R Intro MES 623 Spring 2020

Matthew Varkony

15 January 2020

Installation

1. Click here to download `R`.
2. Click here to download "R Studio," the Integrated Development Environment (IDE) that allows users to interact with `R`.

- Unique aspect about Studio IDE is that we can run languages besides `R` in them. This presents a good opportunity if code in another language such is needed for certain analysis. Other Language Engineges
- R Studio also provides an excellent platform for reproducible research. As an example this document is created in RMarkdown. The GitHub repository for this class holds both the PDF and .Rmd (RMarkdown File).
  - A good resource for everything R Markdown can be found here
    - Check the link if you want to see more ways that RMarkdown can be used to accomplish different tasks.
    - If you're interested there will be a Wetlab Website Night March 18th at 7pm with free drink tickets and food! A survey will be sent out for RSVPs in the coming weeks. Keep them eyes open.

Git with It!

Git is a version control application that is increasingly being used throughout the research community. Since we will be working in groups in this class for projects, it may be a cool way to familiarize each of you with this app. Whether you are interested in going into research or not, companies that work with data analysis are increasingly using GitHub throughout their departments so it is a skill that may be important to have. Also, if you are interested in doing other things such as creating websites or building resumes, you can search through other's GitHub repositories for outlines that you can build off in creating your own project.

We'll start out by creating a GitHub account. Instead of reinventing the wheel, we are going to follow the steps provided by Jenny Bryan's Happy Git. The set up will be done in class, but if you have any interest in diving a bit deeper, or if you run into any problems in your GitHub endeavors this is a great place to look for help!

1. Go to here and create a Git Hub account.

- A couple suggestions
  - Incorporate your name into your account name, it helps people identify who they are working with in the future.
  - Less is more. Use as few characters as possible, this makes it easier when creating paths or passing along your username.
  - Lower case makes everything easier. And using underscores or hyphens when separating words is probably best. You never know what other programming languages someone is using, and these are universally safe.

  - Since we are all students at the University we get free access to Git Hub Pro. This is beneficial for having private repositories. Such as the one we will be working in, where people will have to be invited to see and contribute to the information.

2. Install Git: Use Chapter 6 for directions. Once you are set up you are done and can move to the next step. This step won't be needed again.

3. Introduce Yourself to Git

We need to create a connection between the computer we are working from and our GitHub account. Think of it as a way for your computer and repositories from GitHub to communicate. We will all do this in R Studio.

For your user.name it can be whatever you would like to identified as when you are making changes to documents within a GitHub. I chose to use my full name to make it easy for future collaborators.

However, for your user.email you need to make sure that the email you type in is the email you used to create your GitHub account.

```
##install.packages("use.this")

library(usethis)
#use_git_config(user.name = "Matt Varkony", user.email = mvarkony@rsmas.miami.edu)
```

4. Connecting to the Hub: Use Chapter 9 of Jenny Bryan's Happy Git.

This process will take you through the terminal set up. Once you have done this, your interaction with the terminal will be whatever you want it to be. You won't need to use the terminal in your everyday GitHubbing unless you would like to. If that is the case there are more resources for learning how to interact with the terminal and GitHub that are beyond the scope of this setup. However, here is a good starting point.

Once you have completed the setup with GitHub on your computer you are ready to follow a similar procedure that links RStudio, Your Computer, and GitHub together for seemless editing.

As an aside you may be asked to input your username and password while going through this step; however, most likely this will be the only time that you are required to verify your identity. If this continues to pop up see Chapter 10.

5. Connecting Git and R Studio: Again, there is no need to reinvent the wheel so we will go to Jenny Bryan's explanation of connecting RStudtio to GitHub from this Chapter 12.

Awesome! We are all setting up your computer to GitHub. Now we are one step closer to working collabortively with others in class. I am still working on the details of how we will do this so that groups can be set up, but regardless this is a good way to manage version control of your projects for this class. While it may not seem necessary right away, this is a good skill to have for both future research or work within industry teams.

To Set Directory, or Not?

I have read over and over, setting a directory is a slippery slope. Its a good place to start, but a bad place to stay. Therefore, I will suggest that you create a new project for an assignment or group of assignments that have over-lap.

In this class you will have to complete four group assignments. This creates a perfect opportunity to practice efficient and effective coding and project management.

Lets start out by creating an project that is specifically for labs we will be doing this semester. Once we have `R` and `RStudio` we can open `RStudio`. From here we will

- Go to File > New Directory (creates a new Folder) > New Project

  - Here you will input the new directory name. This will be the name of the folder that the project and its contents are held in.
  - Make sure that this folder is in a location (a subdirectory) that you are comfortable storing your work in.
  - Now your new project is set up, and we can work on linking that to GitHub later. Chapter 16 and 17 does a good job of explaining this.

Once this new project is created it is **always** important to work within an `R Script`. This way we can track the functions and steps we use in our analysis. Think of it as a recorded history that is easily retrieved.

The way to open an `R Script` is to go to the top left corner of `R Studio` and click on the paper with the plus sign on top of it. Select `R Script` and a new window will drop down that you can start doing work in. * To run script while inside this `R Script` you need to go to the end of your commmand and click `command + enter`. This is a key board shortcut. You can also highlight the code and click the run button in the upper right hand corner of the window.

Typically I like to write a quick comment to start this script that describes what I will be doing. A great way to comment things out in R is to use the `#`. I typically include two `##` because there is a keyboard shortcut that removes one `#` and if I accidentally hit this, then I don't have to worry about uncommenting my code.

Packages: If You Can Imagine It, They've Probably Got It!

There are a few main packages that will be the workhorses of your day to day data work. So let's start by installing and calling the main one, if you haven't already. I've commented out the install.packages command because I already have this package installed. You only need to install a package once for it to be stored in your R folder.

However, for us right now, when using a specific package we will call `library(package.name)`. This equips our current `R` session with the package that has the functions we want to use for our data work.

```r
##install.packages("tidyverse")
library(tidyverse)
```

The tidy language was created by Hadley Wickham as a way to streamline and increase the ease of data manipulation and visualization. As you can see, `tidyr`, `dplyr`, and `ggplot2` are included in the `tidyverse`. These packages are synonymous with the `R` programming language at this point. Once you learn and become comfortable with the functions in these packages, data analysis will become 100x easier.

Another package that I like to load when I am doing data manipulation is called `tidylog`. This is package does a great job of summarizing what is going on "under the hood" of your function calls. It does not come with the `tidyverse` package, but it works well in tandem with the package.

```r
#install.packages("tidylog")
library(tidylog)
```

```
##
## Attaching package: 'tidylog'

## The following objects are masked from 'package:dplyr':
##
##     add_count, add_tally, anti_join, count, distinct, distinct_all,
##     distinct_at, distinct_if, filter, filter_all, filter_at, filter_if,
##     full_join, group_by, group_by_all, group_by_at, group_by_if,
##     inner_join, left_join, mutate, mutate_all, mutate_at, mutate_if,
##     rename, rename_all, rename_at, rename_if, right_join, sample_frac,
##     sample_n, select, select_all, select_at, select_if, semi_join,
##     slice, summarise, summarise_all, summarise_at, summarise_if,
##     summarize, summarize_all, summarize_at, summarize_if, tally,
##     top_frac, top_n, transmute, transmute_all, transmute_at,
##     transmute_if, ungroup

## The following objects are masked from 'package:tidyr':
##
##     drop_na, fill, gather, pivot_longer, pivot_wider, replace_na,
##     spread, uncount

## The following object is masked from 'package:stats':
##
##     filter
```

```r
##just as an example we will do some quick data maniuplation with the stock data set in R

#this command loads in the data to our Global Enviornment, even though we can't see it
#it is part of the utils package.
data(iris)
#check the names of the variables in the package
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```r
#check the structure of these variables
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Data Wrangling Resources

The most important thing to understand about learning to program is practice and exposure is the only way to get better. There isn't a pressing requirement to understand the theory behind computer science to be a good data analyzer. Being functional takes time, including failing, getting stuck, and working through your problems. With that being said, the best resource for all questions related to coding is the `Google` search bar. Believe that if you have a question about your code, someone else (or everyone else) has had a similar question. That means that the answer is somewhere online. The best coders are those that know how to ask the right question to get the result they want. So if you ever get stuck, try searching for the problem you are facing online!

Websites

Stack Overflow is a great resource, that will repeatedly pop up in your `Google` searches for code related questions.

Another great resource for Data Analysis is R for Data Science. Here you can go step by step through chapters and perfect your data manipulation skills. Something that is very important for preparing your data for analysis. The answer key to end of the question chapters can be found here.

One of the individuals heavily involved in the R coding language is Jenny Bryan. She was a professor at the University of British Columbia, and now she is part of Hadley Wickham's team at `RStudio`. They are the individuals that develop and maintain the open-source software that we all use. Her website is a treasure trove of resources for `R` and `Github`.

If you are already pretty familiar with `R` and you want to dig a little deeper, the Advance R workbook by Hadley Wickham is a great place to up your coding game. He goes through the theory a little more, helping explain conventions used in programming. By understanding the rationale behind your code, it allows you to tweak script in a way that is more efficient and accommodating to your needs. As they say, in order to break the rules you need to understand the game.

Another great resource for learning how to code and everything that goes along with it, including operating in the terminal and using Git comes from an economics professor at the University of Oregon. Grant McDermott. He has a class on Data Science for Economists where he discusses R, Git, the Terminal and the relationship between these three computer "tools". He also discusses cool things like parallel programming and web-scrapping data.