Rec8-Git-Intro

November 17, 2016

1 Introduction to Git

1.1 Use git for tracking your own work, and for sharing with others

- Mark Histed
- mark.histed@nih.gov

161106: created

2 Why git?

Git is a change tracker. (Or, git is version control.)

2.1 Why version control?

1. Version control helps you manage your own changes, organize what you did in the past, and roll back changes that don't work out.
2. Version control helps you collaborate with others. In modern science and data science, version control is mandatory to work with collaborators both local and global. While several other version control systems are used, git is the largest and the standard.

2.2 Git vs. github

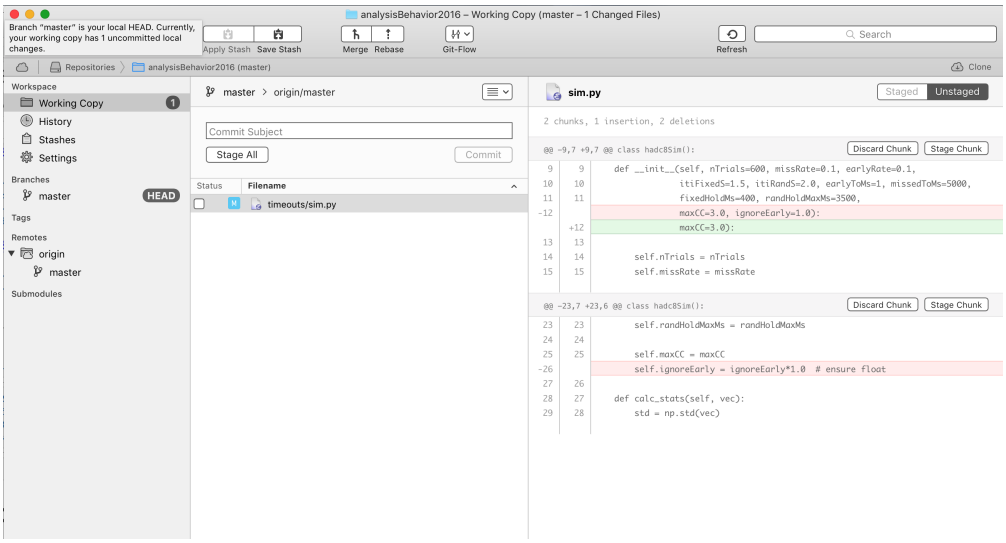
- **git** is software and a command line tool. (Tower is a good Mac GUI client that can be used instead of the command line tool.)
- **github.com** is a website. It hosts remote git repositories. You don't need to use github with git, but we will.

2.3 Terms

- Repository
- Working directory

---

1. *Example: Look at repo, git status. Make a change. Remove it.*



image

Example: Make a change. Keep it, add/commit. - index, working tree, revision history
Usual workflow - can use it for keeping track of your own work.
Push it to the server for everyone to see

- two graphs, git provides tools to keep them up to date.
- fetch, pull, push, merge
- Show some changes on github. List of commits. Current versions. Blame.

2.4 Setup details (for git shell)

Personalize git so it knows who is making changes:

```
git config --global user.name "Your Name"
git config --global user.email "your_email@whatever.com"
```

Add an alias that helps you track the history using character based "graphics".
Copy the following into ~/.gitconfig for your full color git lola action:

```
[alias]
        lol = log --graph --decorate --pretty=oneline --abbrev-commit
        lola = log --graph --decorate --pretty=oneline --abbrev-commit --all
[color]
        branch = auto
        diff = auto
        interactive = auto
        status = auto
```

3 Recap

- git can show you what you are working on, allow you to revert, and keep track of old changes
- by using a central repository, allows collaboration
- show command line versions

4 Other resources

- This git cheat sheet is pretty good for the basic commands.
- This tutorial (part 1, part 2 is good for illustrating graphically how branches and merges work, using gitk.
- Main git website.
- A quick git tutorial from github.com.
- Git reference page, a good concise summary of a lot of commands.
- The official git book. This is worth a read, in my view. But be ready to spend a few hours of time on Chapters 1,2, and 3 to get the basic commands down.
- The git tutorial from Bitbucket Cloud is interesting. Bitbucket is a competitor to github, so I'd largely ignore when they tell you to send your changes to Bitbucket – use github. The best chapters are "Getting Started" and "Collaborating". You can ignore anything about Subversion (a prior competitor to git that is now fairly obsolete.)

4.1 Key commands and tasks to know

- Make a change: add, commit
- Inspect changes and history: status, diff, lola
- Add just a subset of changes: add -i, or use a GUI like Tower
- Roll back a non-committed change: checkout, reset
- Roll back a committed change: git branch (Tower: reset HEAD)
- Merge two branches: merge
- Merge a local and remote branch: pull
- Push your latest changes onto the server: push
- Copy a repo to your computer for the first time: clone

4.1.1 Other commands I use from time to time

- Create a new branch: branch
- Grab a single commit onto another branch: cherry-pick (when you have made mistakes, this can sometimes help fix them)
- Store your current changes *temporarily* so you can change branches, etc.: stash
- Don't count on these stashes lasting very long; they can get lost in the process of a few merges. Use them once, then delete. If you need long-lived stashes, use actual commits onto a new branch.

4.2 My general advice about learning git

- **Keep backups**. It is possible to bork your entire repository with git. If you ever use –force, it's possible to bork the server too. (Don't listen to online sources saying git always keeps every old change. It does, but sometimes not in a way that's worth spending the hours it would take to recover.) Time Machine keeps you safe. Back up daily.
- Understanding git is hard. Spend some time thinking about what's going on. Stare at the history graphs and figure out how they work. Learning git is worth it, it will more than make up for the time you spend learning in the time saved working with others and organizing your own code.
- Ask me questions if you are confused.

4.2.1 Things to ignore if you are looking online

- Ignore the online noise about **merge vs. rebase** - it's irrelevant until you are an advanced user. Never use rebase until you're an advanced user.
- Mostly ignore the **debates about git "workflows"** until you are an advanced user. Workflows refer to the number of branches teams keep open, which branch new commits go to, and when and how merging happens between branches. For example, one workflow is to keep `master` as the stable version of your code, make new commits to another branch, say `Run/CurrentTest`, and after testing, merge `Run/CurrentTest` into `master`. Many code shops use more complicated workflows, with more branches, daily automatic merges, and daily automated tests. For new projects we can discuss worksflows, but initially you can just emulate what we do.
- Ignore whether your merges are fast-forward or not. It's not important until you are an advanced user (if ever).

4.3 Need to check (MH)

- Download git for Mac? https://git-scm.com/download/mac
- Tower