NR995 Module 9

2017 Fall
Part III: How to use Git & GitHub
Basic workflow

After you've sufficiently configured git, GitHub, and RStudio as described in Part II, using git should be generally painless.

1. (Open RStudio and **Pull** to ensure you're working on the latest version)
2. Work normally
3. Save normally
4. **Stage** and **Commit** each discrete task (e.g., code for data cleaning/analysis/plot, a paragraph or section in a written document, etc) separately for simple & meaningful commit messages
5. **Push** your commit(s) to the remote repository

Merge conflicts

Merge conflicts occur when the same line(s) of a file are edited in two locations. This is most common in collaborations, but could happen if, e.g., you edit a file on GitHub and then edit the same file on your local computer without pulling the updated repository first. Git does a good job of merging files generally, but (rightly) throws a warning if there are simultaneous commits that both changed the same line. These are clearly marked and should be manually reconciled.

Accessory files

There are a few files that are unique to git or that GitHub treats differently.

- *.gitignore*: This file lists all of the files, file types, and subdirectories that are in your local folder that you do *not* want to track with git. When you initialize a repo on GitHub, you have the option of choosing an R-based gitignore. This automatically includes temporary and hidden files created by R & RStudio that are irrelevant. You can also add specific files to ignore (e.g., an R script published with an article you are using as a reference, intermediate files, etc). Anything listed in the gitignore file will not show up in the *Git* tab in RStudio.
- *readme.md*: GitHub will automatically render and display readme.md files. It is a useful place to provide a brief overview of the project, a description of the folders and files, etc. This works not just in the main directory, but in each subdirectory. For example, a folder called *code/* might have a readme.md that describes each R script in the folder.

RStudio features

You can perform all basic git operations from within RStudio. The interface makes it simple and (relatively) intuitive. We've already staged, committed, pushed, and pulled, but there are a few more handy features.

- *Diff* shows you line-by-line what has changed in the selected document since the previous commit. In this view, you can stage particular lines or chunks of your code in addition to the entire file.
- *Log of* in the *Git* dropdown menu shows a log of all commits for a given file
- *Revert* allows you to discard all changes you've made to restore the version from the previous commit
- *History* shows you the commit history for your project

(Some) GitHub features

While you can use git locally on your own computer, you'd be missing out on a lot of git's potential. GitHub has many features that are useful both for collaborating and for working individually. You should take some time to explore, but here are a few of the bells and whistles.

- Automatically rendering output from .Rmd
- History, blame, editing in place
- Commenting
- Issues
- Insights
- Deleting a repo