A Step by Step Guide to Learning SAS

The Fundamentals of SAS Programming
and an Introduction to
Simple Linear Regression Models

September 29th, 2003

Anjali Mazumder

Objective

- Familiarize yourselves with the SAS programming environment and language.
- Learn how to create and manipulate data sets in SAS and how to use existing data sets outside of SAS.
- Learn how to conduct a regression analysis.
- Learn how to create simple plots to illustrate relationships.

LECTURE OUTLINE

- Getting Started with SAS
- Elements of the SAS program
- Basics of SAS programming
- Data Step
- Proc Reg and Proc Plot
- Example
- Tidbits
- Questions/Comments

Getting Started with SAS

1.1 Windows or Batch Mode?

- 1.1.1 Pros and Cons
- 1.1.2 Windows
- 1.1.3 Batch Mode

Reference:

www.cquest.utoronto.ca/stats/sta332s/sas.html

1.1.1 Pros and Cons

Windows:

Pros:

- SAS online help available.
- You can avoid learning any Unix commands.
- Many people like to point and click.

Cons:

- SAS online help is incredibly annoying.
- Possibly very difficult to use outside CQUEST lab.
- Number of windows can be hard to manage.

1.1.1 cont'd…

Batch Mode:

Pros:

- Easily usable outside CQUEST labs.
- Simpler to use if you are already familiar with Unix.
- Established Unix programs perform most tasks better than SAS's builtin utilities.

Cons:

- Can't access SAS's online help.
- Requires some basic knowledge of Unix.

1.1.2 Windows

- You can get started using either of these two ways:
  1. Click on **Programs** at the top left of the screen and select **CQUEST_APPLICATIONS** and then **sas**.
  2. In a terminal window type: `sas`

A bunch of windows will appear – don't get scared!

1.1.3 Batch Mode

- First, make sure you have set up your account so you can use batch mode.
- Second, you need to create a SAS program.
- Then ask SAS to run your program (`foo`) using the command:

`sas foo` or `sas foo.sas`

Either way, SAS will create files with the same name as your program with respective extensions for a log and output file (if there were no fatal errors).

1.2 SAS Help

- If you are running SAS in a window environment then there is a online SAS available.
- How is it helpful?

  You may want more information about a command or some other aspect of SAS then what you remember from today or that is in this guide.
- How to access SAS **Help**?
  1. Click on the **Help** button in task bar.
  2. Use the menu command – Online documentation
- There are three tabs: **Contents**, **Index** and **Find**

1.3 SAS Run

- If you are running SAS in a window environment then simply click on the Run Icon. It's the icon with a picture of a person running!
- For Batch mode, simply type the command: *`filename`*`.sas`

Elements of the SAS Software

2.1 SAS Program Editor: Enhanced Editor

2.2 Important SAS Windows: Log and Output Windows

2.3 Other SAS Windows: Explorer and Results Windows

2.1 SAS Program Editor

- What is the Enhanced Editor Window?

This is where you write your SAS programs. It will contain all the commands to run your program correctly.

- What should be in it?

All the essentials to SAS programming such as the information on your data and the required steps to conduct your analysis as well as any comments or titles should be written in this window (for a single problem). See Section 3-6.

- Where should I store the files?

In your home directory. SAS will read and save files directly from there.

2.2 Log and Output Windows

- How do you know whether your program is syntactically correct?

Check the Log window every time you run a program to check that your program ran correctly – at least syntactically. It will indicate errors and also provide you with the run time.

- You ran your program but where's your output?

There is an output window which uses the extension `.lst` to save the file.

If something went seriously wrong – evidence will appear in either or both of these windows.

2.3 Other SAS Windows

- There are two other windows that SAS executes when you start it up: Results and Explorer Windows
- Both of these can be used as data/file management tools.
- The Results Window helps to manage the contents of the output window.
- The SAS Explorer is a kind of directory navigation tool. (Useful for heavy SAS users).

Basics of SAS Programming

3.1 Essentials
- 3.1.1 A program!
- 3.1.2 End of a command line/statement
- 3.1.3 Run Statement

3.2 Extra Essentials
- 3.2.1 Comments
- 3.2.2 Title
- 3.2.3 Options
- 3.2.4 Case (in)sensitivity

3.1 Essentials of SAS Programming

3.1.1 Program

- You need a program containing some SAS statements.
- It should contain one or more of the following:

1) data step: consists of statements that create a data set
2) proc step: used to analyze the data

3.1 cont'd…

3.1.2 End of a command line or statement

- Every statement requires a semi-colon (;) and hit enter afterwards. Each statement should be on a new line.
- This is a very common mistake in SAS programming – so check very carefully to see that you have placed a ; at the end of each statement.

3.1.3 Run command or keyword

- In order to run the SAS program, type the command: `run;` at the end of the last data or proc step.
- You still need to click on the running man in order to process the whole program.

3.2 Extra Essentials of SAS Programming

3.2.1 Comments

- In order to put comments in your SAS program (which are words used to explain what the program is doing but not which SAS is to execute as commands), use `/*` to start a comment and `*/` to end a comment. For example,

```
/* My SAS commands go here. */
```

3.2 cont'd…

3.2.2 Title

- To create a SAS title in your output, simply type the command:

```
Title 'Regression Analysis of Crime Data';
```

- If you have several lines of titles or titles for different steps in your program, you can number the title command. For example,

```
Title1 'This is the first title';
Title2 'This is the second title';
```

- You can use either single quotes or double quotes. Do not use contractions in your title such as don't or else it will get confused with the last quotation mark.

3.2 cont'd…

3.2.3 Options

- There is a statement which allows you to control the line size and page size. You can also control whether you want the page numbers or date to appear. For example,

```
options nodate nonumber ls=78 ps=60
```

3.2.4 Case (in)sensitivity

- SAS is not case sensitive. So please don't use the same name - once with capitals and once without, because SAS reads the word as the same variable name or data set name.

4. Data Step

- 4.1 What is it?
- 4.2 What are the ingredients?
- 4.3 What can you do within it?
- 4.4 Some Basic Examples
- 4.5 What can you do with it?
- 4.6 Some More Examples

4.1 What is a Data Step?

- A data step begins by setting up the data set. It is usually the first big step in a SAS program that tells SAS about the data.
- A data statement names the data set. It can have any name you like as long as it starts with a letter and has no more than eight characters of numbers, letters or underscores.
- A data step has countless options and variations. Fortunately, almost all your DATA sets will come prepared so there will be little or no manipulation required.

4.2 Ingredients of a Data Step

4.2.1 Input statement

- INPUT is the keyword that defines the names of the variables. You can use any name for the variables as long as it is 8 characters.
- Variables can be either numeric or character (also called alphanumeric). SAS will assume that variables are numeric unless specified. To assign a variable name to have a character value use the dollar sign $.

4.2.2 Datalines statement (internal raw data)

- This statement signals the beginning of the lines of data.
- A ; is placed both at the end of the datalines staement and on the line following the last line of data.
- Spacing in data lines does matter.

4.2 cont'd…

4.2.3 Raw Data Files

- The `datalines` statement is used when referring to internal raw data files.
- The `infile` statement is used when your data comes from an external file. The keyword is placed directly before the `input` statement. The path and name are enclosed within single quotes. You will also need a `filename` statement before the data step.
- Here are some examples of infile statements under 1) windows and 2) UNIX operating environments:

```
1) infile 'c:\MyDir\President.dat';
2) infile '/home/mydir/president.dat';
```

4.3 What can you do within it?

- A data step not only allows you to create a data set, but it also allows you to manipulate the data set.
- For example, you may wish to add two variables together to get the cumulative effect or you may wish to create a variable that is the log of another variable (Meat example) or you may simply want a subset of the data. This can be done very easily within a data step.
- More information on this will be provided in a supplementary documentation to follow.

4.4.1 Basic Example of a Data Step

```
options ls=79;

data meat;
  input steer time pH;
  datalines;
 1 1 7.02
 2 1 6.93
 3 2 6.42
 4 2 6.51
 5 4 6.07
 6 4 5.99
 7 6 5.59
 8 6 5.80
 9 8 5.51
10 8 5.36
;
```

4.4.2 Manipulating the Existing Data

```
options ls=79;

data meat;
  input steer time pH;
  logtime=log(time);
  datalines;
 1 1 7.02
 2 1 6.93
 3 2 6.42
 4 2 6.51
 5 4 6.07
 6 4 5.99
 7 6 5.59
 8 6 5.80
 9 8 5.51
10 8 5.36
;
```

4.4.3 Designating a Character Variable

```
options ls=79;

/*
Data on Violent and Property Crimes in 23 US Metropolitan Areas
violcrim = number of violent crimes
propcrim = number of property crimes
popn = population in 1000's
*/

data crime;
/* city is a character valued-variable so it is followed by
   a dollar sign in the input statement */
  input city $ violcrim propcrim popn;
  datalines;
AllentownPA 161.1 3162.5 636.7
BakersfieldCA 776.6 7701.3 403.1
;
```

4.4.4 Data from an External File

```
options nodate nonumber ls=79 ps=60;

filename datain 'car.dat';

data cars;
infile datain;
input mpg;
  datalines;
/* some data goes here */
;
```

4.5 What can you do with it?

4.5.1 View the data set

- Suppose that you have done some manipulation to the original data set. If you want to see what has been done, use a `proc print` statement to view it.

```
proc print data=meat;
run;
```

4.5 cont'd…

4.5.2 Create a new from an old data set

- Suppose you already have a data set and now you want to manipulate it but want to keep the old as is. You can use the `set` statement to do it.

4.5.3 Merge two data sets together

- Suppose you have created two datasets about the sample (subjects) and now you wish to combine the information. You can use a `merge` statement. There must be a common variable in both data sets to merge.

4.6 Some Comments

- If you don't want to view all the variables, you can use the keyword `var` to specify which variables the `proc print` procedure should display.
- The command `by` is very useful in the previous examples and of the procedures to follow. We will take a look at its use through some examples.
- Let's look at the Meat Example again using SAS to demonstrate the steps explained in 4.5.

5. Regression Analysis

5.1 What is `proc reg`?

5.2 What are the important ingredients?

5.3 What does it do?

5.4 What else can you do with it?

5.5 The cigarette example

5.6 The Output – regression analysis

5.1 Proc Reg

- What is a `proc` procedure?

It is a procedure used to do something to the data – sort it, analyze it, print it, or plot it.

- What is `proc reg`?

It is a procedure used to conduct regression analyses. It uses a `model` statement to define the theoretical model for the relationship between the independent and dependent variables.

5.2 Ingredients of Proc Reg

5.2.1 General Form

```
proc reg data=somedata <options>;
by variables;
model dependent=independent
  <options>;
plot yvar*xvar <options>;
run;
```

5.2 cont'd…

5.2.2 What you need and don't need?

- You **need** to assign 1) the data to be analyzed, and 2) the theoretical model to be fit to the data.
- You don't need the other statements shown in 5.2.1 such as the *by* and *plot* keywords nor do you need any of the possible *<options>*; however, they can prove useful, depending on the analysis.

5.2 cont'd… options

- There are more options for each keyword and the `proc reg` statement itself.
- Besides defining the data set to be used in the `proc reg` statement, you can also use the option `simple` to provide descriptive statistics for each variable.
- For the `model` option, here are some options:

`p` prints observed, predicted and residual values

`r` prints everything above plus standard errors of the predicted and residuals, studentized residuals and Cook's D-statistic.

`clm` prints 95% confidence intervals for mean of each obs

`cli` prints 95% prediction intervals

5.2 cont'd… more options

- And yes there are more options….
- Within `proc reg` you can also plot!
- The `plot` statement allows you to create a plot that shows the predicted regression line.
- Use the variables in the model statement and some special variables created by SAS such as `p.` (predicted), `r.` (residuals), `student.` (studentized residuals), `L95.` and `U95.` (cli model option limits), and `L95M.` and `U95M.` (clm. Model option limits). *Note the (.) at the end of each variable name.

5.3 What does it do?

- Most simply, it analyzes the theoretical model proposed.
- However, it (SAS) may have done all the computational work, but it is up to you to interpret it.
- Let's look at an example to illustrate these various options in SAS.

5.4 What else can you do with it?

- Plot it (of course!) using another procedure.
- There are two procedures that can be used: `proc plot` and `proc gplot`.
- These procedures are very similar (in form) but the latter allows you to do a lot more.
- Here is the general form:

```
proc gplot data=somedata;
plot yvar*xvar;
run;
```

- Again, you **need** to identify a data set and the plot statement. The `plot` keyword works similarly to the way it works in `proc reg`.

5.4 cont'd… plot options

- Some plot options:

*yvar*xvar='char'* obs. plotted using character specified

yvar(xvar1 xavr2)* two plots appear on separate pages

yvar(xvar1 xavr2)='char1'* two plots appear on separate pages

yvar(xvar1 xavr2)='char2'* two plots appear on the sample plot distinguished by the character specification

5.5 An Example

- Let's take a look at a complete example. Consider the cigarette example.
- Suppose you want to (1)find the estimated regression line, (2) plot the estimated regression line, and (3) generate confidence intervals and prediction intervals.
- We'll look at all the key elements needed to create the SAS program in order to perform the analysis as well as interpreting the output.

5.6 Output

- Identify all the different components displayed in the SAS output and determine what they mean.
- Begin by identifying what the sources of variation are and their respective degrees of freedom.
- The last page contains your predicted, observed and residual values as well as confidence and prediction intervals.

Analysis – some questions

Now let's answer the following questions in order to understand all the output displayed.

- What do the sums of squares tell us? Or What do they account for?
- How do you determine the mean square(s)?
- How do you determine the *F*-statistics? What is it used for? What does the p-value indicate?
- What are the root mean square error, the dependent mean and the coeff var? What do they measure?

More questions….

- What is the R-square? What does it measure?
- What are the parameter estimates? What is the fitted model expression? What does this mean?
- What do the estimated standard errors tells us?
- How do you determine *t*-statistics? What are they used for? What does the p-value indicate?

Now you can….

You should be able to do the:

- Create a data set using a data step in order to:
  - manipulate a data set (in various ways)
  - use external raw data files
- Use various procedures in order to:
  - find the estimated regression line
  - plot the estimated regression line with data
  - generate confidence intervals and prediction intervals

6. Hints and Tidbits

- For assignments, summarize the output, and write the answer to the questions being asked as well as clearly interpreting and indicating where in the output the numbers came from.
- You will need to be able to do this for your tests too – so you might as well practice…..
- Practice with the examples provided in class and the practice problems suggested by Professor Gibbs.
- Before going into the lab to use SAS, read over the questions carefully and determine what needs to be done. Look over examples that have already been presented to you to give you an idea. It will save you lots of time!
- Always check the log file for any errors!

Last Comments & Contact

- I will provide you with a short supplementary document to help with the SAS language and simple programming steps (closer to the assignment time).

Anjali Mazumder

E-mail: mazumder@utstat.toronto.edu

www.utstat.toronto.edu/mazumder

References

1. Delwiche, Lora D. (1996). *The Little SAS Book: a primer*. (2nd ed.)
2. Elliott, Rebecca J. (2000). *Learning SAS in the Computer Lab*. (2nd ed.)
3. Freund, Rudolf J. and Littell, Ramon C. (2000). *SAS System for Regression*. (3rd ed.)