Module 2 Notes

Justin Millar

2018-09-06

Download notes as a PDF

Using the Working Directory

As Dr. Baiser discussed in the lecture, R is **not** a point-and-click program, and this is a *good thing*. However, this means that using local files (i.e., on our computer) in R is different than it is in programs like Microsoft Excel. Rather than click through folders in a window, we have to specify the path to where the file is located on our computer.

Finding the absolute path

To find the absolute path use the File Explorer in Windows (or Finder in Mac) to navigate to the file you wish to read into R. Right-click on the file and select `Properties`. Under `General` you should find a field called `Location:`. The absolute path is the string in the location and the file name.

Setting the working directory

You can use the `setwd()` function to set the working directory. Think of this as telling R that you'll always be starting from this location. Once you do this you'll only need to specific the path based on the working directory.

Use Auto-complete with the `Tab` button

R Studio has a useful auto-complete function for specifying paths. When you press the `Tab` button when the cursor is in between quotation marks you will get a popup menu for navigating the paths to different files/folders. The auto-complete will start in the current working directory. If you haven't defined this yet, then it will be working directory defined in the global settings. You can always check the working directory by running `getwd()` in the Console.

I will demo this feature in my office hours.

Using Projects

One issue that can arise from setting working directories is that when the file paths change our scripts will have to be updated, otherwise they won't work. This is why you have to change the `setwd()` in the Data Exercises to match the file paths on your specific computers.

R Studio has a really nice method for dealing with this limitation called Projects. You can make a new Project folder by selecting `File >>> New Project >>> New Directory`. This will create a new folder on your computer, and inside the folder there will be a `.Rproj` file. This file tells are that all the paths to files in this Project start in this folder. This means that we don't ever use `setwd()`, the working directory is already set to be project folder. And this also means that if we move the Project folder, say from the Desktop to Documents, or even send it to another computer, we don't have to change the file paths.

You don't have to see Projects in course, but I highly recommend trying them out. I'll demo Projects in my office hours, and you can check out more info at https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects.

Differences between brackets [ ], parentheses ( ), and quotations " "

Parenthesis `( )`: used by functions:

```r
seq(1, 10, 2)
c("cat", "dog", "bird")
# Note that `c()` is a function, which stands for
# concatenate, used to grouping data into a vector
```

Brackets `[ ]` are used to index or subset parts of dataframes or vectors:

```r
# Subsetting vectors
rivers[1]
rivers[1:5]

# Subsetting dataframes, indexed as [row,column]
mtcars[1, ]
mtcars[, 1]
mtcars[1, 1]
mtcars[, "mpg"]
```

Quotation marks `" "` or `' '` are used to define character strings. Single and double quotes act the same in R, however they may be cases where you need to use quotes inside of quotes so it's best to pick one you use most often and stick with it.

Installing and loading packages

Using packages in R is a two-step process:

1. Install the package to your computer:

```r
install.packages("package")
```

2. Load the package to your current R session with `library()`:

```r
library(package)
```

You only need to install a package once on your computer to use it, but you do have to load it each time you want to use it in a new session. The `install.packages()` functions takes in character strings, which is why we need `""`, but the `library()` functions does not. This also means that we and install multiple packages at once using `c()`, but we have to load them individually.

```r
# This works
install.packages(c("package1", "package2"))

# This does not work
library(c(package1, package2))

# But this does
library(package1)
library(package2)
```

Finally, sometimes you'll load two packages that have functions with the same name. To specify which function you want, you use package name followed by `::` and the function name:

```r
# package1 and package2 both a function called coolFunc() To
# specify which version I want:
package1::coolFunc()
package2::coolFunc()
```

Changing the Settings in R Studio

You can access the setting by clicking `Tools >>> Global Options...`. R Studio is constantly being updated, so there be some small differences between your options interface and the lecture video.

You may want to change your CRAN mirror to the nearest geographic location, however this setting is unlikely to significantly change your experience so don't worry if you're not using the same one as shown in the lecture.

Code editing

This are most a user-preference. The one that I strongly recommend is checking`Insert spaces for tabs`. In fact, I would try to get in the habit of never using tabs, just spaces.

Other recommended settings

Here are some additional settings that I recommend. Leave the following settings **unchecked**:

- Restore most recently opened projects at start up
- Restore previously open source documents at startup
- Restore .RData into workspace at startup

And set "Save workspace to .RData on exit" to `Never`.

These settings will ensure that each time you launch R Studio you're starting with a clean, fresh session. This will make your code more easily reproducible, and help for debugging if problems come up.

Other notes

- All stored variables can be removed by running `rm(list=ls(all=TRUE))`
- When you first start R Studio you may only have three panels, the Script panel will automatically open when you make a new script `File >>> New File >>> R Script`
- When saving a script you shouldn't have to specific the "Save as Type", it should default to saving as a `.R` file type. If for some reason you wanted to save it as a different fie type, just change the extension (i.e., `script.txt`).
- Avoid using spaces in your file names, these can cause problems for programs like R. I use `-` instead.
- 'rivers' is a dataset built into R, you can access it by typing `rivers` in the Console.
- The lectures show you some point-and-click ways to do things (e.g., setting working directory, loading data). We want you to use what works for you best, but I **strongly encourage** you try to not go the point-and-click route. Making your code reproducible using scripts makes it way easier to solve problems when things break down. If it's not in the script, it's going to be really hard to troubleshoot.