Introduction to Git/GitHub

Tools for today: Git and GitHub

Make sure that you have installed **git** on your computer:
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

If you haven't already, **please sign up for GitHub** (https://github.com)

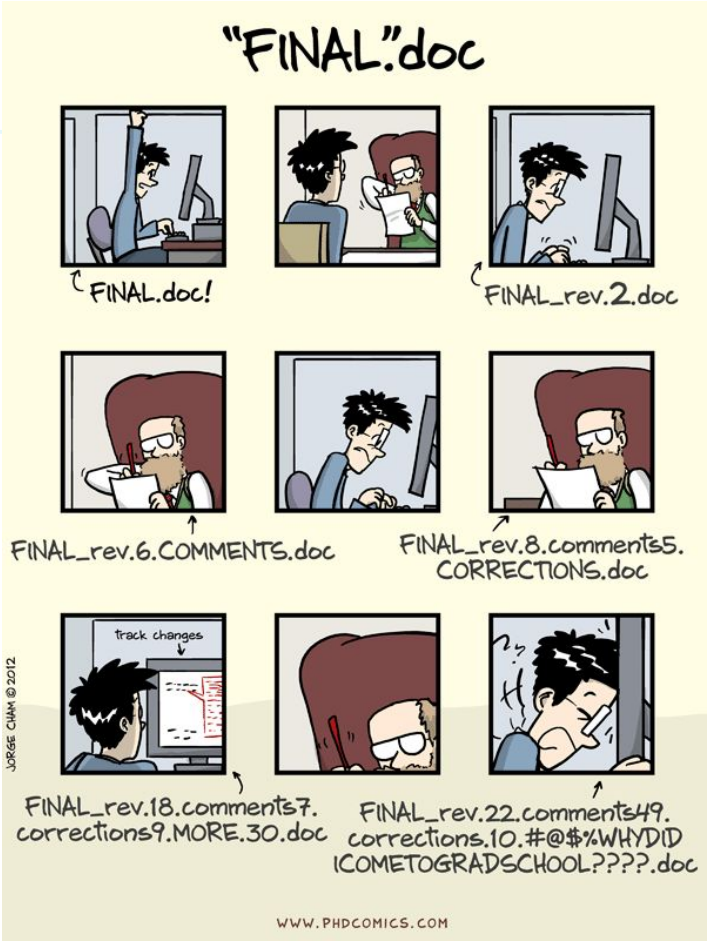
- Sign up for the student pack (https://education.github.com/) to get unlimited private repositories. You are a "student" and want an "individual account".

While optional, I highly recommend downloading the following software:

- **GitHub Co-pilot:** https://github.com/features/copilot
  - AI code completion tool developed by GitHub and OpenAI
- **GitHub Desktop:** https://desktop.github.com/download/
  - GUI for interacting with GitHub (as opposed to command line only)
- **GitKraken:** https://www.gitkraken.com/
  - A fancier and nicer alternative to GitHub Desktop

What is git?

- A version control system
- Stores data as a series of snapshots
- If files have not changed, it will simply access the file from a previous commit instead of saving it again
- Allows access to all the committed steps along the way



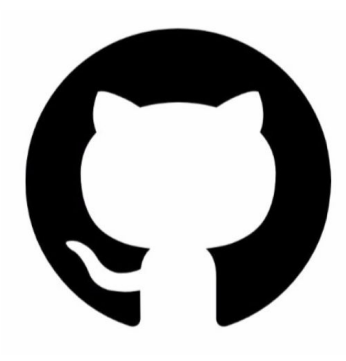
Git vs. GitHub

Local Git Repository

- You have a local version of the folder on your computer
- History stored in .git file
- Only you can see the changes made in the local version

Remote GitHub Repository

- A remote version of the folder is hosted on the GitHub website
- Everyone can see these changes (if repository is public)



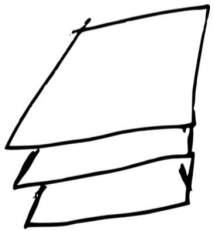
Why do we need Git/GitHub?

- Imagine working on a project with several collaborators...
- Using Git/GitHub allows everyone to have their own local version of the project while still maintaining a “main” version of the project, hosted remotely on GitHub
- You can make changes freely without people seeing what you are doing
  - You can thoroughly test your changes before adding to the master copy
- Version control!!
  - Especially great if your changes create bugs because you can backtrack/revert

Typical Git/GitHub Pipeline

(2) make local changes

(e.g., create file called filename.txt)



LOCAL REPOSITORY

(1) git pull

(to retrieve the most recent version from the server)

REMOTE REPOSITORY

(3) git add filename.txt

(changes are staged/waiting to be committed)

(4) git commit -m "[description of changes]"

(commit when you have made some changes and want to be able to save your current checkpoint as a snapshot)

(5) git push

(make changes available to everyone with access to the repo)

Warning: remember to "git pull" before "git push" to mitigate potential merge conflicts

Let’s set up a GitHub repository for this class

Setting up a GitHub repository

1. Three (of many) possible starting points:
   a. Start with an *existing* remote repository from GitHub
   b. Create a *new* remote repository on GitHub
      i. I usually initialize a **private** repo with a **README** file and with a **license** (e.g., MIT)
   c. Convert local folder into a git/GitHub repository

Setting up a GitHub repository

1. Three (of many) possible starting points:
   a. Start with an *existing* remote repository from GitHub
   b. Create a *new* remote repository on GitHub
      i. I usually initialize a **private** repo with a **README** file and with a **license** (e.g., MIT)
   c. Convert local folder into a git/GitHub repository

2. Clone remote repository to your local computer via one of the following:
   a. **Command Line:**
      `git clone https://github.com/[username]/[repositoryname].git`
      e.g,. `git clone https://github.com/tiffanymtang/dsip-s25.git`
   b. **GitHub Desktop:** click green “Code” button > “Open with GitHub Desktop”

Setting up a GitHub repository

1. Three (of many) possible starting points:
   a. Start with an *existing* remote repository from GitHub
   b. Create a *new* remote repository on GitHub
      i. I usually initialize a **private** repo with a **README** file and with a **license** (e.g., MIT)
   c. Convert local folder into a git/GitHub repository

2. Clone remote repository to your local computer via one of the following:
   a. **Command Line:**
      `git clone https://github.com/[username]/[repositoryname].git`
      e.g,. `git clone https://github.com/tiffanymtang/dsip-s25.git`
   b. **GitHub Desktop:** click green “Code” button > “Open with GitHub Desktop”

You should now see a local folder named `repositoryname/`

Making changes to your repository

3. Make changes to your repository, e.g.,
   a. Create a new file called info.txt
   b. In your info.txt file, add the following two lines:
      name = “Jane Smith”
      github_name = “janesmith”

4. Follow “Typical GitHub pipeline”: in command line,

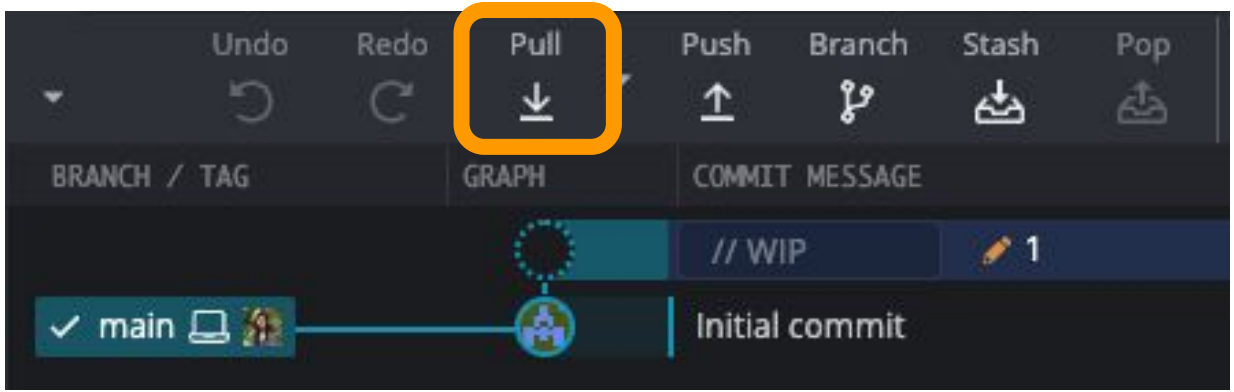
```
a. git pull                          # Retrieve latest version of repository from remote
b. git add info.txt                  # Stage changes (can add/remove multiple files)
c. git commit -m “added info”        # Commit staged changes to save a snapshot
d. git push                          # Push from local to remote repository
```

Alternatively, can accomplish all of these steps using GUIs like GitHub Desktop or GitKraken.

Managing your repository using GitKraken

3. Open README.md and edit file
4. Follow “Typical GitHub pipeline”: in GitKraken,

```
(a) git pull    # Retrieve latest version of repository from remote
```

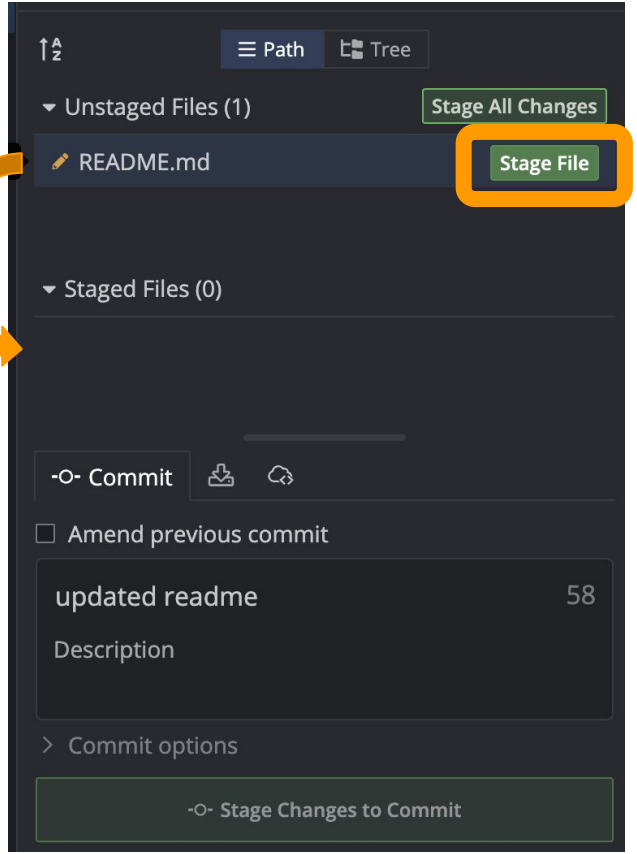


Managing your repository using GitKraken

3. Open README.md and edit file
4. Follow “Typical GitHub pipeline”: in GitKraken,

```
(b) git add README.md

# Stage changes (can add/remove multiple files)
```

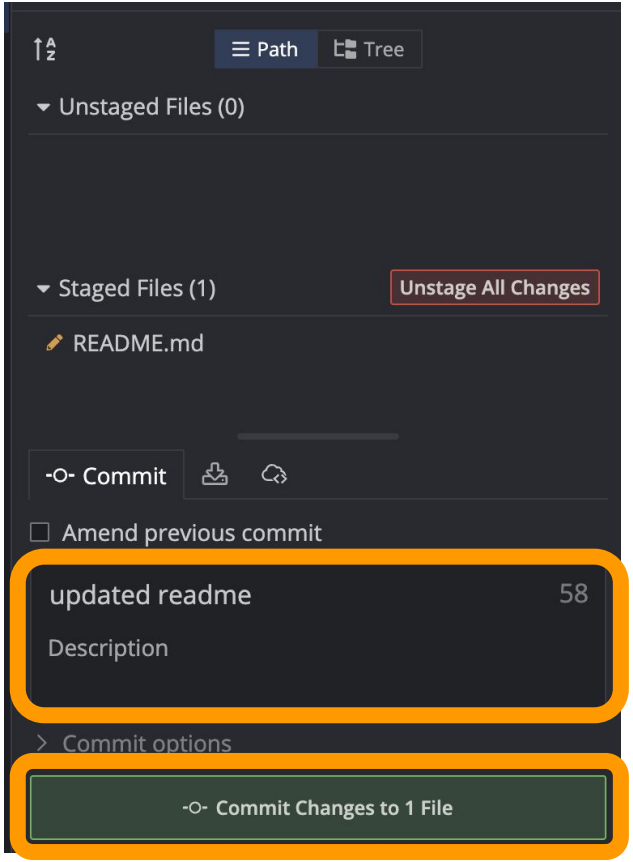


Managing your repository using GitKraken

3. Open README.md and edit file
4. Follow “Typical GitHub pipeline”: in GitKraken,

```
(c) git commit -m “updated readme”

# Commit staged changes to save a snapshot
```



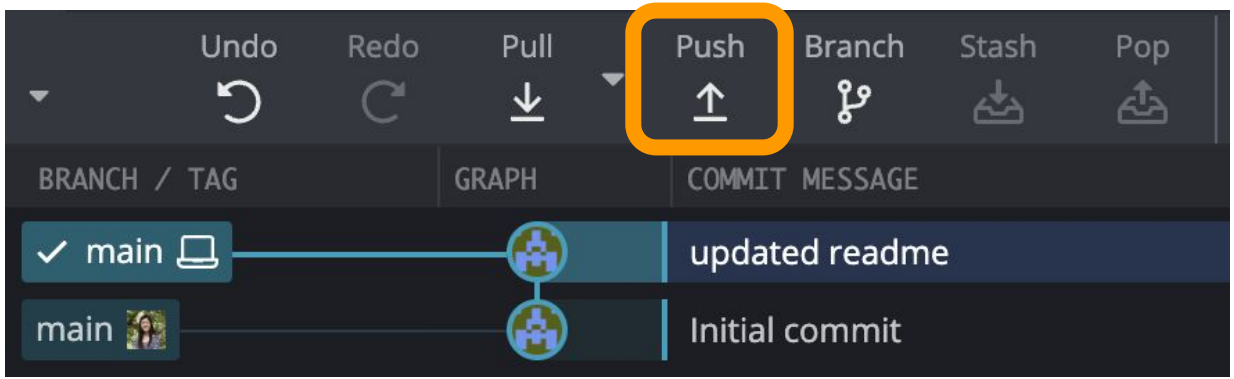
(i) add description

(ii) click commit

Managing your repository using GitKraken

3. Open README.md and edit file
4. Follow “Typical GitHub pipeline”: in GitKraken,

```
(d) git push    # Push from local to remote repository
```



Utilizing .gitignore

- Often, we do not want to track changes and push all files to GitHub
- Some types of files or folders that we usually do not want to track:
  - __pycache__/
  - .DS_Store
  - data/
- We can add these files to the .gitignore file:

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
.DS_Store

__pycache__/*
data/*
```

Git Cheat Sheet

- To check the status of working directory and staging area: **git status**
- To see what is different/changed in file(s) but not staged: **git diff**
- To create a new branch at the current commit: **git branch [branch-name]**
- To switch to another branch: **git checkout**
- To save modified and staged changes (e.g., in order to change branches or pull from remote): **git stash**
- To retrieve previous stash: **git stash pop**

Full cheat sheet: https://education.github.com/git-cheat-sheet-education.pdf

Adding collaborators to your GitHub repository

On GitHub, go to:

Settings > Collaborators > Add “tiffanymtang”

Getting started with Lab 1

- Instructions, redwood paper, and data found on Canvas
- Carefully read through Tolle et al.
- You should be able to start thinking about and answering the problem formulation/data collection questions in Part 1

Recap + What's Coming Up

Recap

- **Git** is a version control system. **GitHub** hosts git repositories remotely + does more.
- **Typical GitHub workflow:** git pull, add, commit, push

Coming up...

- Reproducible environments with renv in R and conda in python
- Hands-on practice with data cleaning + exploratory data analysis [chapters 4 and 5 from VDS textbook]