171:161: Introduction to Biostatistics
Breheny

Lab #1

The goal of lab #1 is to introduce SAS and R as well as establish that you can do the most basic of tasks in each: read in a data set and look at it.

Both R and SAS are perfectly good software packages for analyzing data. Their styles are different, but underneath it all, they do the same thing and give you the same results. Which one you use is up to you – go with whatever one you feel most comfortable with.

One clear advantage of R, however, is in performing simple calculations – this is much easier in R than SAS. So, even if you prefer to learn and use SAS for data analysis, you should still know how to use R as a calculator. Again, however, feel free to use any statistical software however you see fit. They are all tools; use whichever tool works best for the job at hand.

1 Software overviews

SAS

When you open SAS, you will notice that three windows appear by default: the "Editor" (where you type your commands), the "Log" (which is a log of everything SAS does; most of the time this information is not particularly interesting, but if there are any errors, they will show up here), and "Output", which is a relic from earlier versions of SAS and you will probably never use. Once you start analyzing data, a fourth window will appear, the "Results Viewer". SAS used to output its analyses to the Output window, but now uses the Results Viewer.

There are two types of SAS commands: `PROC` steps and `DATA` steps. `DATA` steps are for manipulating data, `PROC` steps are for performing analytical procedures on data. In this class, the data will come to you already formatted (for the most part), and we will not spend a great deal of time on `DATA` steps. However, be aware that SAS offers extensive tools for the formatting of data.

NOTE: Computers do *exactly* what you tell them to do, so any failure to, say, include a semicolon, may result in completely useless code. So, it is important to type the commands exactly as we have written them. With one exception: SAS commands are not case-sensitive, so you do not have to capitalize `PROC`, `CONTENTS`, `DATA`, or `RUN`. There is a widely-used SAS convention of capitalizing SAS *keywords* that you may choose to implement.

Note that the SAS Editor helps make your code more readable in a number of ways. It represents keywords in special colors, represents `PROC` steps bold. This is quite valuable. For example, if you type what you think is a keyword and it doesn't turn blue, this is an early indication that your program is not going to work, and that you should double-check everything.

Color	Explanation
Black	General part of code
Bold Dark Blue	Procedure Names
Bold Green	Number
Light Blue	Statements and options in procedure
Green	Comment
Purple	Quote
Yellow highlighted	Data input when using DATALINES

You will benefit tremendously from saving everything you write so that you can access it again later. You can save what's in the Editor window by pressing **Ctrl+S**, clicking on the disk icon, or going to **File** → **Save**. You can open it again later by going to **File** → **Open Program**.

R

RStudio is free, open-source software, so if you would like to install it on a different computer, you can download it from `www.rstudio.com/ide/download`.

Unlike in SAS, a semi-colon is not required (but can be used) to end each line of code and capitalization does matter. For example, the names `lab`, `Lab`, and `LAB` refer to three different objects. Green text in the code also indicates a comment in R as in SAS.

2 Reading in data

In real life, data comes in all kinds of different formats. For this class, all data will be made available as tab-delimited text files. Download one of the course data sets and save it somewhere.

SAS

There are several ways to read data into SAS, but the easiest is through the "Import Wizard". The steps:

1. Go to **File** → **Import Data**.
2. For the data source, select "Tab Delimited File (*.txt)". Click **Next**.
3. Browse to wherever your data is located. Click **Next**.
4. SAS calls its working directory "WORK", and calls the data sets in that directory "Members". In the **Member:** field, type whatever you want to call the data set (*i.e.*, if you downloaded the Titanic data set, you may want to call it "titanic"). Click **Finish**.

An alternative method is to write the code yourself using the following syntax:

```
proc import datafile = "M:\General Research Directory\171-161\Data\malaria.txt"
    out = tempdata
        /* name of the file within the temporary SAS directory */
    dbms = tab
        /* tells SAS the file is tab-delimited */
    replace
        /* if we were to alter this text file and read it in again,
it would replace the old SAS version */;
    getnames = yes;
        /* tells SAS whether or not to use the first row as variable names */
run;
```

Suppose we knew we would be working with this dataset on multiple occasions; it might be helpful to already have the dataset loaded after the first time. To accomplish this, we use the `LIBNAME` statement to create a permanent SAS dataset.

```
libname lab "M:\General Research Directory\171-161\Data";
proc import datafile = "M:\General Research Directory\171-161\Data\malaria.txt"
    out = lab.tempdata
        /* the dataset is now saved permanently in the directory referenced by lab */
    dbms = tab
    replace;
    getnames = yes;
run;
```

If we have a small dataset, we can type the data in by hand using the `DATALINES` statement.

```
data temp;
input x y $;
/* the $ indicates y is a character variable */
datalines;
2 a
5 y
8 k
;
run;
```

R

Files can be read into R directly from the internet. Sometimes, we would like to save this dataset somewhere locally and use that file. For instance, if a dataset is updated periodically but the name remains unchanged, we might wish to distinguish between older and newer versions used in analyses. This can be accomplished by saving the dataset and naming it accordingly. The same file can be read into R using the following code:

```
malaria <- read.delim("http://myweb.uiowa.edu/pbreheny/161/data/malaria.txt")
#malaria <- read.delim(file="M:\\General Research Directory\\171-161\\Data\\malaria.txt")
#malaria <- read.delim(file = "M:/General Research Directory/171-161/Data/malaria.txt",
header = T)
```

The "header" option is similar to the "getnames" option — both allow the user to specify whether the first row contains variable names or not. In R, the default option for "header" is true; unless your data does not contain variable names, including this option is not necessary.

3 Looking at your imported data

SAS

There are many ways of looking at your imported data, but the most helpful in our opinion is the "Table Editor".

1. Go to **Tools** → **Table Editor**.
2. Go to **File** → **Open**. NOTE: The context of SAS's menus change depending on which window is highlighted. For **File** → **Open** to work, the Table Editor window needs to be highlighted (it should be, unless you clicked somewhere else on the screen in between the first two steps).
3. Browse to your data set and open it. Recall that when we imported it, we saved our file in the "Work" directory and gave it a name.

Your data should now be visible in a spreadsheet. You can see all the data as well as the names of the columns.

Another option is to use the Explorer Window.

1. Click on Libraries.
2. Click on the name of the library where your data is saved — Work or Lab.
3. Double click on the named dataset.

NOTE: Be sure to close the tables before trying to use the data in a SAS command.

R

There are several very similar methods in which we can print or display our datasets.

To print the dataset to the console we can use the print command when sourcing our code, use parentheses around the object we wish to print when running the code line by line, or type the name of the object into the console.

```
> print(malaria)
> (malaria)
> malaria
```

To view the dataset we can use the view command or double click on the object in the upper-right box of the RStudio window.

```
> View(malaria)
```

NOTE: the "V" on View is capitalized.

4 Using SAS off campus

SAS is not a free-software program and requires a license. To access SAS from off campus, you can use virtual desktop. More information can be found at `http://helpdesk.its.uiowa.edu/virtualdesktop`.