Some Tools You Need For Reproducible Research

This document will walk you through various pieces of software that I want you to download (or create an account for) before attending my talk at the Athenaeum next Tuesday. These include:

- R and RStudio
- RMarkdown and Latex
- Git and Github

You should definitely go through this document and get all set up **before Tuesday**, as we all know how unlikely it is that everything downloads, installs, and works on the first try.

Reproducible Research

First, why am I asking you do get these things. Well, a complete project can be a massive beast. You have to gather, clean, and analyze data, write up your findings, and compile a document with all of your text, charts, figures, and bibliography. This becomes exponentially more complicated as you work on a project with someone else or with a team. The software I'm asking you to download will help you create a good workflow to make your day-to-day work on the project better, and it will help you stay organized when you come back to a project after many months away from it.

These tools provide a good workflow because R/RStudio can be used for data acquisition, cleaning, and analysis. RStudio then, in combination with RMarkdown and Latex, can compile your code, the results from your code (aka graphs and tables of regression results), and text into a nicely formatted document without you ever having to copy-paste. Git and Github provide a backup and version control for your files, as well as an effective way to collaborate on the same project/files with others.

You might say that this is overkill and that you don't need this. But I would argue that you can apply this workflow to small projects like homework assignments, so that when you do finally do a big data-analysis project (ahem, senior thesis), you'll have some of these important parts figured out. Plus, these might be great things to put on a resume, especially if you're hoping to go into tech, consulting, etc. So let's dive in to the software you need to get.

R Language

R is statistical programming software for gathering and analyzing data. It's like Python, but specifically for *statistical* work; it's like Stata, but better for programming; it's like Matlab, but free.

You need to download and install R. R is hosted on the "Comprehensive R Archive Network" (aka CRAN). It's not the prettiest website out there. To download R, go to https://cran.r-project.org/ and click on the download link for your operating system (Linux, Mac OSX, or Windows). Depending on your OS, you'll be taken to a second page where you have to choose which "version" or "package" to download. Linux users should choose their flavor of Linux (Debian, Ubuntu, etc.); Mac users should pick the top link (R-3.3.1.pkg); Windows users should choose "Base". You might have to click through one more page – just get whatever the recommended version is for your OS. After R downloads, install it as you would any other piece of software.

Once you've installed R, open up the application. It will be ugly. You'll basically get one floating window called the "R Console" with a bunch of text and a command prompt. At the prompt (which looks like a greater-than sign, >) type 2+2 and hit enter. R will talk back to you, telling you 4, like so:

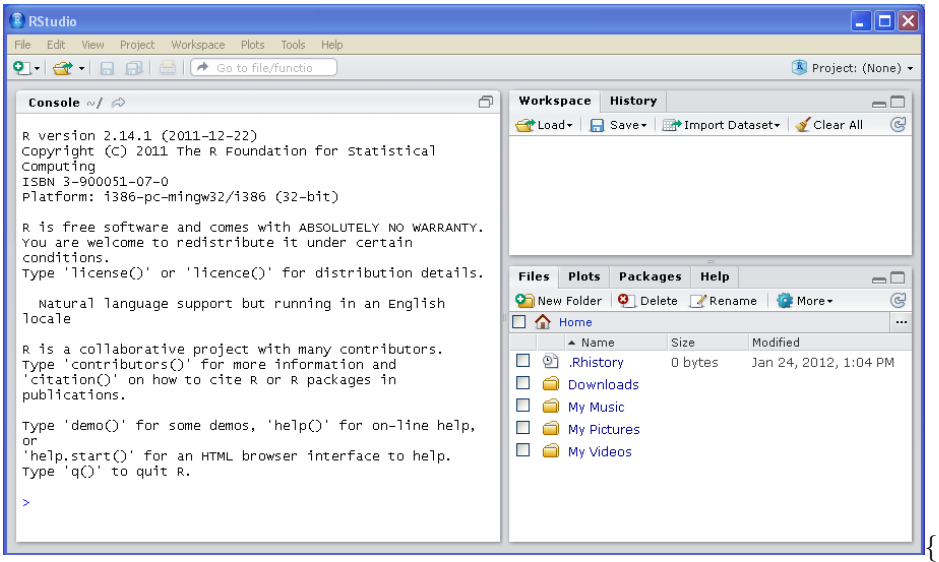
```
> 2+2

[1] 4
```

RStudio

That's a pretty unfriendly way to use R, so close the R application. To make the experience with R friendlier, you want something called an "Integrated Development Environment" or (IDE for short). My favorite IDE for R is RStudio. You need to download and install this as well. To do so, go to https://www.rstudio.com/. At the top, click on "Products" and then "RStudio". Choose "Desktop" and then "Download RStudio Desktop". Once downloaded, install it.

Now open up RStudio. It should find the R application you previously installed. It's running that application (R), but now with a much prettier interface:



{ width=40% }

On the left side of the screen in a window called "Console", again type `2+2` and hit enter. R will return the result. This time however, type:

```
x = 2+2
```

Now you'll see the object `x` in the "Environment" window on the top right. Go back to the console and type `x`. R will return `4`.

RMarkdown

Let's say we don't want to just code, but we want the code, some results from our code, and some free text (like paragraphs explaining what the code does) all to show up in the same document. RMarkdown is a way to make that kind of document.

Unlike what-you-see-is-what-you-get (WYSIWYG) programs like Microsoft Word, with RMarkdown you "code" your document, including the document's contents and its formatting. That means, for example, that you write "\beta" and when you compile the document, you see "β". Or, as another example, you code _word_ in order to set "word" in italics, like so: *word*. For a little intro on the RMarkdown "language", check out this site: http://rmarkdown.rstudio.com/lesson-1.html.

You don't need to install anything for RMarkdown. It's built into RStudio. In RStudio, click on the dropdown arrow next to the icon for a new document (this is at the top left of RStudio), then choose "RMarkdown". The first time you do this RStudio will tell you that it needs to install packages, say yes/ok and let those install.

If there is any error with this step, go to the bottom right window in RStudio and click on the tab called "Package". Then click "Install". Type "rmarkdown" (all lowercase) and click "Install".

Once you've got the packages installed, go back to the blank page icon at the top left and create a new RMarkdown document. A diaglog box will pop up asking for you to provide a document title and author, it also gives you the option to select "html" or "pdf", among other things. Right now, you have enough software to make an html document, but you don't have enough to create a pdf. For that, you need Latex. Hit "Cancel" and quit RStudio; let's go get LaTex.

LaTeX

This can get complicated fast, so let me try to provide you with just the right information. TeX is a computer program for typesetting documents. Whereas RMarkdown is basic and easy to use, TeX is super-unfriendly to use. But, TeX is like Microsoft Word for people who write math – everybody in academia uses TeX – so we need to learn how to use it as well.

Thankfully someone made LaTeX (pronounced la-tech). LaTeX is a set of macros for TeX that takes the experience of working with TeX from horrible to somewhat-manageable (there is no good/fun way to do TeX/LaTeX). You need to install LaTeX and doing so will automatically install TeX for you. LaTeX has a bunch of different "flavors" called MacTex or MikTex – they're all basically the same.

I don't remember all the steps to install LaTeX. So, for Mac users, google "install MacTex". For Windows users, google "install MikTex". For Linux users, type the following into the Bash shell: `sudo apt-get install texlive`. It should be a simple process though – just download and install like anything else.

We'll be using TeX/LaTeX as an intermediate step when creating pdf documents. In fact, if we do everything right, you may not even notice that TeX is being used. However, if you want to write LaTeX documents and compile straight to pdf, you can download an editor for LaTeX (this is like how we will use RStudio for R). I use TexMaker on Windows, but there are lots of options, and you should feel free to get whichever you like (or none of them).

Git

When you write code (which is how we'll be making documents – not with WYSIWYG), you often revise that code many times. It can be super helpful to have a shared document with a history of changes for your code/document. For example:

- your computer dies and you need a backup copy
- you made a bunch of changes you don't like and want to revert to a previous version
- your working with another person or team and everybody needs to work on the same document

Git with Github offers these solutions. Git is a version control system that is local to your machine. Github is a cloud-based location to store those versions and (as is clear from the name) works great with Git.

Let's first get Git. Go to https://git-scm.com/download/ and follow the directions.

GitHub

Now let's set you up on Github. If you have an account, log in. If not, create one by going to https://github.com/.

Getting It All Linked and Working

Let's see if we can get everything working, and to test that, we'll use the RMD version of this document. There are quite a few steps in this section. I'll explain what they do on Tuesday. For now, just get through them.

Checking that LaTeX and RStudio Work

On your computer, create a new directory (also known as a folder) and put the RMD version of this file in that directory. Open RStudio (remember, you should have closed it earlier – if not, close it and reopen it now). In the very top right there is a small arrow. Click it and choose "New Project". You should be able to select "Version Control" and then "Git". Choose the new directory you just created and click "Create Project" (the screen might flash once).

Go to RStudio's "Tools" menu and click "Global Options". Click "Sweave" on the left and then change the top dropdown from "Sweave" to "Knitr". Then exit global options.

Open the RMD file "prerequisites.rmd". A new window should open in RStudio on the top-left. That's this document! Click the "Knit" button in the top-middle. A new tab will open near the "Console" in the bottom-left called "RMarkdown". If everything works, in a couple seconds, this PDF will pop up. If not, there will be an error message in the "RMarkdown" tab. Google that error and try to fix your problem.

Getting Git and GitHub Connected

Go to RStudio's "Tools" menu, click project options, and then under "Git/SVN" change the dropdown at the top to "Git". If it asks you about "initializing a new Git repository" say yes. So far, we've created a project and started version control on our local machine. But we need to also link our local version control to the one on GitHub. Go back to RStudio's "Tools" menu and select "Global Options". In the box that pops up, choose "Git/SVN" and at the bottom click "Create RSA Key". There's no need to create a password. Click OK a couple of times until your back at the Global Options. There will now be blue-link text that says "View Public Key". Click that and copy the key. This makes your machine ready to talk to GitHub.

Now go to GitHub in a browser. On the top right, click your picture and choose "Settings". On the left, choose "SSH and GPG Keys". At the top right, click "New SSH Key". Give you key a name and then paste that long key that you copied in the "Key" box. GitHub is now ready to talk to your machine.

Now we need to link them. On GitHub, click the plus sign at the top right and choose "New Repository". Give the repository a name – it's a good idea to name it the same thing as that new directory you created on your computer a minute ago – and click "Create Repository". On the new webpage that popped up, there are 3 options. We want the middle one that says "...or push an existing repository from the command line". Go back to Rstudio and click on the "Git" tab in the top-right window. Under "More", choose "Shell...". A scary-looking terminal window will pop up. Type the two lines that you see on the GitHub screen into the terminal (the `git remote add` and `git push` commands). Be sure to type them *exactly* as GitHub tells you to (or copy and paste them). If you run into any errors – google until you figure it out. When done, close the shell/terminal window.

All right – finally everything is linked. Let's check that it works. In RStudio's "Git" tab at the top right, you should see the files that live in your project's directory and there should be two yellow question marks by each file. Click the checkbox next to "prerequisites.rmd" and the checks will turn into a green "A". Click "Commit" and a new window will pop up. Enter a "commit message" and click "Commit", then close that window. What you've done is saved a version of "prerequisites.rmd" in your local Git. Now click "Push" and you will have pushed that backup to GitHub. To confirm, go to GitHub in a browser (refresh the page if you've left it open) and you'll see that you've added "prerequisites.rmd" to GitHub. Well done!

Next

If you can do all of those things, then you're good to go for Tuesday's talk (and for a promising career in software development or research). I look forward to meeting you guys on Tuesday.

-Dan Yavorsky
dyavorsky@gmail.com
951-201-0927