

Git/Github Lecture

Getting Started

Git is a version control system. Think of it as a better Dropout - you can track changes of your files and revert back to old versions of projects, but all without pinging your internet every microsecond. Why not just use Dropbox? I will not explicitly tell you why you should not use certain software (by all means, use what works for you), but I will tell you some of the benefits of Git:

- Most private sector jobs we will be qualified for use a combination of Git and Github.
- It sends a signal to employers (and other academics) that you have ability to learn new practical skills.
- It is excellent for collaborating.
- It integrates seamlessly with RStudio.

Unfortunately, Git has a rather steep learning curve. This is mainly due to the jargon that comes along with it. I will do my best to define terms, and hopefully make the process a little less confusing.

The next few subsections are dedicated to getting all of software installed and communicating with each other for integration. Honestly, this is an extremely painful process without good resources. Fortunately, there is a good one made by Jenny Bryan that I will be following closely throughout this document.

Installing Git

Git is much different from Github. Git is the actual version control software, while Github is hosting service that provide a home for your Git-based projects. Think of Github as an online repository of files that Git can communicate with to make changes, while Git is the actual software that is tracking your changes. Before we can get to Github, we need to install the Git software on our local machine. But first, let's check if you installed it at one point already (all of us probably tried learning at one point). Open up the terminal in RStudio and type in the following:

```
which git
```

```
## /usr/bin/git
```

Did you get output showing you a file path? Great! You have Git installed and are ready to move on to the next section. If not, you should see something like `git: command not found` and you will need to install Git. Here is where to download:

Windows:

- <https://gitforwindows.org/>

Mac:

- <http://git-scm.com/downloads>

Create a Github account

Once you have installed Git, it is time to make a Github account. Go [here](#), and create an account. **Give your username some thought** since it can be a pain to change in the future. I recommend incorporating your actual name into your username. Why? Github is a website many professionals use and it's important to associate yourself with your work. Additionally, you can create your own website and host it for free using Github, but the url will be something like `yourusername.github.io`. Hence, choose your username wisely, as it will be how many access your research and materials for the foreseeable future. Also I recommend **not using your ucsb email** for your Github account. This email will disappear when you graduate. There is no need for a headache 5 years from now.

Introduce yourself to Git

In the shell (terminal), type the following:

```
git config --global user.name 'Jane Doe'
git config --global user.email 'jane@example.com'
git config --global --list
```

The Main Commands

Assuming all of this is done on the main branch.

Stage changes (get ready for adding them to a version)

```
git add -A
```

Commit changes (commit to adding these changes to your version)

```
git commit -m "this is a commit to take the staged changes and save them as a version"
```

Push the changes (push)

```
git push
```

Branches

Branches are one of the most appealing features of Git. As an example, suppose you want to try a new spin on your paper - you want to try a whole new analysis, but don't necessarily want to "commit" (bad pun) to the changes. This is where branches come in. A branch is essentially a clone of your files. You can make changes, create new documents, etc. but you can do this all without changing your master copy. However, if you like the changes you make on your branch, you can merge them into the master copy. Don't like your changes? Then just delete the branch and do a hard reset (command coming soon) to bring yourself back to the original. For the purposes of this document, we will call our main copy of our files the "master branch". This is generally the default on Github, and you should either adopt this convention or find a very intuitive substitute.

Branches have a steep learning curve (as does all of Git), but they are extremely important when working collaboratively. Consider the following workflow which, as you can see, would cause problems:

*You and your coauthor both decide to work on the R script titled **regressions_main.R** at 10:00am. You want to try one analysis, while your coauthor wants to try another. At 10:30am, you find some unique results that you believe really enhance your paper. You save the file, commit the changes, and (try to) push to Github. However, 5 minutes before you, your coauthor found different results that they believe are worthy of saving. They saved, committed, and pushed to Github at 10:25am. Now when you try and push your changes, there is an issue. Git had already saved the version that your coauthor pushed at 10:25, so it does not know how to merge in your changes with the new lines your coauthor made.*

Branches solve this problem. Each person can make their own copy of the master branch (call this `branch_objective`), and begin working on their changes. They can save, commit, and push their changes to their branch, thus allowing Git to keep track of their changes, all without merging to the master branch. If you want to change the master branch you can “create a pull request” on Github and assign your coauthor to look over the changes on which they can accept or deny. If accepted, these changes get merged into the master branch. Pull these new changes in, and rinse and repeat.

Common Commands in the Shell:

Make a new branch and switch to it:

```
git checkout -b "branch_name"
```

Push the branch to Github so it will sync.

```
git push -u origin branch_name
```

Now you can continue making changes as you did before. One new thing will be that you will need to go to your Github account and merge the changes you made by “creating a pull request”.