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 Enginges
- 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:
3. Introduce Yourself to Git

We need to introduce our Git account with our computer that is operating Git. Think of it as a way for your computer and the repository you are working in to communicate. We will all do this in R Studio since using Mac is relatively straightforward, and Windows is .... not.

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

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

You can use whichever name you would like, this is just how Git will identify who is making the changes within the repository. However, the user email that you input must be the exact same as the email you used to create your git account.

4. Connecting to the Hub

We will work through Chapter 9 of Jenny Bryan's Happy Git tutorial.

5. Connecting Git and R Studio
6. Our first Repo!

- Go to GitHub and log in.
- Let's click the green "New repository" button.
  - Give it any title you want, such as `myrepo`
  - Description: Describe what we are doing, so in this case: testing my setup
  - Public
  - Yes initialize this project with a `READ ME`
- Copy the URL via the green "Clone or Download" button.

2. Clone this repository to your computer via R Studio

- Open R Studio and go to `File --> New Project --> Version Control --> Git`. In "Repository URL" paste the URL of your Git Hub repository.
- Accept the default repo name (it should be the same as your repo on Git Hub)
- Save your repo in your class folder for this class. This is important to remember that you know where you are saving these repos.
- Check "Open in new session" and click "Create Project".

3. Local Changes to the Master File

- Add something to your read me file. A word, a letter. Doesn't matter just something that adds to the Read Me file.
- Commit these changes ...
  - Click the Git tab in the upper right hand corner of R Studio
  - Check the "Staged" box for `README.md` file
  - Click the Commit Button
  - Type a message in the commit box signifying what you did. ie) Commit from R Studio
  - Click Commit

- Click the green arrow and push your changes to the master repo.
- Now go to your repo on your internet window and check if the updates went through. If they did, great! We are done!

To Set Directory, or Not?

As 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 in. With that being said, why even start there? 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` downloaded click...

- Let's find the Git Repo that you want to clone. Search `mv-p/MES623_Spring2020`
- Clone this repository
- Open `RStudio` and go to the drop down `file` menu and select `New Project`
- Click on `Version Control` –> `GIT` paste the copied repo
- I typically leave project directory name the same as the repo name to keep things simple
- Then we are going to select the file path that we want it to be located in. For myself, I have folders for each class. So within my file path looks like this `"~/Desktop/College/MES_PhD/MES_623"`.

Okay, great. We have created a project in the directory of our choosing. This will help us stay organized in the future when we are dealing with bigger projects that require multiple scripts for analysis.

Next, 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 we are in business. 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.

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

The tidy language was created 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 `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 a package I recently found, and it 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.

```
#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
```

```
##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"
```

```
#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.

PRactice