
DATA kids;
INPUT name $ males females;
DROP name;
DATALINES;
Smith 1 1
Jones 2 1
Johnson 0 3
Cooper 2 2
;
RUN;
PROC PRINT;
RUN;
```

External Data Entry

Suppose we have our "kids" data in an external text file and want to read it. We could write a program similar to the one above but now it has FILENAME and INFILE statements:

```
FILENAME kiddata 'd:/mydata/kids.txt';
DATA kids;
INFILE kiddata;
INPUT name $ males females;
RUN;
PROC PRINT;
RUN;
```

The FILENAME statement assigns a location on the computer to the name "kiddata" so SAS knows where to look for it.

The INFILE statement informs SAS to use the file named "kiddata" from the FILENAME statement as the source of the data. Except for the DATALINES statement and entering of data, the other statements mentioned in the instream data entry section still apply to external data entry steps.

PROC Steps

Procedure (PROC) steps are functions applied to the data. PROC steps can be used to create tables, calculate summary statistics, and make plots among other things. Since the syntax of PROC steps varies by the function requested, we will look at common syntax. First there will be some example code followed by an explanation.

PROC PRINT

```
PROC PRINT data=kids;
VAR name females;
TITLE 'Household Daughters';
RUN;
```

PROC PRINT is used to display the data to the Output Window. The "data=" is used to instruct SAS as to which data set to use and can be used in any PROC step. If

you omit the "data=" part, SAS will assume the last data set created before the PROC PRINT should be used.

The VAR line tells SAS to only display the household name and the number of female children in the household. If the VAR line is omitted, SAS will display all of the variables.

The TITLE line makes SAS display whatever words are in the single quotes above the output. (You cannot use single quotes such as apostrophes within the title text.) The TITLE statement is very useful in labeling your output.

PROC SORT

```
PROC SORT;
BY males;
RUN;
```

PROC SORT is used to reorder the data set in ascending order by variables.

The BY line gives the sorting variable.

NOTE: PROC SORT must be run prior to any PROC that uses a BY statement to divide the data into subgroups for analysis. We will return to this in PROC UNIVARIATE.

PROC PLOT

```
PROC PLOT;
PLOT males*females;
RUN;
```

PROC PLOT is used to make scattergrams. This plot will be text based and restricted to lines of text.

The PLOT line tells SAS which variables to plot in the form (y-axis)*(x-axis).

PROC GPLOT

```
PROC GPLOT;
PLOT males*females;
SYMBOL V=circle;
RUN;
```

PROC GPLOT does the same thing as PROC PLOT except now the plot is created using images making the plot more graphically appealing. The "SYMBOL V=" part tells SAS what how to draw the points. See the SAS online for more information on this procedure if you want to control the plot's appearance beyond the default SAS options. (See end of this document to learn how to use online help.)

PROC UNIVARIATE

Before beginning this section we will start with a new data set containing measurements of heart rates and body temperatures from 130 men and women. The file containing the data is located at

http://www.amstat.org/publications/jse/datasets/normtemp.dat. Gender in this data set is coded as Male =1 and Female = 2. The data step looks like:

```
DATA health;
INPUT temp gender $ hrtrate;
DATALINES;
96.3 1 70
96.7 1 71
96.9 1 74
…
96.4 2 69
96.7 2 62
96.8 2 75
…
;
RUN;
```

Now we can do a PROC UNIVARIATE to produce summary statistics for each gender. Since we will use a BY statement in the PROC UNIVARIATE to analyze the data for each gender separately, we must sort the data first:

```
PROC SORT data=health;
BY GENDER;
RUN;
PROC UNIVARIATE PLOT;
VAR temp;
BY gender;
RUN;
```

PROC UNIVARIATE provides descriptive statistics such as the mean, median, and standard deviation. The additional PLOT statement in the PROC UNIVARIATE line adds a box plot and a normal Q-Q plot.

The VAR statement gives the variable(s) that we want descriptives for.

The BY statement informs SAS to perform separate analyses for each level of gender. Used in conjunction with the PLOT option in PROC UNIVARIATE above, the BY statement will cause SAS to make a side-by-side box plot for each gender.

PROC FREQ

Let us consider the following data step first, which has a new type of input structure:

```
DATA types;
INPUT typeA typeB @@;
DATALINES;
1 1 1 1 1 2 1 2 1 2 2 1 2 2 2 2 2 3 2 3 2 3 2 3 2 3
1 1 1 1 1 2 1 2 1 2 1 2 1 3 1 3 1 3 1 3 1 3 1 3 1 3
2 1 2 1 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 3 2 3 2 3
;
RUN;
```

The @@ in the input statement tells SAS to keep looking for observations on the same line.

Now for the PROC FREQ part:

```
PROC FREQ;
TABLES typeA;
TABLES typeA*typeB;
RUN;
```

PROC FREQ creates tables in a variety of formats.

The first TABLES statement will create a one-way table of frequencies and proportions for the two levels of typeA.

The second TABLES statement creates a two-way table in the form (y-axis)*(x-axis).

PROC SURVEYMEANS

For this section, we will return to using the health data:

```
PROC SURVEYMEANS data=health MEAN STDERR CV CLM ALPHA=.1;
VAR hrtrate;
WHERE gender='M';
RUN;
```

PROC SURVEYMEANS is used to obtain summary statistics such as means, standard errors, and confidence intervals for survey data gathered from several different types of sample designs. For right now we will only look at a simple example in which we assume the data come from a simple random sample, and build on this knowledge through the semester. In the future there will be WEIGHTS, STRATA, DOMAIN, and CLUSTER statements added to the PROC SURVEYMEANS statements above.

The MEAN, STDERR, CV, and CLM notifies SAS to display the mean, standard error of the mean, coefficient of variation, and a confidence interval for the mean. The ALPHA=.1 part makes the interval be a 90% confidence level interval. There are several other statistics that can be asked for here.

The WHERE statement makes SAS do the analysis for males, so only 65 of the 130 observations were used.

Additional Help

For syntax help you can consult the SAS online help. Click the purple book icon with the question mark on it to access the help files. It may be easiest to go to the search mode and type in the PROC that you are interested in, then choose the index version such as "PROC SURVEYMEANS: Index" from the choices provided.

We will discuss syntax in future labs for procedures such as PROC SURVEYMEANS, PROC SURVEYREG and SURVEYSELECT.