Introduction to SAS

In this handout, we'll introduce the components of SAS and SAS coding. SAS is a statistical programming software and is a powerful tool for data exploration and analysis. Many statistical techniques are tedious to calculate by hand, and many more are impossible to compute without the aid of modern technology.

Learning SAS does not have to be a stressful endeavor! Since software is crucial to statistics, some familiarity with this widely used program is important. The main goals of the SAS exercises in recitation are to

- give you some experience in SAS programming and simple data analysis
- provide you with some basic tools that you can use in your future research
- provide resources for where to go if you need further help

Remember: this experience is solely for your benefit. You will not be tested on SAS code!

1 Parts of SAS

1.1 Windows

There are five main windows in SAS.

1. **Editor** - This is where you write, modify and/or 'run' your SAS code. The editor is a text box that you can type in. SAS does not do anything with this code until you 'run' it. Running code is when we tell SAS to actually implement our code (ie, make SAS do what we want it to do). This is the only window you can write in.
2. **Log** - This is where SAS records the code it has run for you. It tells you how long your code took to implement, and also tells you whether things worked or not. The Log will tell you if there are any errors and where they are. After running code, always check the Log for errors.
3. **Output** - This is where SAS displays the results of the code that you ran, assuming there were no errors. The Output window gives you your test statistics, p-values, and any other analysis results that you need to answer questions about your data.
4. **Explorer** - A place to view and manage the folders and files available to you in SAS. The datasets you create can be found here.
5. **Results** - List of all SAS results available in the Output window. Enables you to jump directly to the results you want, without having to scroll through the Output window to find them.

Upon opening SAS, you can see the **Editor** (bottom right), **Log** (top right) and **Explorer** (left) windows. The Output and Results windows are completely covered. To view the **Output**, we select the Output tab from the task bar located directly below the Editor. Now the Output window covers the Editor and the Log.

To view the **Results** window, click the Results tab at the bottom of the Explorer window. Now the Explorer window is covered. Click the Explorer tab to view it again.

When you open SAS, the Explorer window always shows the Work folder. The Work folder is the default temporary storage place for all new datasets you import or create. Every time you close and reopen SAS, the Work folder contents are deleted.

1.2 The Toolbar: Run, Delete and Other Icons

Directly above the Log or Output windows, you will see a row of icons or buttons; this is the Toolbar. In the middle of this row is an icon with a running man. This is the Run button; whenever we want to run a section of code that we have written in the Editor, we highlight that section and press Run. SAS will then read your highlighted code, implement it, print any errors or notes in the Log, and print the results in the Output Window (assuming no errors).

To the right of Run, there is an icon with a big X on it. This is the Delete button, which will delete your highlighted section of code. Be really careful when you press Run - don't press Delete instead! Or you will be very sad! However, if this happens, you can undo the Delete by selecting Undo from the Edit menu at the top of the screen (or by pressing Ctrl+Z).

If you hit Run without highlighting a specific section of code, then SAS will run your entire program (everything in your Editor). Likewise, clicking Delete without highlighting any code will delete your entire program, so be careful!!

TIP: *Press F3 on the keyboard instead of clicking Run. They work the same, but there's less chance of hitting Delete instead of Run.*

2 Coding (Programming) in SAS

In order to make SAS do what we want it to do, we write code in the Editor window. This code is written in a language that SAS can understand. Code provides instructions that tell SAS what we want it to do. When this code is run, SAS "reads" it and performs the tasks or procedures that we asked for. Your job is to first figure out what you want SAS to do and then to figure out how to code it properly; that is, how to write instructions telling SAS what you want.

2.1 Types of Code

There are only two forms that SAS code can take. All SAS instructions are given in one of these two forms: data steps and performing procedures.

- **Data Step** - Used to create new datasets or manipulate old datasets.
- **Procedure** - Used to implement statistical procedures in SAS.

SAS procedures will make up the majority of your code and can range from very simple to very complicated. Procedural code tells SAS what we want it to do with our data. If we want to calculate some means or standard deviations, we use the `MEANS` procedure. If we want to sort a dataset, we use the `SORT` procedure.

2.2 Code Structure

SAS code adheres to strict rules regarding structure. Code structure and the rules that govern it are collectively known as SYNTAX. Incorrect syntax will cause SAS to throw out an error and not perform the desired task, or cause it to perform the task incorrectly. Misspellings and forgotten words or symbols are all examples of syntax errors.

To help you with syntax, SAS has some built-in code features. For instance, some words in your code will automatically change color after you type them. These are called SAS Keywords, and they can be either dark blue boldface or light blue. In recitation handouts, SAS Keywords will be written in capital letters, in order to highlight them for those who may be using black and white printers. SAS Keywords have a special meaning in SAS.

Take a look at the code below in Figure 1. Don't worry about what the code might be doing. Rather, focus on the structure of the code.

```
PROC MEANS DATA=your_data ;
   CLASS gender ;
   VAR bmi ;
RUN ;
```

Figure 1: Simple example of SAS code.

2.3 Code Statements

Each line is called a STATEMENT. The type of statement is indicated by the first word in each line. For instance, the first line is the `PROC` statement and the third line is the `VAR` statement. The first word in each statement of a SAS Procedure will always be a SAS keyword (see §2.5).

2.4 Semi-Colon (;)

Another VERY IMPORTANT part of the above code is the use of the semi-colon (;) at the end of each statement or line. In the SAS language, the semi-colon ends a statement, just like a period (.) ends a sentence in English. When SAS is reading your code, the semi-colon tells SAS, "This statement is over. Start a new statement with the next line."

If you forget the semi-colon at the end of a statement, then SAS doesn't know to separate statements. For instance, if you accidentally drop the semi-colon after the word `gender`, then SAS will *incorrectly* interpret your code as shown in Figure 2.

```
PROC MEANS DATA=your_data ;
   CLASS gender var bmi ;
RUN ;
```

Figure 2: SAS code from Figure 1 with an error (left out `;` between `gender` and `var`).

> **TIP:** *Always start a new line of code after a semi-colon. This will help you remember to include them!*

2.5 SAS Keywords

Another helpful feature of SAS is the color coding of certain words. These are called SAS Keywords, and they can be either dark blue boldface or light blue. In recitation handouts, SAS Keywords will be written in capital letters, in order to highlight them for those who may be using black and white printers.

> **TIP:** *If the first word of every statement in a SAS Procedure is not a color-coded SAS Keyword, then you probably forgot a semi-colon!*

SAS Keywords have a special meaning in SAS. Let's look more closely at some of the SAS Keywords found above.

`PROC` – is short for PROCEDURE. SAS analyses will *ALWAYS* begin with this word! It tells SAS, "I'm initiating a statistical procedure". The Keyword after `PROC` always tells SAS exactly which procedure I want.

`MEANS` – describes the procedure for SAS to use. We have initiated a procedure with `PROC` and now we are telling it to use the `MEANS` procedure.

`RUN` – ends the procedure. SAS analyses will *ALWAYS* end with this word! Every procedure (`PROC`) must be closed with a `RUN` statement.

This code demonstrates the Procedure type discussed in §2.1. These three Keywords are very important for procedural coding! Here are the other SAS Keywords from Figure 1.

`DATA` – tells SAS the dataset to which the current procedure should be applied.
`CLASS` – begins a statement specific to `PROC MEANS`. More on this later.
`VAR` – begins a statement specific to `PROC MEANS`. More on this later.

3 Formatting

3.1 Appearance

Look at Figure 1 on page 3. Notice that the middle two code statements (starting with light blue SAS keywords) are indented while the opening (first) and closing (last) statements are not indented. This is the format we usually apply to SAS code to make it more readable to others who may look at your code later. Note that the format emphasizes

- *beginning* a procedure with the opening line which starts with `PROC`
- giving further SAS instructions *inside* the procedure with indented statements
- *ending* or *closing* the procedure with `RUN;`

3.2 Commenting

Commenting is another important feature of SAS code. SAS Comments are statements in your code that are ignored by SAS - SAS reads them and prints them in the log, but does not act on them. In other words, comments are instructions for YOU, not for SAS. SAS comments will always appear in green font in your Editor window.

There are several reasons why we might include comments in our code. For instance, you may want to

1. Remind yourself why you are using certain pieces of code
2. Help others (like your boss!) review and understand your code more easily.
3. ‘Comment out’ big chunks of code that you don’t need, but are not quite ready to delete yet.

You can put anything you want in a SAS comment. There are two ways to insert a SAS comment.

3.2.1 Method 1 - Asterisk (*)

This method is used to create comments one line at a time. Starting a statement with an asterisk (*) will initiate SAS comment mode. Everything after the asterisk (*) will be a comment, and therefore ignored by SAS when you run code. Figure 3 (page 6) demonstrates the inclusion of comments in SAS code.

```
*First I sort my dataset by the variable gender ;
PROC SORT DATA=your_data ;
   BY gender ;
RUN ;

*Next I compute descriptive statistics on BMI by gender ;
PROC MEANS DATA=your_data ;
   CLASS gender ;    *to get stats for Males and Females separately ;
   VAR bmi ;
RUN ;
```

Figure 3: Simple example of SAS code with comments explaining what code is doing.

SAS comments like these **MUST be ended with a semi-colon (;)**, just like any other statement!!! Everything between the asterisk (*) and the next semi-colon (;) will be considered a comment by SAS. If you forget to end the comment with a semi-colon (;), then SAS continues in comment mode, as shown in Figure 4.

```
*First I sort my dataset by the variable gender
PROC SORT DATA=your_data ;
   BY gender ;
RUN ;

*Next I compute descriptive statistics on BMI by gender ;
PROC MEANS DATA=your_data ;
   CLASS gender ;    *to get stats for Males and Females separately ;
   VAR bmi ;
RUN ;
```

Figure 4: SAS code from Figure 3 with forgotten semi-colon (;) at the end of the first line.

Clearly, the first line of the `SORT` procedure was incorrectly treated as a comment, and so SAS never knew to sort the data. SAS will throw errors and none of the code from Figure 4 will run, all because of a forgotten semi-colon (;).

TIP: *Before running code, double-check that there are semi-colons ending every line.*

3.2.2 Method 2 - Using /* ... */

We use this method to comment out large sections of code or text at once. Let's say there are 150 lines of code that we want SAS to ignore. Rather than going through and typing *at the beginning of all 150 lines, we can simply insert `/*` above line 1 and `*/` below line 150. Figure 5 demonstrates this.

```
/*
In this program I will compute simple descriptive statistics on BMI from
the dataset your_data.  First, I will sort the dataset by gender using PROC
SORT. Then I will use PROC MEANS to compute descriptive statistics on BMI
for females as well as males all in one procedure.
*/

*First I sort my dataset by the variable gender ;
PROC SORT DATA=your_data ;
   BY gender ;
RUN ;

*Next I compute descriptive statistics on BMI by gender ;
PROC MEANS DATA=your_data ;
   CLASS gender ;    *to get stats for Males and Females separately ;
   VAR bmi ;
RUN ;
```

Figure 5: SAS code from Figure 3 including both types of comments.

4 Questions

1. What are the three windows in SAS? Which ones can you write in?
2. What are SAS keywords? What color(s) do they appear?
3. What are the two types of SAS code? Which type is used most often for statistical analyses?
4. What is the main difference between the two methods of SAS commenting? What color are SAS comments?
5. What is the most important symbol used in SAS code? (Hint: It gets forgotten a lot!)
6. Why is code format important? Which lines should be indented in a SAS Procedure?
7. How do you run code after it is written?