1.3: SAS Crash Course

Dr. Bean - Stat 5100

1 Why SAS?

SAS is a popular statistical software package used by many companies and researchers.

Advantages:

- Lots of output without having to write much code.
- Provides lots of graphical diagnostics automatically.

Disadvantages:

- Expensive (if you want a desktop version).
- Hard to customize output (particularly visualizations).

Teaching SAS in this class:

- (For non-math/stat majors): Has the smallest learning curve to create basic linear models.
- (For math/stat majors): Provides you exposure to multiple programming languages as several of our upper division courses use R.

In this class, we will focus on using SAS Studio, which is a free online version of SAS.

2 SAS Studio Online

- Navigate to `https://odamid.oda.sas.com/SASLogon/login?service=https%3A%2F%2Fodamid.oda.sas.com%2FSASODAControlCenter%2Fj_spring_cas_security_check`.
- Select the "Not registered or cannot sign in?" option.
- Follow directions for creating a SAS profile.
- Note that the email confirming your profile may take some time (5-10 minutes) to receive.
- Note also that the only tool we will use in this class is "SAS Studio".

3 Getting Started

Once you have an account, your SAS studio window will look something like this.

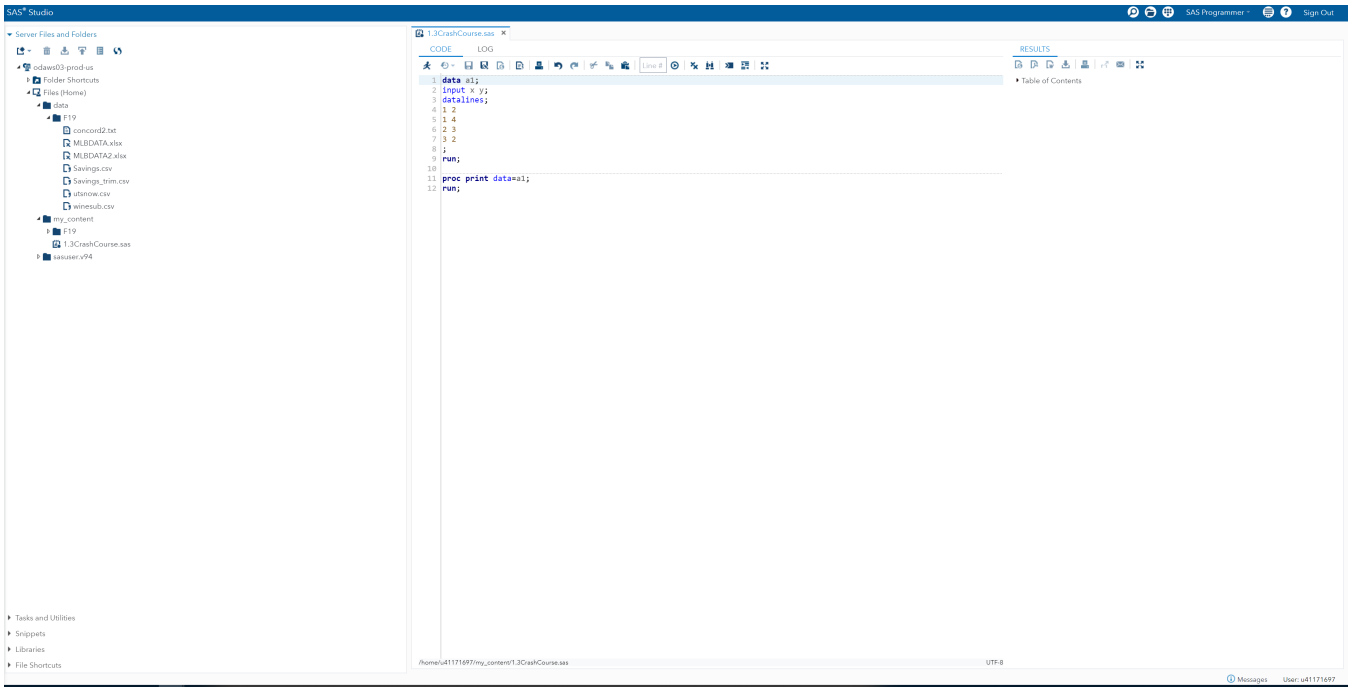


Figure 1: Sample SAS studio interface.

Main windows:

- CODE (or Editor): The window where you type in SAS commands. Font color matters in this window.
- LOG: tells what was "done". Red notes indicate errors. This is the first thing you should check if your results are unexpected.
- RESULTS (or Results Viewer): displays results in HTML format. You can export the HTML output as a pdf or a rft (rich text file - Microsoft word) to obtain nice looking output that you can copy into your reports. Or, you can use the snipping tool to save relevant output and import them as figures in LaTeX (more on LaTeX in 1.4).

You run SAS code by clicking on the "Running Man".

Necessary Components of a SAS Program:

- a semi-colon at the end of every statement
- a data statement that either creates or imports a dataset
- at least one space between each word or statement
- (almost always) a **procedure** that performs some type of analysis with your data
- a run statement

Data

There are three ways to read data into SAS. The first way is the easiest, but also not practical for large datasets.

Read in data "by hand"

```
/* Text between the asterisks are considered comments in SAS */

data a1; /* Create a new dataset named a1 */
input x y; /* List the variable names of the data that will be included in a1 */
datalines;  /* (or 'cards') tells SAS to start reading in data */
1 2
1 4
2 3
3 2
;
run; /* Tells SAS to execute the above code */

proc print data = a2; /* Print the dataset to the output screen */
run;
```

Obs	x	y
1	1	2
2	1	4
3	2	3
4	3	2

Read in data from a file.

Assume that the same data as before our located in the SAS studio folder:

`/home/u41171697/data/mydata.txt`

Note that SAS Studio will not recognize file paths on your actual computer. All data that you wish to read into SAS must be uploaded to SAS studio first.

We could read the data directly from the file using:

```
data a1;
infile '/home/u41171697/data/mydata.txt'; /* Specify path to file */
input x y;
run;
```

Read in data from a file using a procedure.

Now suppose that the data are in the excel file mydata.xlsx with the variable names included on the first row.

```
proc import
datafile =  ''/home/u41171697/data/mydata.txt''
dbms=xlsx /* Specify the file type (what separates the variables) */
out=work.a1 /* Specify the name of the dataset */
replace; /* Overwrite any datasets in the directory with the same name */
run;
```

Altering Datasets

We can add or change variables in a dataset using the commands:

```
data a2;
set a1; /* Create a copy of the dataset a1 */
  xy = x*y; /* Multiplication */
  xsq = x**2; /* Exponentiation */
  xeq1 = 0;
  if x = 1 then xeq1=1; /* Conditional Assignment */
run;

proc print data = a2;
var x y xy xeq1; /* Print all variables to the screen except xsq */
run;
```

Obs	x	y	xy	xeq1
1	1	2	2	1
2	1	4	4	1
3	2	3	6	0
4	3	2	6	0

Filtering Datasets

We can subset datasets using conditional statements.

```
data a3; set a2;
  if y < 3.5;
    /* Default 'then' is keep */
run;
/* Same as: */
data a3; set a2;
  if y >= 3.5 then delete;
run;
proc print data=a3;
 var x y xeq1;
```

```
 title1 'Smaller Set';
run;
```

Smaller Set

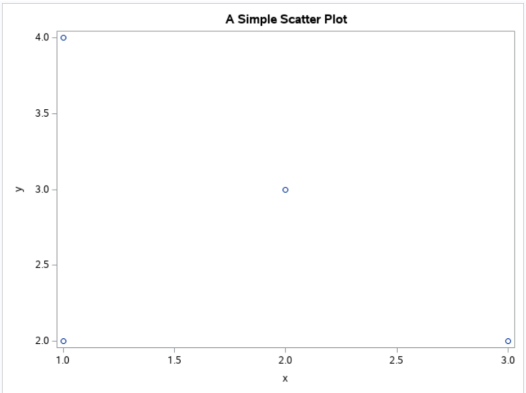
Obs	x	y	xeq1
1	1	2	1
2	2	3	0
3	3	2	0

Procedures

We prepare data in a SAS-friendly format using DATA steps. We analyze data using SAS procedures (PROC). Some PROCs that we will commonly use in this class include:

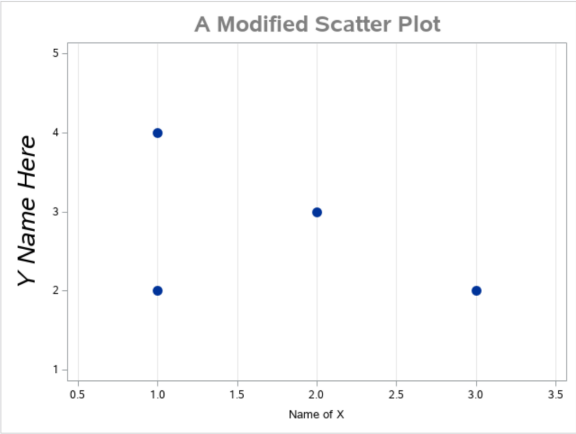
- Fitting models: PROC REG, PROC LOGISTIC, PROC ARIMA, and more
- Graphical Checks: PROC SGPLOT, PROC SGSCATTER, PROC BOXPLOT, PROC UNIVARIATE

```
proc sgplot data=a2;
  scatter x=x y=y ;
  title1 'A Simple Scatter Plot';
run;
```




```
proc sgplot data=a2;
  scatter x=x y=y /
    markerattrs=(symbol='CIRCLEFILLED'
                 size=12);
  xaxis min=.5 max=3.5 grid
    label='Name of X';
  yaxis values=(1 to 5 by 1)
    label='Y Name Here'
    labelattrs=(size=20 style=ITALIC);
```

```
  title1 height=2 color=grey
    'A Modified Scatter Plot';
run;
```




```
/ * @@ reads symbols in variable order and ignores new lines
$ indicates that variable z is a character and not numeric
. - indicates missing values */
 data a1; input x y z $ @@; cards;
   1 2 alpha   1 4 .
   2 3 gamma   3 . delta
   ;
 run;
 proc means data=a1;
   var y;
   title1 "Means Output";
 run;
 proc print data=a1;
   var y x z;
   title2 "Subtitle";
run;
```

Means Output

The MEANS Procedure

Analysis Variable : y				
N	Mean	Std Dev	Minimum	Maximum
3	3.0000000	1.0000000	2.0000000	4.0000000

Means Output
Subtitle

Obs	y	x	z
1	2	1	alpha
2	4	1	
3	3	2	gamma
4	.	3	delta

4 Miscellaneous Notes

- Missing semicolons are perhaps the most common bug in SAS code.
- The best way to get started with SAS programming is to look at the course example code, then figure out how to modify that code for your particular problem.
- Export output by copying from Results Viewer (sometimes helps to use the "Download results as RTF file") and pasting into a word document, or saving the images and importing them into a LaTeX document.
- Help in SAS: The question mark symbol in the upper right hand corner of the SAS studio editor is a good place to look for function documentation. However, reading SAS documentation can be difficult if you aren't already familiar with the procedure.
- SAS code can be written across lines. Line breaks are managed with semicolons. Data can be read in continuously with "@@".
- Missing Values: SAS procedures will completely ignore an observation if one of the variables is missing; to code a value as "missing", use the period (.) character.
- "Strings": Read in character variables with $ after the name in the input line.
- Comment Lines: To comment out a line up to the next semi-colon, put an asterisk (*) before it. To comment out an entire section, start with /* and end with */
- Selective Output: SAS will usually give you more output than you want or need, so you will need to know what you want in order to do anything useful with the output. **It is not appropriate to include ALL SAS output on homework assignments and project papers.**
- **Save Code:** SAS studio will not warn you about unsaved code when you try and close your browser. **Save your code** often to avoid frustrating loss of code.
- This class requires SAS version 9.3 or later. This is automatic for anyone using the online version of SAS studio.

5 How to "win" at SAS

- The best way to learn SAS is to **use it**. Start early on Homework 1 and 2 to start getting experience (and avoid trying to finish Homework 2 last minute, which never works for students).
- Take time to *understand* SAS code before you use it. This will make it much easier to modify the code on homework and projects.