**MATH 3338: Mathematical Modeling (Fall 2017)**

*Instructor*: Dr. Hoa Nguyen

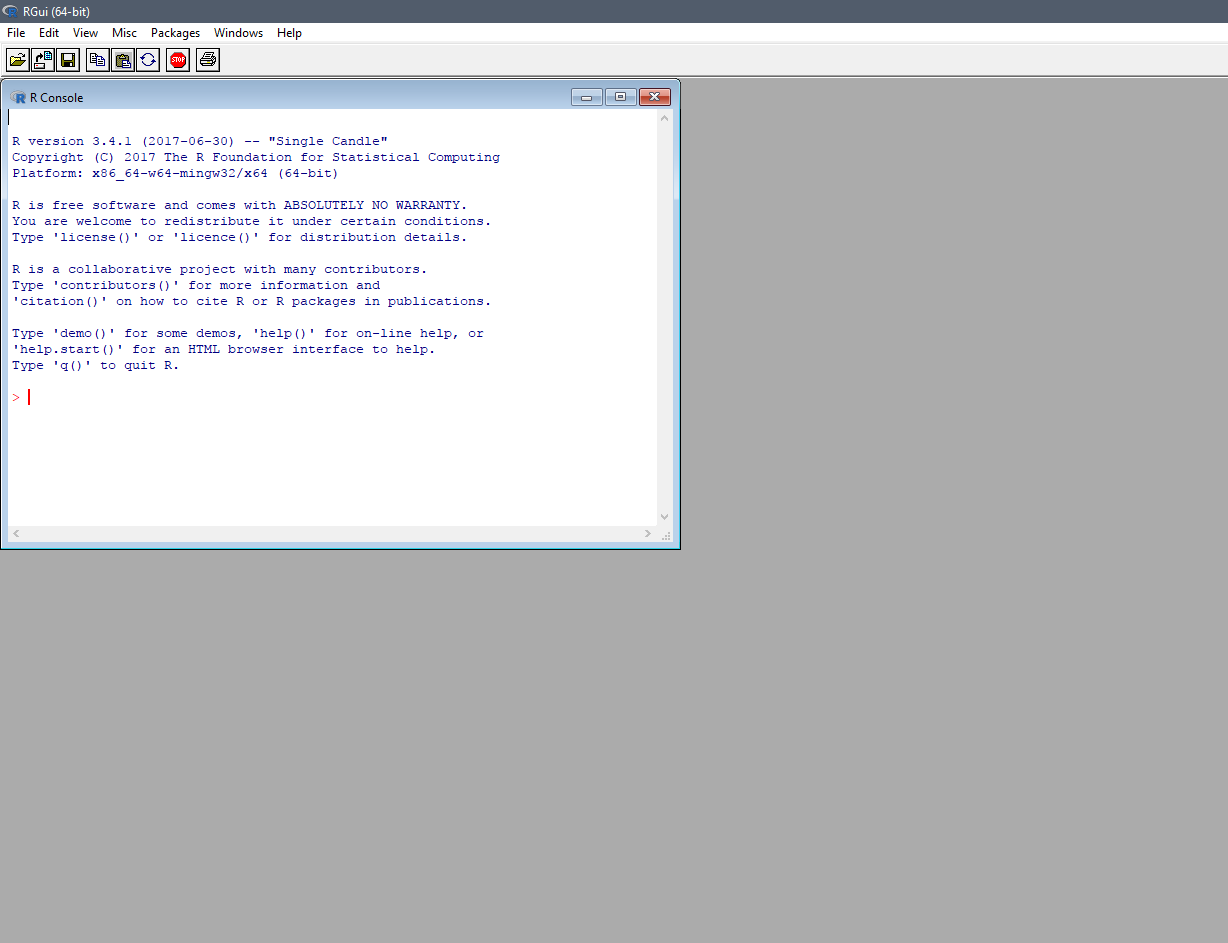
*Project Advisor*: Courtney Rohde, ACAS

**R Demo Instructions**

We will build our models in R. To download R (for Windows or OS X), go to

<http://lib.stat.cmu.edu/R/CRAN/>, select your operating system, download and install the base system (“Binaries for base distribution) R 3.4.1.

When you open R in Windows, it will look like this:



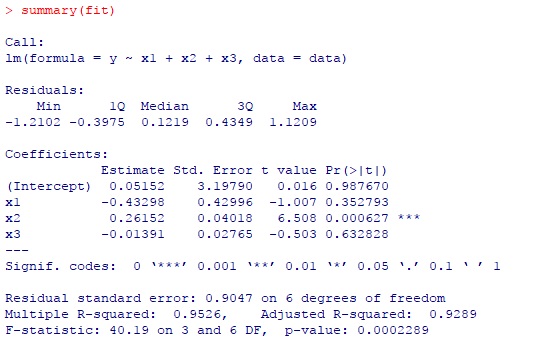
Download the demo data I provided (demo.csv) and put it in a project folder on your computer (you should make a project folder).

With this demo, we want to run a multiple regression. Once the regression is run, we want to view a summary of its coefficients and the create essential plots. To do so, you will need to use the following functions:

1. read.csv()
   1. This allows R to see the data and gives the data a name. You type in a line of code like this:
      1. name <- read.csv(“file path”)
      2. There is an optional second argument you can enter if the first row of your data is a header row (which for us, it is), but I couldn’t tell what difference it makes. The model doesn’t change. But if you want, you could use this code: name <- read.csv(“file path”, header=TRUE)
      3. In place of “name,” you can put whatever you want to call your data. In fact, “data” is a great option. The arrow “<-” is assigning the data to that name.
      4. If you’re curious, both Excel and Notepad can open .csv files (short for comma separated value. maybe that’s why they look different?) I encourage you to take a quick look at what’s actually in the data file if you want. But don’t save over it – it needs to stay saved as a .csv, not a .xlsx or .txt or whatever.
   2. In R, file paths look a bit different than what you might be used to. Here's how you specify a file path in R:
      1. Windows - C:\\Users\\Courtney\\Documents\\Project\\demo.csv
      2. OS X - /users/Courtney/documents/project/demo.csv
   3. If you type in your line correctly, R will just accept it and give you another command prompt, denoted by “>” like this:

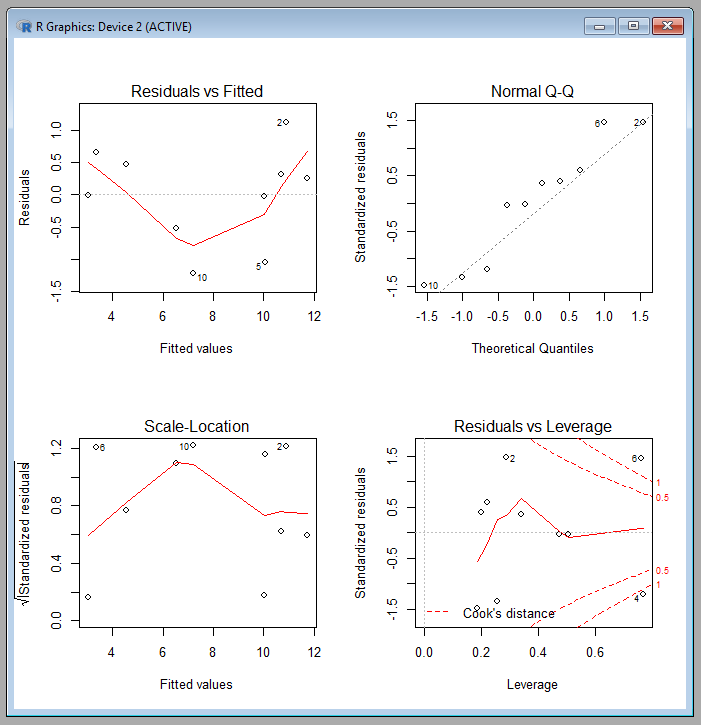


1. lm()
   1. This is the linear model function. It will fit a nice straight line to the data you had R read in the previous step. The linear model also needs a name, and it has two arguments. The first argument specifies the model form, and the second specifies the data. So that we all get the same answers to compare, I’ll give this one to you:
      1. fit <- lm(y ~ x1 + x2 + x3, data)
      2. Here, “fit” is the name we are giving the model, we are using a simple additive linear formula with all three x variables predicting y, and “data” is the name we gave the data we imported in the previous step.
   2. Again, if you’ve done it properly, you should just get a new command prompt and no errors.
2. summary()
   1. Were you wondering what actually happened just then, and why R didn’t show you anything? It’s a very polite software – it will only show you what you ask to see. The summary function will give you details about pretty much anything that has a name.
   2. Now if you want to know what intercept and slope parameters R estimated given our three x variables (aka “predictor” variables), just use your model name as the argument of the summary function. For the model form I specified above, the results look like this:



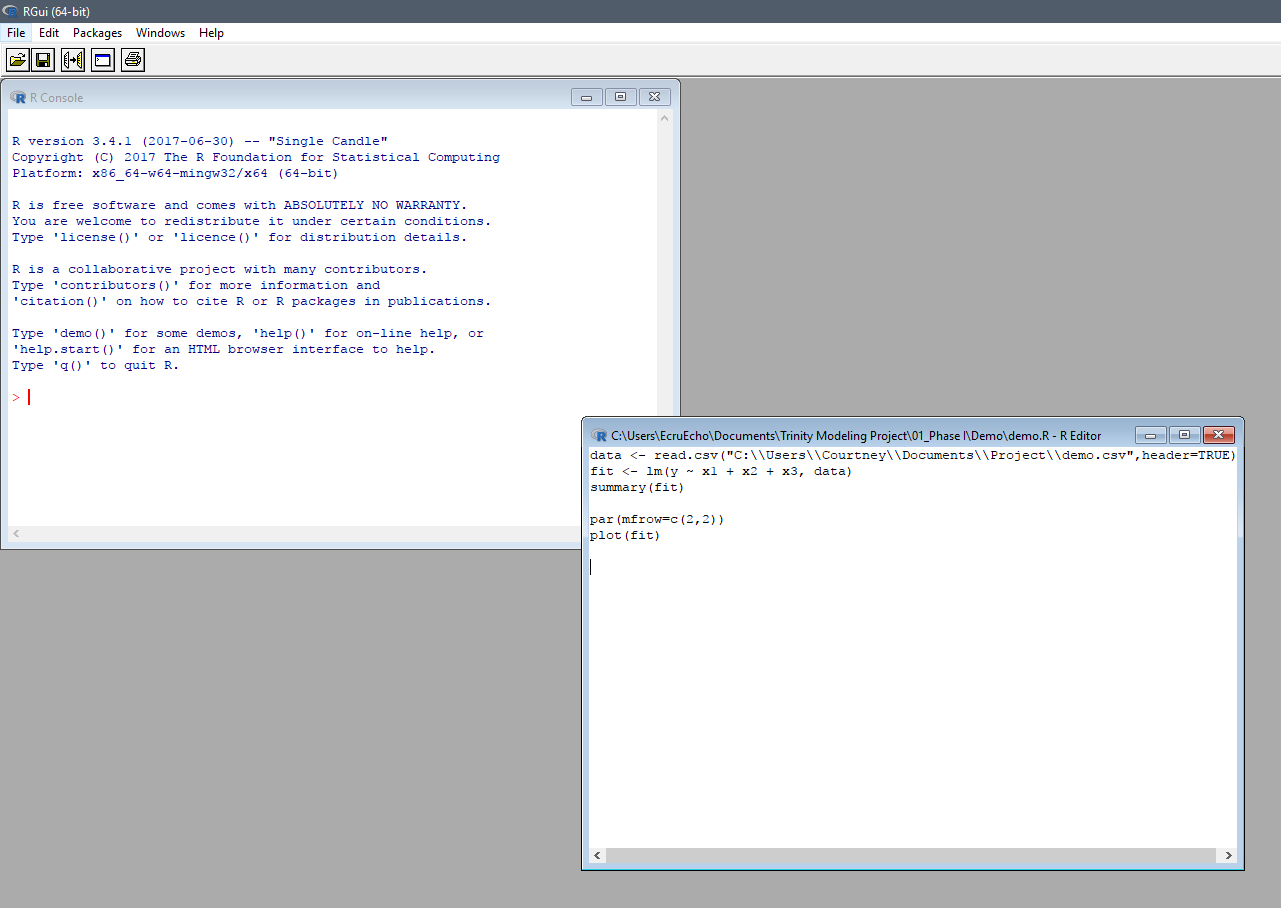
* 1. The parameter estimates are in the “Estimate” column under “Coefficients.” The summary gives other useful statistics like the standard error and the p-value in the last column, which we will learn to use later to improve our model.
  2. If you were curious, you could run the lm() function again (maybe with a different name this time?) editing the formula to drop a variable. Does that change the parameter estimates?
  3. Also if you were curious, you could ask R for a summary of your data. That has a name too.

1. par(mfrow=c(2,2)) – This is used to display 4 plots on a 2 x 2 grid
2. plot()
   1. The par() function will open a new window in R, but nothing will display until you’ve also used the plot() function. The argument of the plot function is the name of the model you want to plot. For the linear model we used above, the output should look like this:



Sometimes it can get annoying typing in all this code over and over! This is where scripts can come in handy. I want you to go through the rest of the demo first, to learn the code line by line. But at the end, you can open and run the R script file I’m providing, which will do all five commands in a row! Put the demo.R file in your project folder.

From the R main interface, you can either click the first button (with the tooltip “open script”), or go to File > Open Script… and navigate to demo.R. When it opens, it will look like this:



You will just need to update the file path to point to the demo.csv file on your own computer. Then you can highlight all of the text in the script, and click the third button (with the tooltip “Run line or selection”). You will see R execute each line in succession, automatically giving you the summary and plots as well.

Congrats, you now know how to run a model in R!