

Lab1: Introduction to SAS
STT 422: Summer, 2004
Vince Melfi

Most of the methods we'll learn in this course are computationally intensive, and require a reasonably sophisticated statistical package. We will learn to use SAS, which is one of the most popular statistical packages both inside and outside of universities. We will be using SAS in the Windows microlabs, and I assume that you have used Windows-based software before. If not, please let me know right away, and I will try to help you learn to use Windows well enough to use SAS.

Starting SAS

1. Log on to the computer using your pilot ID and password.
2. Start SAS by clicking on the **Start** menu, then choosing **Programs**, then **Math Apps**, then **sas**, and then **sas8**. Since we'll often be using menus in our computer work, I'll use the symbol ► to indicate a submenu. For example, starting SAS can be abbreviated **Start ► Programs ► Math Apps ► sas ► sas8**.
3. When SAS starts, you will probably see a few windows. For now, it will be important to have an **Editor** or **Program Editor** window¹, a **Log** window, and an **Output** window. If one or more of these is missing from your display, you can view it via the **View** menu. For example, **View ► Program Editor** will open up the **Program Editor** window.

A first SAS program

Type the following lines, up to and including **run;**, in the **Program Editor** window. Once you've done this, use **Run ► Submit** to run the program. (Note that the **Program Editor** window has to be on top in order for the **Run** menu to appear.) If all goes well, you should see the output in the **Output** window. I'll explain each of the sections of the program below.

```
data baberuth;
input year homerun;
cards;
1914 0
1915 4
1916 3
1917 2
1918 11
1919 29
1920 54
;
```

¹The Editor, also known as the Enhanced Editor, is a bit nicer version of the Program Editor. I'll always call it the Program Editor, but either will work.

```

proc print data = baberuth;
title 'Babe Ruth HR data';

proc univariate data = baberuth;
var homerun;
title 'Summary Statistics for Babe Ruth HR data';

run;

```

Take a look at the **Output** window first. This should contain two chunks. The first is just a printout of the data, which should be titled “Babe Ruth HR data.” This was created by the `proc print` chunk of the program. The second chunk is a printout of some summary statistics, and was created by the `proc univariate` part of the program.

More Details

Here’s an more complete explanation of what we just did, starting with some basic structure and rules. First, most SAS programs consist of **data steps** and **procedures**. Loosely, data steps create and manipulate data, while procedures analyze the data in some way. Second, all SAS program lines must end with a semicolon. Third, variable names should be eight or fewer characters long, and should contain only letters and numbers. Also, they should not begin with a number. For example, `2yearold` would not be a valid variable name. Also, the blank lines are not necessary, but help in program readability.

Step by step, here’s what we did. First, the data step.

1. The **data baberuth** statement named the data set “baberuth.”
2. The **input** statement named the variables in the dataset “year” and “homerun”.
3. The **cards** statement told SAS that we were planning to enter the data at the keyboard. (The name “cards” is a relic from the days when programmers worked with punched cards rather than at a keyboard.)
4. The lines after **cards** and until the line containing only a semicolon contained the actual values for the two variables. For example, the first of these lines contains `1914 0`. Thus the first year is 1914, in which Babe Ruth hit 0 home runs.

Now, the procedures.

1. The line **proc print data = baberuth** told SAS that we wanted to apply the `print` procedure to the `baberuth` dataset.
2. The line **title 'Babe Ruth HR data'** gave SAS a title to put at the top of the resulting output. Note that the title is enclosed in single quotes. It’s a good idea to title outputs this way.
3. The line **proc univariate data = baberuth** began the `univariate` procedure.

4. The line **var homerun** told SAS to apply the procedure only to the variable **homerun**.
5. The line **title 'Summary Statistics for Babe Ruth HR data'** titled the output.
6. The **run** line told SAS to run the above steps when the program was submitted. All programs end with a **run** line.

Getting data from a file

Entering data at the keyboard soon gets tedious. The next SAS program illustrates how to get data from a file. All the data from the text has been placed on the U: drive under the directory `msu\course\stt\422\summer04\Text`, so you'll be able to do text exercises and examples without entering the data the old-fashioned way. The files are all plain text files. For example, the Babe Ruth data is from Example 1.4, so it is in `U:\msu\course\stt\422\summer04\Text\Ch01\EG01_004.TXT`.²³ Open this file in Notepad or another text editor just to see what it looks like. Once you've done this, enter and run the following SAS program.

```
data baberuth;
infile 'U:\msu\course\stt\422\summer04\Text\Ch01\EG01_004.TXT';
input year homerun;

proc print data = baberuth;
title 'Babe Ruth Home Run Data';

proc gplot;
  plot homerun*year;
  title 'Plot of Babe Ruth Home Runs';

run;
```

Much of this should make sense to you already.

One major difference from the previous program is the omission of the **cards** line and the fact that we didn't have to enter the data. Instead, we specify the file that contains the data via the **infile** statement, and again specify the variables via the **input** statement. The **proc print** section is the same as before. Note from the output, however that we now have the complete data from 1914 through 1935.

A new **proc** is **gplot**. This is the basic plotting procedure. In our case, we specify a scatterplot of **homerun** versus **year** via the line **plot homerun*year** and then specify a title. Note also that I've indented the lines below **proc gplot** that serve to expand on what we want out of the procedure. This is good programming practice for readability, although it doesn't affect how SAS treats the program.

²In the data from the text, the prefix "EG" stands for "example;" the prefix "EX" stands for "exercise;" and the prefix "TA" stands for "table."

³The data file that comes with the text contains only the homerun totals and doesn't contain the years. I've replaced it with a file that contains both.

More information

The SAS Help system provides a huge amount of information about using SAS, although not necessarily in an easy-to-use format. As an example, you can learn more about **proc univariate** by using **Help ► Sas System Help**, then clicking on **Index** in the left window, then typing **univariate** as the keyword, and then double-clicking on univariate below. Look at the **Introduction** and **Syntax** in the right-hand window.

You also might find it worthwhile to spend a few minutes on two of the tutorials, **Tour SAS windows** and **Work with SAS programs**. To do this, go to **Help ► Getting Started with SAS software**. Then click on the graphic and choose the two tutorials above.

SAS Exercise

Do Exercise 1.67 on Page 61 of the text using SAS. A hint to save some time: If you select all the text in the **Program Editor** window, it won't disappear when you submit the program. Since you'll want to rerun the program a few times, just changing the numbers slightly, this is a good idea!