Overview

This page should give you a quick introduction in how to load files and look at data in SAS. It assumes you have succeeded in the arduous task of installing SAS 9.3 from the UKy Site Licensed Software page. This quick start is for the SAS 9.3 executable file, not the Enterprise Manager, or other parts (one of which actually contains a graphical interface to do much of this).

Additional resources

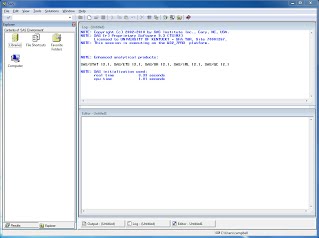
Need more help?

Check the resources, and then see Ken

Main content

**Part 1:**

The default of SAS screen should look something like this:



The main window you need to Editor window on the bottom left.  This is where you will write your SAS code (instructions or more accurately, Programs).  The log window on the top left will help you identify errors.  The Explorer window on the left will automatically switch to the Results window once you execute (Submit) your code (Program).

The concepts that I've introduced here are Commenting, basic coding, loading data files, displaying data including subsets, sorting, and summarizing data.

Except for the filepath, you should be able to copy and paste most of the examples here sequentially into the SAS program window (Editor).  Then you can either "Submit All" or highlight the new stuff and "Submit Selection".  I've included 3 example \*.csv and  \*.xlsx files at the bottom of this wiki page if you want to try any of the code.  The [Basic Linear Mixed Model code](https://sites.google.com/a/campbellmusclelab.org/wiki/protocols/statistical-analysis/Mixed-Models-Statistics/basic-linear-mixed-model-code) implements some of these procedures- along with MIXED model code.

**BASICS**

**How to Comment in SAS**

SAS code comments are either short (<1 line) comments that start and end with a slash-asterisk or start with an asterisk and end with a semicolon.

Charles has had trouble with code when SAS comments show up in comments, so watch out for errors of that sort.  Also, try to put comments only after semicolons; they don't seem to work properly if you break up sub-sections of commands.

*Example:  (this is not usable SAS code)*

/\* This is a long comment.

It can go many lines.

Comments usually show up in GREEN in SAS.

It started with a slash-asterisk and ended with an astrisk-slash

\*/

Proc Stuff;  \* Here is a short comment that ends with a semicolon. ;

  <SAS does stuff>

Run;

**All code should end in a Semicolon (;)**

Depending on the function you are typing, almost all lines should end in a semicolon.  Only if the function goes over many lines should it not end in a semicolon.

Really, if you have a problem with the code or it doesn't look right (SAS is color coded!), then you should

**Procedures need to be run.**

You will see Run; end most of the procedures and functions.

**Be aware of the placement of equals signs (=) and spaces.**

Sometimes you might expect to call a command or procedure like by=condition, but SAS actually wants you to use the statement  by condition (with a space between).

**SAS does not recognize case**

You can see in the log file, everything in a SAS is run without regard to case, and is listed in all CAPS.

**Running Code (AKA Instructions AKA Programs)**

Once the instructions (called programs in SAS) are written in the editor, you need to run them by submitting them.  This can be done by using the Run>Submit command in the menu bar, but is easier to just right-click the editor window and Submit All (or  Submit Selection if you have highlighted a subset of code).

You do not need to compile any code, but watch the Log file for maroon colored errors.

**WORKING WITH DATA**

**Loading Data Files**

There are a variety of ways to load data into SAS.  Here's a two examples, starting with the longer method and the shorter method.

*1) CSV files: The HARD way!*

Use the "DATA" step to import data.

Data allData0;

  infile "C:\Lab\charles\example1.csv";

  options spool;

  input power tag layer condition prep;

run;

Here, Data creates an a set of data called allData.  To load this data, you must first define the input file (infile) using the full file path in double quotes.  You might need to call some options.  But you also must define your columns using the input command.   The function ends with the Run command.

By the way, remember that all code should end a semicolon?  Here's an example of what happens if you want to roll over lines.

Data allData0;

  infile "C:\Lab\charles\example1.csv"

  firstobs=2 DLM='2C0D'x

  lrecl=1000 dsd missover;  \* the previous 2 lines don't end w semicolons! ;

  options spool;

  input power tag layer condition prep;

run;

Aside: ...just remember that the comments DO end in semicolons!  Also, if the comment appears one line above, an error occurs because the comment happens before a semicolon.

Important note here, that infile actually needs some modifiers. For CSV data, read the SAS code and check out: [http://www.nber.org/sys-admin/sas-csv.html](http://www.google.com/url?q=http%3A%2F%2Fwww.nber.org%2Fsys-admin%2Fsas-csv.html&sa=D&sntz=1&usg=AFQjCNGF5Pg10C1Bp_v8hP0uOEEUimxnog) (Charles accessed 2013.03.01).  The most important line is firstobs=2; this tells SAS that the first row doesn't have data, the data starts on the second row.  The DLM modifier says something about comma delimited files.  The dsd modifier is also important in that it opens missing data; and missover make sure that missing data at the end of the row does not get skipped.

One more important note, is that you need to tell SAS what is a number and what is text.  So you may have noticed in the log file there were many errors.  Adding a "$" modifier will allow it to load the column as text, not a number.  If you leave it off, it will fail to load.

input power tag $ layer $ condition $ prep $;

In the above line, tag, layer, condition and prep will come out as text and power will come out as a number.

*2) MS Excel: The Easy way!*

Most people actually don't want to reformat data into a CSV file.  You already have your data in MS Excel format so why not use it?!  Here you will use the Import procedure.

Proc Import Datafile="C:\Lab\charles\example.xlsx"

  out=allData

  DBMS=xlsx replace;  \* note the lack of semicolons until here! ;

  sheet='Sheet1';

run;

In this example, the Import procedure access a datafile (instead of infile).  All of the data is sent to an output (out) dataset, like the previous data, it is called allData.  The DBMS modifier just tells SAS that you are using a MS Excel file and the replace modifier replaces any data that already exists in allData.  The sheet modifier tells you which worksheet to load.

**So you've loaded some data.  Now what?**

Why not look at it?  Here we use the print procedure:

... \* this ellipsis isn't actually code.  Just indicates you better have ;

    \* something before this that opened some data.;

title "all data"

proc print data=alldata;

run;

You don't need a title statement, but trust me, when you start looking at the output file, you'll want them!  (If you copy-paste this code into SAS, exclude the ellipsis (. . .).)

(Oh and BTW: if you just copied and pasted the code above, you may have gotten an error.  Why?  Remember the semicolon thing?  Yeah, title should have had a semicolon before moving on to proc!)

Wait, you only want to see a few of the columns?  Well, use the var statement to just list a few.

...

title "Just some of the data";

proc print data=alldata;

  var power layer condition;

run;

Here, you would exclude tag, and prep variables from the table that is displayed.

Oh, you are more a novice than even I am?!  Well, that means some of your groups were in number form.  Lets say that you opened a CSV file, but you had layers as a numeric.  So 1=Epi, 2=Mid, 3=Endo.  Well, you can reformat data on the fly.  First you will need to use the format procedure to define the data, then use call the format in the print procedure call.

Data allData2;

  infile "C:\Lab\charles\example2.csv"

  firstobs=2 DLM='2C0D'x lrecl=1000 dsd missover;

  options spool;

  input power tag layer condition $ prep $;

run;

proc format;    \*this is how you define the new format ;

   value layerformatted 1 = "Epi"

                        2 = "Mid"

                        3 = "Endo";

run;

title "Data that shows numeric layer";

proc print data=alldata2;

run;

title "Data that does have the layer formatted";

proc print data=alldata2;

  format layer layerformatted.;

run;

The format procedure allows you to redefine the variables (value) that can be called anywhere later on.  But when you call the format in the print procedure you need to define the variable you want to replace (layer) and the varaible name you gave the new format- ending with a period (layerformatted.).

**I'm lazy and I should have removed a data set but I forgot.**

Well, you can actually exclude data by creating a new dataset.  Lets say you didn't want to use any data with tag=1.

Data allData2;

  infile "C:\Lab\charles\example2.csv"  \*this row and the next ;

  firstobs=2 DLM='2C0D'x lrecl=1000 dsd missover;

  options spool;

  input power tag layer condition $ prep $;

run;

proc format;

   value layerformatted 1 = "Epi"

                        2 = "Mid"

                        3 = "Endo";

run;

Data notAllData;

  set allData2 (where=(tag>1)) ;

run;

title "data where tag=1 is removed";

proc print data=notAllData;

  format layer layerformatted.;

run;

In the new data set (notAllData), you take the old set of allData, and only add the data if the conditional where is true (e.g. where tag>1).

Note on the where conditional of the set function of the Data procedure, you can add Boolean operators, such as:

  set allData2 (where=(tag>1) OR (tag<4)) ;

**Well, that's all fine and good, but what if I want to sort the data?**

There is a quick and easy sort procedure.

...

Proc sort data=allData;

  by condition;

run;

title "Sorted data";

proc print data=allData;

run;

This code does a sort of the Data (allData) by a variable (so data will show all HF before NF).

**Ok, but before I look at Mixed Models, can you help me summarize my data?**

The means procedure can help you quickly summarize your data.

...

title "Summary statistics";

Proc means data=allData maxdec=4;

  class layer condition;

  var power;

run;

Here, you will get the mean of the var from the allData dataset grouped by the class variables.  So in this case it will show you the average of layer, separated into each condition.  Oh, the maxdec modifier says that the maximum decimal place that you should show is 4.

Time to do work on something more complex?  Check out the [Basic Linear Mixed Model code](https://sites.google.com/a/campbellmusclelab.org/wiki/protocols/statistical-analysis/Mixed-Models-Statistics/basic-linear-mixed-model-code).