Git status identifies

* Which files have been changed;
* Which of these files have been “added” to the staging area, and which have not;
* Which files in your repository folder are not currently being tracked by git (because they have never been added and committed).

Git add

This adds files to the staging area. It identifies any files that have changed from previous commits and adds them to a staging area. You don’t have to add all files, but can identify them by specifying them by name, but we are going to add all of them in this instance.

Git commit

It's like when you save a file, except with Git, every time you save it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. You can move between commits by referencing the hash.

Git push

the push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo for others to access.

Jacobian matrix is the matrix of all its first-order partial derivatives.

A Hypothetical SEIR Model

Chapter 2

* The first thing we’ll be doing is using Git to download the tutorial files from GitHub.
  + If you already have a GitHub account set up and are used to using it from your computer, some of this won’t be new to you. Sorry!
* Git has become the world’s most popular version control system. It’s fast, flexible, and pretty easy to learn well enough to use day-to-day.
  + There are a number of popular, powerful websites that will do remote git hosting and collaboration; today we’ll be using GitHub. While you may not be able to use GitHub for your DoD work, there are quite a few options run from within DoD, and you can likely make one of those work for you.
* Prereqs:
  + Git installed on your computer
  + A GitHub account
  + Terminal access
* Terminology
  + Git at the most basic level: software that detects changes between different versions of files, tracks the history of those changes, and provides tools for merging versions of those files.
  + “Git Bash” – a version of the bash shell for Windows, which comes with Git for Windows. Lets you run some Linux command line stuff on Windows, plus, of course, Git.
  + GitHub, GitLab, BitBucket, etc: websites that host Git repositories online and have built additional tools around them to make collaboration and project management easier.
  + There are also plenty of Git clients that add graphical user interfaces, which we won’t really be using today. One example is GitHub Desktop.
  + TODAY: we’ll use the command line (Git Bash if you’re on Windows, your built-in shell on Linux or MacOS) to run various git commands on our own computers, some of which will involve communicating with GitHub. We’ll also use the GitHub website itself.
* Global Git Configuration
  + We need to set up Git with your default username and email address.
  + [Run code in book.]
  + This is all you need to use Git locally, on your own computer – you could run the example code to create a local repo now. But Git is more useful when you connect a local repo with a hosting site like GitHub and use it to share your work.
* To do this, we’re going to get you set up to communicate with GitHub using the SSH protocol, which, after setup, is the easiest and most secure way to do it.
  + We’ll be using public key authentication. This means we need to generate a pair of cryptographic keys, then provide the “public” key to the site we’d like to authenticate with. When we connect later on, your computer will perform some cryptographic magic to use your private key to prove that you really are the person associated with the public key GitHub has on file, without ever actually exposing your private key.
  + **BEFORE RUNNING THE CODE SHOWN: run “ls ~/.ssh”. If there’s a .pub file in that folder already, you may want to skip creating a key pair and just use what you have.**
  + To generate a key pair, run the following code.
    - Might need to replace “ed25519” with “rsa –b 4096” but probably not
    - Hit enter to accept the default file path
    - Add a passphrase if you want – good practice, but not absolutely required
  + We need to get it on to GitHub now. Run the “cat” line to do this – it’ll print out the public key we just created.
    - Copy EVERYTHING THAT PRINTED OUT with with ctrl+ins or highlight -> right click -> copy
    - Go to the GitHub SSH Key settings page and paste it in.
    - Very similar process for other Git hosting sites – just find the ssh keys page in the settings menu.
  + Fork + clone the repo
    - Now that our computers can talk to GitHub, we can download the tutorial files.
    - 1. Go to the repo page on GitHub
    - 2. Click the “Fork” button – this will create a copy of the project/repo in your own GitHub account. This isn’t always necessary, but we don’t have an easy way to give you all permission to directly “clone” our repo, so we’re having you create your own copies.
    - 3. On the new project page, click the “Code” button and then click the stylized clipboard to copy the URL that shows up. You should see “SSH” underlined in red and a URL that starts with [git@github.com](mailto:git@github.com); if you see HTTPS in either spot instead, try picking SSH instead before copying.
    - 4. `cd` to where you want to create your repo folder
    - 5. Run `git clone` on the URL you downloaded. This will create a new folder called etf-r-git-2020, with the repo files inside.
* [DUSTY DOES STUFF]
* Dusty finishes differential equations section
* Dusty says, that plot looks great, but it needs a title
  + Dusty adds the title, explains the add-commit-push process as he goes
* Robert takes over screen sharing, pulls, shows how he got the change
  + Share my screen, with repo open
  + NOTE to everyone: because you are on a forked version of the repo, you won’t be able to exactly reproduce what we’re doing. There are ways to contribute back to the original repo that you forked from, but we’re not going to cover that here.
  + Check the file – nope, no change. Because it doesn’t sync automatically; I have to tell git to pull down the latest changes.
  + Run `git pull`
  + Show that I now have the changed title
  + You may also see git fetch – this checks for changes without actually downloading and merging them into your current working version of the files.
* Robert decides to add a subtitle, adds and commits the change, tries to push – but fails, because Dusty has done the same thing in the background
  + Ok… this is better, but I think we need a subtitle!
  + Add code: `subtitle = “System reaches equilibrium around day 180.”
  + Save file
  + Add file
  + Commit changes
  + Try to push… get an error
  + Dusty, what happened?
  + Dusty explains that he also pushed the same change
* Robert goes through the merge and shows how to resolve conflicts
  + Robert pulls again, then has a merge conflict
  + Run git status
  + Show which file has a conflict
  + Open the file, look for where the conflict is
  + Get it to how I want it
  + Save
  + Add
  + Commit
  + Push
  + Show commit history on GitHub with merge commits in it
* Robert explains branching, with simple demo if time.
  + How could we have avoided things like this? This was trivial, but merges can be WAY more complicated.
  + BRANCHING!
  + Parallel histories, which can then eventually be merged all at once
  + GitHub/Lab/etc have tools to help you with review and approval before merging branches, to avoid this kind of conflict breaking important pieces of code
  + Branches are really just pointers to certain commits on the tree; nothing super fancy going on here, but incredibly useful.