Lab 1 - Introduction to R, RStudio, and RMarkdown

Lab Goals

A gentle introduction to R, RStudio, and RMarkdown

Introduction to R and R Studio

The goal of this lab is to introduce you to R and RStudio, which you'll be using throughout the course both to learn the statistical concepts discussed in the texbook and also to analyze real data and come to informed conclusions. To straighten out which is which:

- **R** is the name of the programming language itself.
- **RStudio** is a convenient environment for writing and running R code.

As the course progresses, you are encouraged to explore beyond what the labs and homework dictate; a willingness to experiment will make you a much better programmer. Before we get to that stage, however, you need to build some basic fluency in R. Today we begin with the fundamental building blocks of R and RStudio: the interface and basic commands.

Files, Plots, Packages, Help

The tabs included in the lower right hand corner of RStudio contain the following information:

1. *Files* - Through this tab you can navigate and open files stored on your computer. You can also save files here and create new folders.
2. *Plots* - When you create plots through commands typed into the console, your plots will show up here. We will see an example of this below.
3. *Packages* - This is a list of packages that you currently have loaded in R.
4. *Help* - For any function you have access to in the console, typing `?functionname` in the console will bring up the documentation for that function under this tab. We will also see an example of this below.

The Workspace

There are two useful sources of information in the panel in the upper right of RStudio:

1. Your console's *workspace* under the **Environment** tab. The *workspace* consists of all variables you have defined in the *console*. R will remember anything you have defined in the console unless you clear your workspace. Before you start any new project you should start with a clear workspace.
2. The **History** tab includes a history of the commands that you've previously entered into the *console*.

The Console

The panel on the left is where the action happens. It's called the *console*. Everytime you launch RStudio, it will have the same text at the top of the console telling you the version of R that you're running. Below that information is the *prompt*, >. As its name suggests, this prompt is really a request, a request for a command. You type an R command at the prompt and then you hit *return* (or *enter*) for it to be carried out. Initially, interacting with R is all about typing commands and interpreting the output. These commands and their syntax have evolved over decades and now provide what many users feel is a fairly natural way to access data and organize, describe, and invoke statistical computations.

R can do a lot of things, but for now let's use R as a fancy calculator.

1. Type `3/2` in the console (followed by hitting return). What is the output?
2. Type `a` into the console. What happens?
3. Type `a <- 1`. And now type `a`. What just happened? (hint: `<-` is called the assignment operator)
4. Type `w <- 1:10`. And now type `w`. What does `:` do?
5. Type `z <- w^2`. And now type `z`. What does `^2` do?
6. Suppose we want `v` to be a numerical vector of length 10 that starts at 3 and ends at 12. Describe two ways to form this (hint: look back at 4 and 5)
7. Type `plot(x = w, y = z)` into the console. Can you describe what `plot` is doing? It is a *function*, meaning it is a piece of pre-defined code that takes some inputs, does something with them, and then produces an output and/or takes some action.
8. Type `z[5] <- 100`. Type `z`.
9. Type `plot(x = w, y = z)` into the console.
10. Suppose we want to add a title. Type `?plot` in the console.
11. Add the text "My Favorite Plot" as a title to the plot. Hint: the inputs of an R function are called its arguments. Look at the help page for `plot` to find an argument that you can add to `plot(x = w, y = z)` so that a title will appear.

Reproducibility

The *Export* button near the plot lets you save this plot as an image file or pdf. However, this is not a good practice for research. If we save this as an image and close R, we risk forgetting how we created this plot! If later on we want to make a small change (e.g., change the title), we would have to start over.

There's another important reason for saving code and not just the results. When we do analyses (or any coding for that matter), it is important that somebody else can **reproduce exactly what was done**. This "somebody" might just be you six months later trying (desperately) to reconstruct what you did. Or it could be a curious reader wanting to understand exactly what analysis steps you took. There is a growing trend among journals to require authors to include their code with published papers. When you do not provide code, you are asking readers to put a lot of trust in you and the choices you made.

12. Create a new R script ("File > New File > R Script") and put all the lines of code needed to generate the plot in the file (you can copy and paste from the console). Save this file as `my_plot.R` in a location where you can find it (for this lab, you might put it on the Desktop).
13. Type `q()` in the console to close RStudio. Do not save the workspace (i.e., type `n` when prompted).
14. Reopen RStudio and find and open `my_plot.R`. (Find it by navigating to appropriate folder in the *Files* tab. Then click the link to open it.)
15. Press the "Run" button in RStudio. What does this do? (Repeatedly press it until the plot is created.) Congratulations, you've written an R script!

R Markdown

Writing R scripts is a big step to doing reproducible analyses, but we can still do better. Scripts are just raw code telling the computer what to do. Analyses can be complicated, requiring judgment calls and other steps that require explanation. R Markdown is a great tool for **writing code and communicating it well to**

humans at the same time. We will use R Markdown for everything in 6010 so that by the end of the course it should be second nature for you.

16. Create a new R Markdown file in RStudio ("File > New File > R Markdown. . . " and press ok to create the file). Save this file as `my_report.Rmd` in a place where you can find it (e.g., the Desktop).

17. By default, RStudio has an example that demonstrates some of the features of using R Markdown. Notice that this is a combination of (a) a header, (b) human readable text, and (c) R code that appears in "code chunks" (gray areas). Press "Knit HTML" to see the file that is created when you "knit" the file `my_report.Rmd`.

18. (If time remains:) Replace the default example in `my_report.Rmd` with the code you've written in which you define `w` and `z`, and then create a plot with a title. Try breaking these into different code chunks in which you have text explaining what you are doing between the code chunks.