

SAS programs consist of an initial DATA step that creates the data set to be analyzed followed by any number of procedures (PROCs) that examine or analyze the data. The basics of the DATA step, general PROC syntax and some specific PROCs are described below. There is a HELP menu in SAS; if you choose this and then select HELP AND DOCUMENTATION you can get details on any aspect of the data step or any procedure.

The DATA Step

The data step is quite powerful, containing many more options than are described here. For example it is possible to merge two data files or append one on to the end of another. It is also possible to incorporate more than one DATA step in a single program. (This can be useful if you wish to perform an analysis once for the entire data set and once for a subset. Use a 2nd DATA step to create the subset with a different working name.) The focus of this document is on getting a single data set into SAS using a single DATA step. There are at least three ways that one can enter data to use in SAS: read an existing text file, import an existing spreadsheet or SAS dataset, or type the data values directly into the program. These are described next:

1. Read an existing text file – This assumes a data file has been set up as a text file before entering SAS. Sample SAS code (on the left) and some comments (on the right) are provided. Note that you don't have to use uppercase for the SAS program I did so to make things easier to read.

```
DATA MYDAT;                                - assign name to the working data set (MYDAT)
  INFILE 'C:/HAL/Courses/Stat210/mydata.txt' FIRSTOBS=2;
                                          - read in text file (firstobs = 2 tells that the first row is not data)
  INPUT Y X1 X2 X3 AGE;                    - identify the variables in the file
  MEANX = (X1 + X2 + X3) / 3;              - create new variables (next few lines illustrate some possible functions)
  SQAGE = SQRT(AGE);
  LOGX3 = LOG(X3);
  IF (X1 > 5) THEN BIGX1 = 1;
      ELSE BIGX1 = 0;
  IF AGE > 20;                              - select only certain cases from the file for this data set
  IF _N_=25 THEN DELETE;                  - delete a single case (case number 25)
RUN;                                       - SAS commands don't execute until a "run" is entered; you don't have
                                          to put one after each DATA step or PROC, just one at the end
```

This program assumes the variables are separated by spaces in the text file. If your input file has the variables in specific columns, then you can identify which columns in the data file correspond to which variables on the INPUT line ('y 1-4 x1 5-8'). You identify a character/text variable by putting a \$ after it's name ('y \$ x1'). One variation on this approach is to use a FILENAME command first – this creates an "abbreviated" name by which you can refer to the file. If you do this, then the first three lines would read:

```
FILENAME MYSOURCE 'C:/HAL/Courses/Stat210/mydata.txt';
DATA MYDAT;
  INFILE MYSOURCE FIRSTOBS=2;
```

2. Import data using SAS Import Data options – You can import a variety of data types. Use the File→ImportData menu item and navigate to the data source. I will always provide Excel (.XLS) versions of our data sets and these are easily imported. (Note: you can import text files but this does not always work so well.) Fill in the ImportData box (i.e., find the file you want to import). You should place the data in your WORK directory when you are prompted for a home and then give it a name. This is now saved as a SAS dataset and can be used in SAS programs. You refer to the data set as WORK.NAME where NAME is the name you supplied. If no further data processing is required, then you can go directly to your procedures, e.g., to print the data you would use:

```
PROC PRINT DATA=WORK.NAME;
RUN;
```

If you want to do some data processing then you would need a data step. For example if you decide to work with the logarithm of the variable y rather than the actual recorded rainfall, then you might have code like the

following which adds a new variable ($\log(y)$) to the data set. Your data analysis procedures would then refer to the new data set (MYNEWDATA) rather than (WORK.NAME).

```
DATA MYNEWDATA;  
    SET WORK.NAME;  
    LOGY = LOG(Y);  
RUN;
```

3. Enter data during a SAS session – I rarely use this since it only works with very small data sets.

```
DATA MYDAT;  
    INPUT Y X1 X2 X3 AGE;  
    SQAGE = SQRT(AGE);  
    CARDS;  
10 1 1 5 32  
20 1 2 7 24  
18 0 4 6 21  
;  
RUN;
```

PROCEDURES (also known as PROCs)

Once a working data set has been created you will want to look at the data or carry out analyses using various procedures. All procedures use the most recently created working data set unless the DATA=xxxx option is used on the first line of the procedure. It is usually a good idea to use the DATA=xxxx option to explicitly identify the data set that you are working with so that there are no unpleasant surprises. The “xxxx” refers to the name given the dataset when it was created in the DATA step. The general form of a PROC is as follows:

```
PROC procedurename options;           (each procedure has different permissible options)  
    VAR varname1 varname2 ...;         (identify variables to include)  
    MODEL y = x1 x2 / options..;      (specify model for regression/anova and options for the model)  
    WEIGHT wtvar;                     (specify a weighting variable; we won't use this often)  
    CLASS varname1 varname2 ...;      (specify which variables are categorical)  
    BY byvar1 byvar2;                 (defines groups for the procedure to be performed on; data must  
    RUN;                               first be sorted on these variables using PROC SORT;)
```

Not every PROC will require or even recognize all of these commands but the above is a kind of prototype PROC. The remainder of the document illustrates some useful procedures. Sample code appears first followed by comments.

```
PROC PRINT DATA=MYDAT;  
    VAR Y X1 X2;
```

This procedure prints out the identified variables. If you leave out the variable list, then it prints all variables.

```
PROC SORT DATA=MYDAT;  
    BY GENDER;
```

This procedure sorts the data by whatever variables are identified in ascending numerical or alphabetical order. Using “BY REVERSE AGE” would sort in descending order.

```
PROC SORT DATA=MYDAT;  
    BY GENDER;  
PROC PRINT DATA=MYDAT;  
    VAR Y X1 X2;  
    BY GENDER;
```

The code above is a way to print data separately by gender. You need to sort first (unless by some miracle the data were entered with all males first and all females second).

```
PROC MEANS DATA=MYDAT;  
  VAR X1 X2 X3;
```

This prints basic summary statistics (mean, s.d., min, max). You can include options on the PROC line specifying specific statistics you would like to see as in PROC MEANS N NMISS MEAN STD CV MIN MAX SKEWNESS KURTOSIS; One might often include a BY command here to get summary statistics for different groups (again you have to sort first to do this).

```
PROC UNIVARIATE DATA=MYDAT;  
  VAR X1 X2 X3;
```

Print detailed univariate statistics report. This basically does everything that PROC MEANS can possibly do. There is a valuable PLOT option that would go on the PROC line telling it to create several summary graphics (though these are somewhat crudely rendered).

```
PROC FREQ;  
  TABLES var1*var2 var1*var2*var3;
```

Create tables to display counts of different categorical variables.

```
PROC GCHART;  
  VBAR var1 / TYPE=FREQ;  
  VBAR var1 var2 / TYPE=PCT;  
  VBAR var1 / TYPE=FREQ MIDPOINTS=0 TO 100 BY 5;  
  HBAR var1 / TYPE=FREQ;  
  PIE var1 var2;
```

This procedure makes nice graphs. VBAR is for a vertical bar chart or histogram and HBAR is for a horizontal bar chart. PIE is for a pie chart. Several common plot options (TYPE, MIDPOINTS) are illustrated above.

```
PROC GPLOT;  
  PLOT var1*var2;  
  PLOT var1*var3="+";  
  PLOT var1*var4=var5;
```

This procedure creates scatterplots. The basic scatterplot puts a default symbol at each point. The second and third lines show how to change the plotting symbol or plot the value of a 3rd variable at coordinates determined by two other variables. Note there are also options for controlling the axes (not shown here).

```
PROC PLOT;  
  PLOT var1*var2 / VAXIS = 0 TO 80 BY 5 VPOS=25 HPOS=60;
```

This procedure is not recommended. It is an "old" scatterplot command that creates character (low-resolution) graphs. I have used this on occasion so that the graph fits nicely into the rest of the output but would not recommend you use it.

We will likely use the following PROCs during the semester. Details and examples will be provided as we cover them.

```
PROC TTEST;      - two sample tests  
PROC ANOVA;     - analysis of variance  
PROC GLM;       - generalized analysis of variance  
PROC REG;       - least squares regression
```