Reproducible Research I: version control

LPO 9951 | Fall 2015

PURPOSE Today we will discuss the idea of version control more fully than we have in past courses. We will then walk through git and GitHub, the primary syntax and host currently used for version control.

Reading

For today's class, I asked you to read, "Git can facilitate greater reproducibility and increased transparency in science", by Karthik Ram. Let's discuss it with the following guiding questions:

1. Why version control at all?
2. How does version control aid in collaboration?
3. What are some past methods you have used for version control? Currently?
4. What system do you think will use in the future and why is it git?

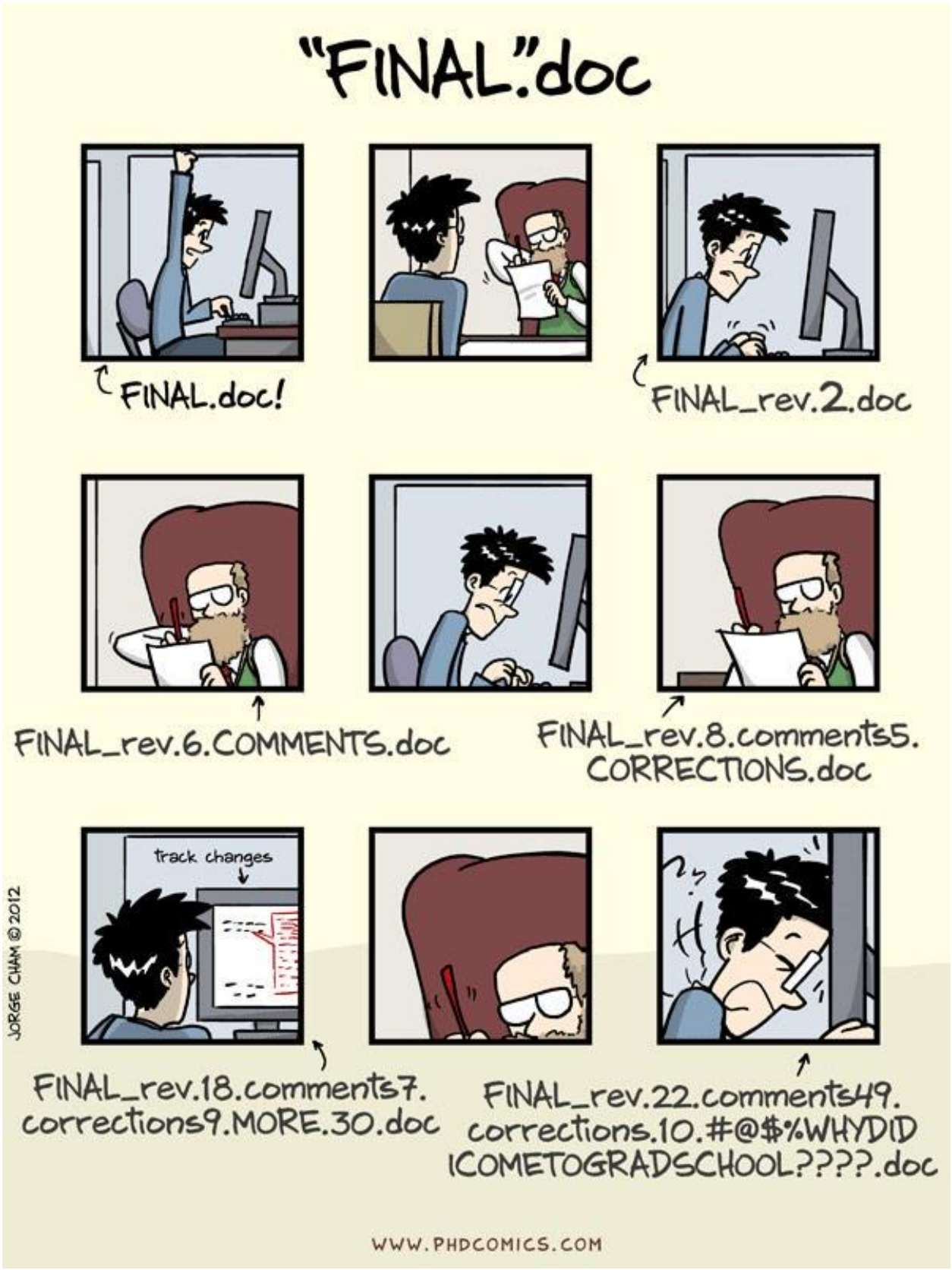
Git

From the git website http://git-scm.com/

> Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Why use git?

With so many other (read: easier!) ways to share and collaborate on documents, why use git? Isn't it a bit overkill to learn an entirely new syntax? Why not just email files or use something like DropBox? Because it is very easy to end up with something like this:



Credit: Jorge Chan

As complexity increases, so does the need for git

$$Project = f(size, scope, collaborators)$$

As any part of that function grows, so too does the need for a work flow that:

- Allows for many moving parts
- Allows for peaceful collaboration (no overwrites)
- Allows for timely collaboration (synchronous)
- Allows for distant collaboration (centralized file location)
- Allows for version control (so you can go back if necessary)

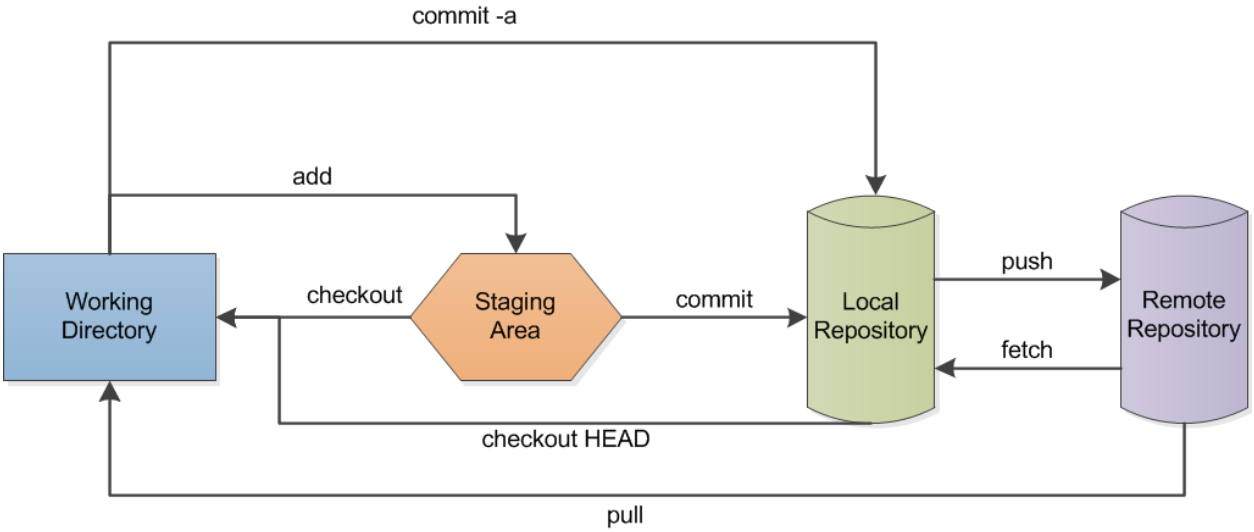
Git was designed to do all these things, so it works better than other systems.

What's the difference between git and GitHub?

Yep, you guessed it. Git is the system/language that supports version control. GitHub is an online service that will host your remote repositories, for free if public or a small fee if private. (Students get an education discount and some free repos. Check out https://education.github.com/.)

GitHub also provides a lovely GUI that automates much of the git workflow. If you want to run git at the command line, go for it! But using the GUI is fine.

How does git/GitHub work?



commit -a: Directly commit modified and deleted files into the local repository (*no new files!*)
add: Add a file to the staging area.
checkout: Get a file from the staging area.
checkout HEAD: Get a file from the local repository
commit: Commit files from the staging area to the local repository
push: Send files to the remote repository
fetch: Get files from the remote repository
pull: Get files from the remote repository and put a copy in the working directory

Credit: Lbhtw (Own work)

Some notes on work flow (good habits)

1. Always `sync` your local repo with your remote repo before starting any work. This makes sure you have the latest files. The GitHub GUI has a `sync` button on the top right of the window. (`Sync` is really just a `pull` command.)

2. Don't wait to `push` your commits. Just like you save your papers every few lines or paragraphs, you should push to your remote repo. This way, you're less likely to lose work in the event of a computer crash. Also, should you want to return to a prior version, small changes make it easier to find what you want.

3. Add useful `commit` messages. The GitHub GUI will make you add something. Don't say "stuff." A version history of 95 "stuff"s is pretty non-helpful.

4. Don't freak out! If you accidentally push and overwrite or delete something on the remote, you can get it back. That's the point of version control! Sometimes weird things like merges happen (two files with the same name are shoved together). They can be a pain to fix, but they can be fixed.

5. Avoid tracking overly large files and pushing them to your remote repository. GitHub has a file size limit of 100 MB per file and 1 GB per repo. The history of large files compounds quickly, so think carefully before adding them. Usually this means that small datasets are okay; big ones are better backed up somewhere else and left on your machine.
6. Remember: even in a private repository, your files are potentially viewable. If any of your datasets or files are restricted, do no push them remotely. Use `.gitignore` to make sure that git doesn't track or push them.

Today in class

For class today, I want you to try to accomplish the following tasks.

Online tutorial

First, complete the try.github.io tutorial. It should take just a few minutes. It will, however, make you use the command line method.

On your own

After completing the tutorial, try to complete the following tasks with your own GitHub account. You can use the GitHub GUI this time, or if you're feeling bold, the command line/terminal.

1. Initialize a private repo on GitHub.
2. Clone that repo to your computer.
3. Add a text file called `hello.txt` to your cloned repo (on your machine) that says "Hello, World!".
4. Commit and push your new file up to GitHub. Check that it's there.
5. Make a change to your `hello.txt` file locally, commit and push the change to the repo. Check the change on GitHub.
6. Make a change to your `hello.txt` file using GitHub's editor and save it remotely. Use the GitHub GUI to sync your change locally.
7. Find my GitHub Gist that gives Stata code to automate the process of converting categorical variables to dummy variables. Fork or clone this gist. The code is now yours to use and/or improve upon!

Init: 30 August 2015; Updated: 31 August 2015