

What We Learned From Lab 4

- First, I apologize to the many students who called me and emailed me about not being able to find the file `student.R` for lab 3. I had not put it on the website (forgot to) and thought (wrongly) that I had announced in class that one needed to type the script in NOTEPAD. Also, some concerned students commented that there had to be a special “format” for the scripts. Let me state that you need only to write scripts in NOTEPAD, save them in the `.txt` NOTEPAD format. The only special thing you should do is in the title of the script, put a `.R` in the title block after you give it a title. Open up some of my posted scripts in NOTEPAD to see what I mean. By putting the `.R` in the title of the `.txt` file, you make it a little easier for R to find the script it is looking for. Again, sorry for the imposed frustration on that part of the lab. You have enough legitimate things to get frustrated with R, etc., and don’t need me to add to the mix!!
- We saw how I generally approach R tasks. I write little scripts for each major part of R, in the script window of R Gui, then execute these little scripts until I get the bugs out of them. Then I save them, once they have been “field tested” in the R Console window successfully. I also use (probably too much) comments throughout the scripts, which are noted by a preceding `#` sign and terminated with a carriage return in NOTEPAD
- We saw that variable names in R can be of any length, as long as they start out with a letter and don’t contain “sensitive” characters used as library commands in R. They can have periods in the names and are *case sensitive* (so watch for that when troubleshooting programs). You just need to be careful not to use preassigned names given to R library functions. This last point is sometimes very hard to know, since there are many preassigned functions I don’t know about in R. So, I use this basic rule of thumb when doing R programs. I try to make my variable names long enough to be descriptive about what they represent, yet short enough to not get weary typing them multiple times in a program. Also, if I suspect a name I want might be a predesigned function name, like `sd` (for standard deviation), I always end the name with a number, like `sd1`, because, as far as I know, no R commands have numbers at the end of them.
- We now see that, when possible, always have your data files written in `.txt` NOTEPAD format, and downloaded to the root directory R goes to to find data files. It is probably prudent to have these `.txt` files in a tab delimited or space delimited format, rather than comma delimited, to avoid our EXCEL.csv problems we had at the beginning of the course.
- We saw that usually we read in data in table format, using the `read.table()` command. If we don’t have titles on our table columns, R will give it generic names `V1`, `V2`, etc.
- We saw that when you have to make some graphical presentations, you have to make the table into a matrix or vector, in order to avoid the error message saying that the format of the data cannot be made into the graph requested.
- We saw that much troubleshooting of programs involve gleaning small syntax or capitalization letter miscues we type. This can be and is frustrating, yet, cannot seemingly be avoided.

- It's always a good idea, after R has indicated that it has read in a data set or read an extended length variable, that you have R print out what it has, to verify that R has what you want it to have. This avoids mistakes later, when you think a table has 3 rows when it only has 2, etc.
- R has a few “nice” functions, like `addmargins()`. This function “automatically” gives row and column totals to 2 way tables.
- Sometimes you have to print out a graph that has a smaller scale than you would like, because you also have to print out a legend with the graph. In cases like this is is sometimes good to also give another graph of the same information, with a closer scale and no legend, so you can see subtle details better.
- Some arguments are logical `TRUE` and `FALSE`, like the `legend.text=` command. Remember to use all upper case letters for the true and false, and remember that you can abbreviate these with, respectively, `T` and `F`, instead of writing out the whole word.
- We see that the left arrow (`<-`) is sort of like the “assignment equal to” symbol. The equivalent is the `:=` sign in C+, and stands for assignment, rather than “equal to”. For instance, if we have the command `x1 <- vec1`, we take vector `vec1` and put it in R's memory location `x1`. We save the `=` sign for function argument delimiters and for logical expressions, rather than for the assignment symbol.