7.4: Introduction to SAS

Learning Objectives

- This page gives an introduction to using the statistical software package SAS. Some of it is specific to the University of Delaware, but most of it should be useful for anyone using SAS.

Introduction

SAS, SPSS and Stata are some of the most popular software packages for doing serious statistics. I have a little experience with SAS, so I've prepared this web page to get you started on the basics. UCLA's Academic Technology Services department has prepared very useful guides to SAS, SPSS and Stata.

An increasingly popular tool for serious statistics is R, a free software package for Windows, Mac, Linux, and Unix. There are free online manuals, and many online and printed tutorials. I've never used R, so I can't help you with it.

SAS may seem intimidating and old-fashioned; accomplishing anything with it requires writing what is, in essence, a computer program, one where a misplaced semicolon can have disastrous results. But I think that if you take a deep breath and work your way patiently through the examples, you'll soon be able to do some pretty cool statistics.

The instructions here are for the University of Delaware, but most of it should apply anywhere that SAS is installed. There are four ways of using SAS:

- on a mainframe, in batch mode. This is what I'll describe below.
- on a mainframe, interactively in line mode. I don't recommend this, because it just seems to add complication and confusion.
- on a mainframe, interactively with the Display Manager System. From what I've seen, this isn't very easy. If you really want to try it, here are instructions. Keep in mind that "interactive" doesn't mean "user friendly graphical interface like you're used to"; you still have to write the same SAS programs.
- on your Windows personal computer. I've never done this. Before you buy SAS for your computer, see if you can use it for free on your institution's mainframe computer.

To use SAS on a mainframe computer, you'll have to connect your personal computer to a the mainframe; at the University of Delaware, you connect to a computer called Strauss. The operating system for mainframes like Strauss is Unix; in order to run SAS in batch mode, you'll have to learn a few Unix commands.

Getting connected to a mainframe from a Mac

On a Mac, find the program Terminal; it should be in the Utilities folder, inside your Applications folder. You'll probably want to drag it to your taskbar for easy access in the future. The first time you run Terminal, go to "Preferences" in the Terminal menu, choose "Settings", then choose "Advanced". Set "Declare terminal as:" to "vt100". Then check the box that says "Delete sends Ctrl-H". (Some versions of Terminal may have the preferences arranged somewhat differently, and you may need to look for a box to check that says "Delete key sends backspace.") Then quit and restart Terminal. You won't need to change these settings again.

When you start up Terminal, you'll get a prompt that looks like this:

Your-Names-Computer:~ yourname$

After the dollar sign, type `ssh userid@computer.url`, where `userid` is your user id name and computer.url is the address of the mainframe. At Delaware the mainframe is Strauss, so if your userid is joeblow, you'd type `ssh joeblow@strauss.udel.edu`. Then hit Return. It will ask you for your password; type it and hit Return (it won't look like you've typed anything, but it will work). You'll then be connected to the mainframe, and you'll get a prompt like this:

strauss.udel.edu%

You're now ready to start typing Unix commands.

Getting connected to a mainframe from Windows

Unlike Macs, Windows computers don't come with a built-in terminal emulator, so you'll need to ask your site administrator which "terminal emulator" they recommend. PuTTY is one popular (and free) program, with a good set of instructions here. Whichever terminal emulator you use, you'll need to enter the "host name" (the name of the mainframe computer you're trying to connect to; at

Delaware, it's strauss.udel.edu), your user ID, and your password. You may need to specify that your "Protocol" is "SSH". When you type your password, it may look like nothing's happening, but just type it and hit Enter. If it works, you'll be connected to the mainframe and get a prompt like this:

strauss.udel.edu%

You're now ready to start typing Unix commands.

Getting connected to a mainframe from Linux

If you're running Linux, you're already enough of a geek that you don't need my help getting connected to your mainframe.

A little bit of Unix

The operating system for mainframes like Strauss is Unix, so you've got to learn a few Unix commands. Unix was apparently written by people for whom typing is physically painful, as most of the commands are a small number of cryptic letters. Case does matter; don't enter `CD` and think it means the same thing as `cd`. Here is all the Unix you need to know to run SAS. Commands are in **bold** and file and directory names, which you choose, are in *italics*.

ls	Lists all of the file names in your current directory.
pico *filename*	pico is a text editor; you'll use it for writing SAS programs. Enter **pico** *yourfilename.sas* to open an existing file named *yourfilename.sas*, or create it if it doesn't exist. To exit pico, enter the **control** and **x** keys. You have to use the arrow keys, not the mouse, to move around the text once you're in a file. For this reason, I prefer to create and edit SAS programs in a text editor on my computer (TextEdit on a Mac, NotePad on Windows), then copy and paste them into a file I've created with pico. I then use pico for minor tweaking of the program. Don't copy and paste from a word processor like Word into pico, as word processor files contain invisible characters that will confuse pico. Note that there are other popular text editors, such as vi and emacs, and one of the defining characters of a serious computer geek is a strong opinion about the superiority of their favorite text editor and total loserness of all other text editors. To avoid becoming one of them, try not to get emotional about pico. Unix filenames should be made of letters and numbers, dashes (-), underscores (_), and periods. Don't use spaces or other punctuation (slashes, parentheses, exclamation marks, etc.), as they have special meanings in Unix and may confuse the computer. It is common to use an extension after a period, such as *.sas* to indicate a SAS program, but that is for your convenience in recognizing what kind of file it is; it isn't required by Unix.
cat *filename*	Opens a file for viewing and printing, but not editing. It will automatically take you to the end of the file, so you'll have to scroll up. To print, you may want to copy what you want, then paste it into a word processor document for easier formatting. You should use **cat** instead of **pico** for viewing the output files (.log and .lst) that SAS creates.

mv *oldname newname*	Changes the name of a file from *oldname* to *newname*. When you run SAS on the file *practice.sas*, the output will be in a file called *practice.lst*. Before you make changes to *practice.sas* and run it again, you may want to change the name of *practice.lst* to something else, so it won't be overwritten.
cp *oldname newname*	Makes a copy of file *oldname* with the name *newname*.
rm *filename*	Deletes a file.
logout	Logs you out of the mainframe.
mkdir *directoryname*	Creates a new directory. You don't need to do this, but if you end up creating a lot of files, you may find it helpful to keep them organized into different directories.
cd *directoryname*	Changes from one directory to another. For example, if you have a directory named *sasfiles* in your home directory, enter **cd** *sasfiles*. To go from within a directory up to your home directory, just enter **cd**.
rmdir *directoryname*	Deletes a directory, if it doesn't have any files in it. If you want to delete a directory and the files in it, first go into the directory, delete all the files in it using **rm**, then delete the directory using **rmdir**.
sas *filename*	Runs SAS. Be sure to enter **sas** *filename.sas*. If you just enter **sas** and then hit return, you'll be in interactive SAS mode, which is scary; enter **;endsas;** if that happens and you need to get out of it.

Writing a SAS program

To use SAS, you first use pico to create an empty file; you can call the first one *practice.sas*. Type in the SAS program that you've written (or copy it from a text file you created with TextEdit or Notepad), then save the file by hitting the control and x keys. Once you've exited pico, enter `sas practice.sas`; the word `sas` is the command that tells Unix to run the SAS program, and `practice.sas` is the file it is going to run SAS on. SAS then creates a file named *practice.log*, which reports any errors. If there are no fatal errors, SAS also creates a file named *practice.lst*, which contains the results of the analysis.

The SAS program (which you write using pico) consists of a series of commands. Each command is one or more words, followed by a semicolon. You can put comments into your program to remind you of what you're trying to do; these comments have a slash and asterisk on each side, like this:

```
/*This is a comment. It is not read by the SAS program.*/
```

The SAS program has two basic parts, the DATA step and the PROC step. (Note--I'll capitalize all SAS commands to make them stand out, but you don't have to when you write your programs; unlike Unix, SAS is not case-sensitive.) The DATA step reads in data, either from another file or from within the program.

In a DATA step, you first say "DATA *dataset*;" where *dataset* is an arbitrary name you give the dataset. Then you say "INPUT *variable1 variable2*...;" giving an arbitrary name to each of the variables that is on a line in your data. So if you have a data set consisting of the length and width of mussels from two different species, you could start the program by writing:

```
DATA mussels;
INPUT species $ length width;
```

A variable name for a nominal variable (a name or character) has a space and a dollar sign ($) after it. In our practice data set, "species" is a nominal variable. If you want to treat a number as a nominal variable, such as an ID number, remember to put a dollar sign after the name of the variable. Don't use spaces within variable names or the values of variables; use `Medulis` or `M_edulis`, not `M. edulis` (there are ways of handling variables containing spaces, but they're complicated).

If you are putting the data directly in the program, the next step is a line that says "DATALINES;", followed by the data. A semicolon on a line by itself tells SAS it's done reading the data. You can put each observation on a separate line, with the variables separated by one or more spaces:

```
DATA mussels; /* names the data set "mussels" */
INPUT species $ length width; /* names the variables, defines "species" as a nominal variable */
DATALINES; /* tells SAS that the data starts on the next line */
edulis 49.0 11.0
trossulus 51.2 9.1
trossulus 45.9 9.4
edulis 56.2 13.2
edulis 52.7 10.7
edulis 48.4 10.4
trossulus 47.6 9.5
trossulus 46.2 8.9
trossulus 37.2 7.1
; /* the semicolon tells SAS to stop reading data */
```

You can also have more than one set of data on each line, if you put "@@" at the end of the INPUT statement:

```
DATA mussels;
INPUT species $ length width @@;
DATALINES;
edulis 49.0 11.0 trossulus 51.2 9.1 trossulus 45.9 9.4 edulis 56.2 13.2
edulis 52.7 10.7 edulis 48.4 10.4 trossulus 47.6 9.5 trossulus 46.2 8.9
trossulus 37.2 7.1
;
```

If you have a large data set, it will be more convenient to keep it in a separate file from your program. To read in data from another file, use an INFILE *datafile*; statement, with the name of the data file in single quotes. If you do this, you don't use the DATALINES statement. Here I've created a separate file (in the same directory) called "shells.dat" that has a huge amount of data in it, and this is how I tell SAS to read it:

```
DATA mussels;
INFILE 'shells.dat';
INPUT species $ length width;
```

When you have your data in a separate file, it's a good idea to have one or more lines at the start of the file that explain what the variables are. You should then use FIRSTOBS=*linenumber* as an option in the INFILE statement to tell SAS which line has the first row of data. Here I tell SAS to start reading data on line 3 of the shells.dat data file, because the first two lines have explanatory information:

```
DATA mussels;
INFILE 'shells.dat' FIRSTOBS=3;
INPUT species $ length width;
```

The DATA statement can create new variables from mathematical operations on the original variables. Here I make two new variables, "loglength," which is just the base-10 log of length, and "shellratio," the width divided by the length. SAS can do statistics on these variables just as it does on the original variables.

```
DATA mussels;
INPUT species $ length width;
loglength=log10(length);
shellratio=width/length;
DATALINES;
```

The PROC step

Once you've entered in the data, it's time to analyze it using one or more PROC commands. The PROC statement tells SAS which procedure to run, and almost always has some options. For example, to calculate the mean and standard deviation of the lengths, widths, and log-transformed lengths, you would use PROC MEANS. It is followed by certain options. DATA=*dataset* tells it which data set to analyze. MEAN and STD are options that tell PROC MEANS to calculate the mean and standard deviation; there are several other options that could have been used with PROC MEANS. VAR *variables1 variable2 ...* tells PROC MEANS which variables to calculate the mean and standard deviation of. RUN tells SAS to run.

```
PROC MEANS DATA=mussels MEAN STD; /* tells PROC MEANS to calculate mean and standard deviation
*/
VAR length width loglength; /* tells PROC MEANS which variables to analyze */
RUN; /* makes PROC MEANS run */
```

Now that you've read through a basic introduction to SAS, put it all together and run a SAS program. Connect to your mainframe and use pico to create a file named "practice.sas". Copy and paste the following into the file:

```
DATA mussels;
INPUT species $ length width;
loglength=log10(length);
shellratio=width/length;
DATALINES;
edulis 49.0 11.0
tross 51.2 9.1
tross 45.9 9.4
edulis 56.2 13.2
edulis 52.7 10.7
edulis 48.4 10.4
tross 47.6 9.5
tross 46.2 8.9
tross 37.2 7.1
;
PROC MEANS DATA=mussels MEAN STD;
VAR length width loglength;
RUN;
```

Then exit pico (hit control-x). At the dollar sign prompt, enter `sas practice.sas`. Then enter `ls` to list the file names; you should see new files named `practice.log` and `practice.lst`. First, enter `cat practice.log` to look at the log file. This will tell you whether there are any errors in your SAS program. Then enter `cat practice.lst` to look at the output from your program. You should see something like thi

The SAS System: The MEANS Procedure

```
Variable Mean Std Dev
----------------------------------------
length 48.2666667 5.2978769
width 9.9222222 1.6909892
loglength 1.6811625 0.0501703
```

If you do, you've successfully run SAS. Yay!

PROC SORT and PROC PRINT

I describe specific statistical procedures on the web page for each test. Two that are of general use are PROC SORT and PROC PRINT. PROC SORT sorts the data by one or more variables. For some procedures, you need to sort the data first. PROC PRINT writes the data set, including any new variables you've created (like loglength and shellratio in our example) to the output file. You

can use it to make sure that SAS has read the data correctly, and your transformations, sorting, etc. have worked properly. You can sort the data by more than one variable; this example sorts the mussel data, first by species, then by length.

```
PROC SORT DATA=mussels;
BY species length;
RUN;
PROC PRINT DATA=mussels;
RUN;
```

Adding PROC SORT and PROC PRINT to the SAS file produces the following output:

The SAS System

Obs species length width loglength shellratio

1 edulis 48.4 10.4 1.68485 0.21488
2 edulis 49.0 11.0 1.69020 0.22449
3 edulis 52.7 10.7 1.72181 0.20304
4 edulis 56.2 13.2 1.74974 0.23488
5 trossulus 37.2 7.1 1.57054 0.19086
6 trossulus 45.9 9.4 1.66181 0.20479
7 trossulus 46.2 8.9 1.66464 0.19264
8 trossulus 47.6 9.5 1.67761 0.19958
9 trossulus 51.2 9.1 1.70927 0.17773

As you can see, the data were sorted first by species, then within each species, they were sorted by length.

Graphs in SAS

It's possible to draw graphs with SAS, but I don't find it to be very easy. I recommend you take whatever numbers you need from SAS, put them into a spreadsheet or specialized graphing program, and use that to draw your graphs.

Getting data from a spreadsheet into SAS

I find it easiest to enter my data into a spreadsheet first, even if I'm going to analyze it using SAS. But if you try to copy data directly from a spreadsheet into a SAS file, the numbers will be separated by tabs, which SAS will choke on; your log file will say "NOTE: Invalid data in line...". To get SAS to recognize data separated by tabs, use the DELIMITER option in an INFILE statement. For inline data, add an INFILE DATALINES DELIMITER='09'x; statement before the INPUT statement (SAS calls tabs '09'x):

```
DATA mussels;
INFILE DATALINES DELIMITER='09'x;
INPUT species $ length width;
DATALINES;
edulis 49.0 11.0
tross 51.2 9.1
tross 45.9 9.4
edulis 56.2 13.2
edulis 52.7 10.7
edulis 48.4 10.4
tross 47.6 9.5
tross 46.2 8.9
tross 37.2 7.1
;
```

If your data are in a separate file, you include DELIMITER='09'x in the INFILE statement like this:

```
DATA mussels;
INFILE 'shells.dat' DELIMITER='09'x;
INPUT species $ length width;
```

More information about SAS

The user manuals for SAS are available online for free. They're essential for advanced users, but they're not very helpful for beginners.

The UCLA Academic Technology Services has put together an excellent set of examples of how to do the most common statistical tests in SAS, SPSS or Stata; it's a good place to start if you're looking for more information about a particular test.

This page titled 7.4: Introduction to SAS is shared under a not declared license and was authored, remixed, and/or curated by John H. McDonald via source content that was edited to the style and standards of the LibreTexts platform.