The Intersection Between Data Sharing and Code: Version Control Best Practices With GitHub

Fall RT2 - Online

Fernando Hoces de la Guardia, BITSS
29 November 2023, slides

Before We Begin

- GitHub desktop app installed?
- Create an account at GitHub.com?
- If no to any, please do it now
- Structure of the presentation:
  - Brief overview data sharing.
  - Hands-on tutorial on Git/GitHub for non-programmers.

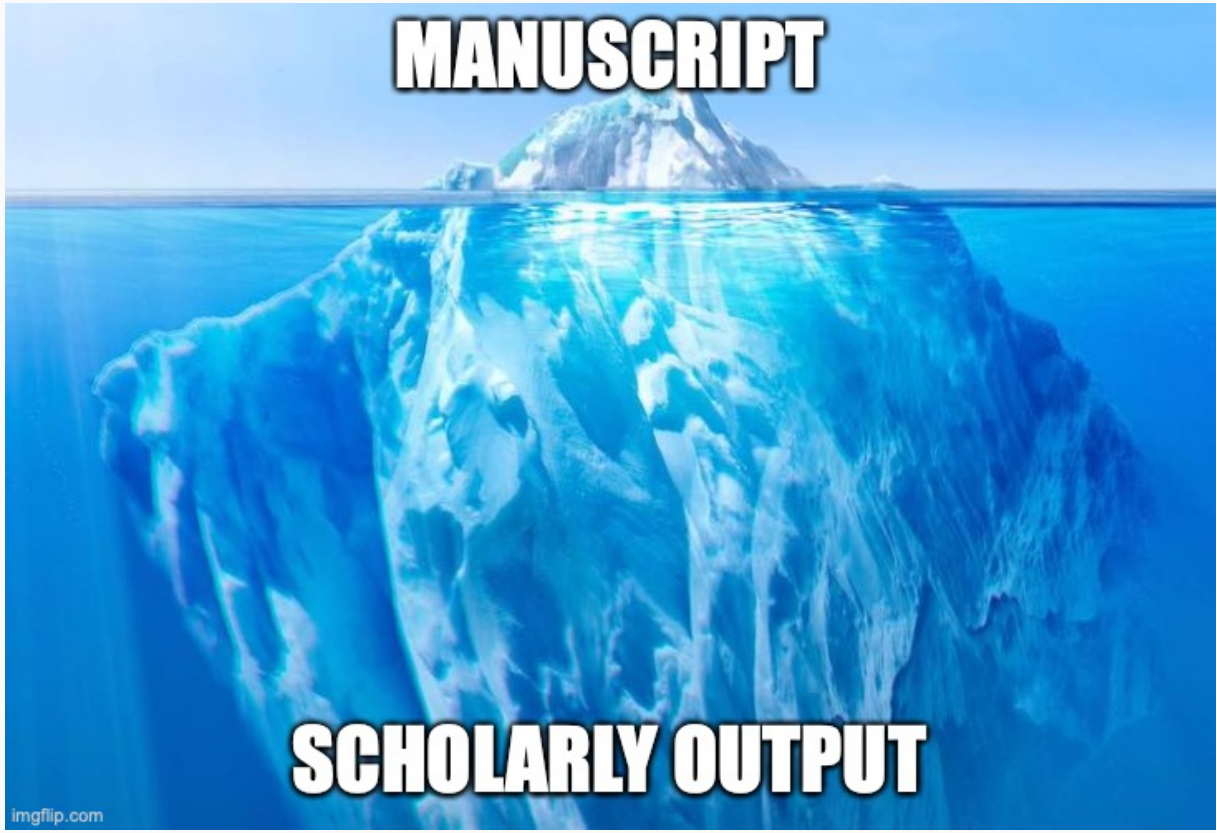
Motivation: Share The Entire Scholarly Output

Clarebout Principle:

"An article about computational science in a scientific publication is not the scholarship itself, it's merely scholarship advertisement. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."

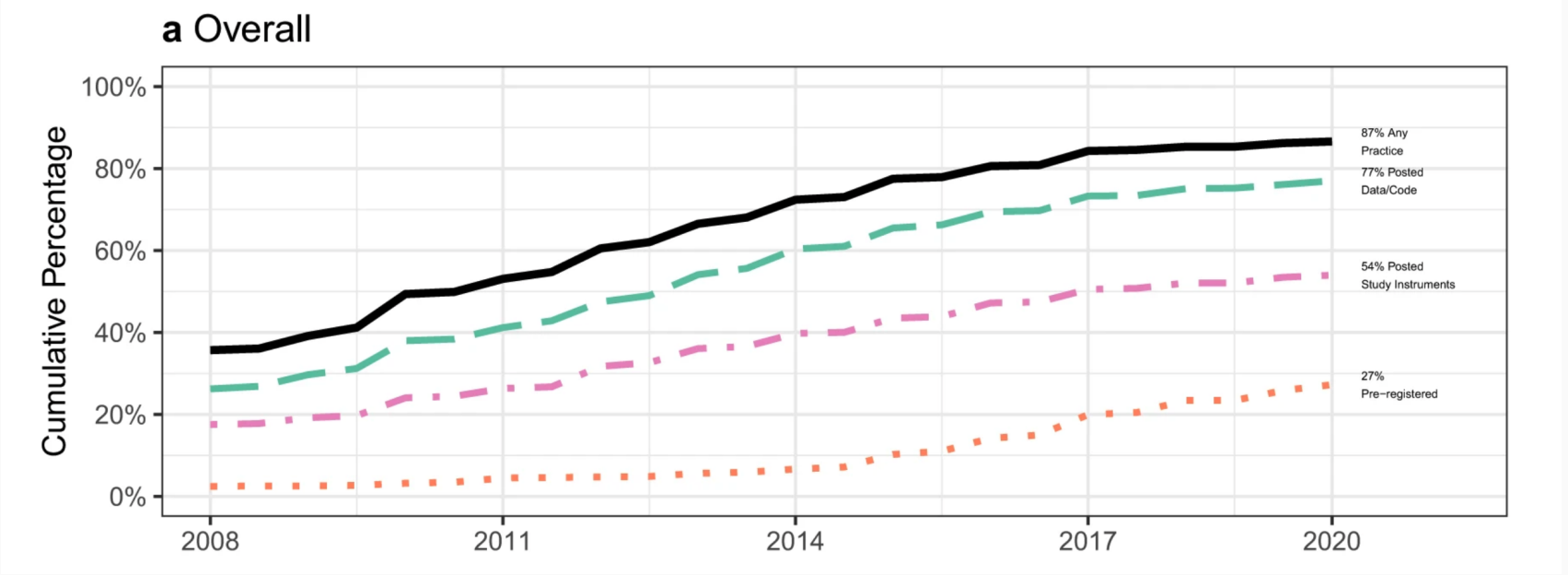
Buckheit and D.L. Donoho (1995, 2009)

Relates to *Communality*, and *Organized Skepticism*, as discussed in Ted Miguel's presentation



Data Sharing: Past

- Distant past: In economics data was rarely available until the early 2000's (Dewalt et. al. 1986, McCullough et al. 2006, Vilhuber 2020)
- Not so distant past: But even after the first policies were put in place most data sets where incomplete (Gertler et. al. 2018)
- However a clear change in norms has taken place over the last decade (Ferguson et. al. 2023):



Data Sharing: Present

- Founders are playing a key role by requiring data availability
  - Latest milestone: Nelson Memo
- Ever-growing infrastructure with solutions. For example:
  - Dataverse
  - Zenodo
  - Dryad (larger datasets)
  - OSF
  - ICPSR
  - See here for a comparison and more alternatives of trusted data repositories

Data Sharing: Future

- Rise of administrative data sets in economics
  - Recent revelations of high profile fraud (Harvard, Duke) where only possible because the data was available.
  - For a large fraction of research in economics, we have no way of detecting such irregularities.
- Data citations
  - Increasing the practice of citing data sets in addition to the paper could help align incentives even more.
- Data provenance
  - Tracing and documenting the origin of different data sets can serve two purposes: (i) share more effectively the knowledge of what it takes to build a given data sets, and (ii) give credit to the proper sources in the process of doing this.

Code Sharing: Git/Github for Non-Programmers

- Version Control Software (VCS) is an increasingly popular tool for computational reproducibility
- Git is a type of VCS and Github company that build on top of.
- They are very popular among programmers, but not so much among non-programmers.
- Why? I believe it has to do with GUIs.

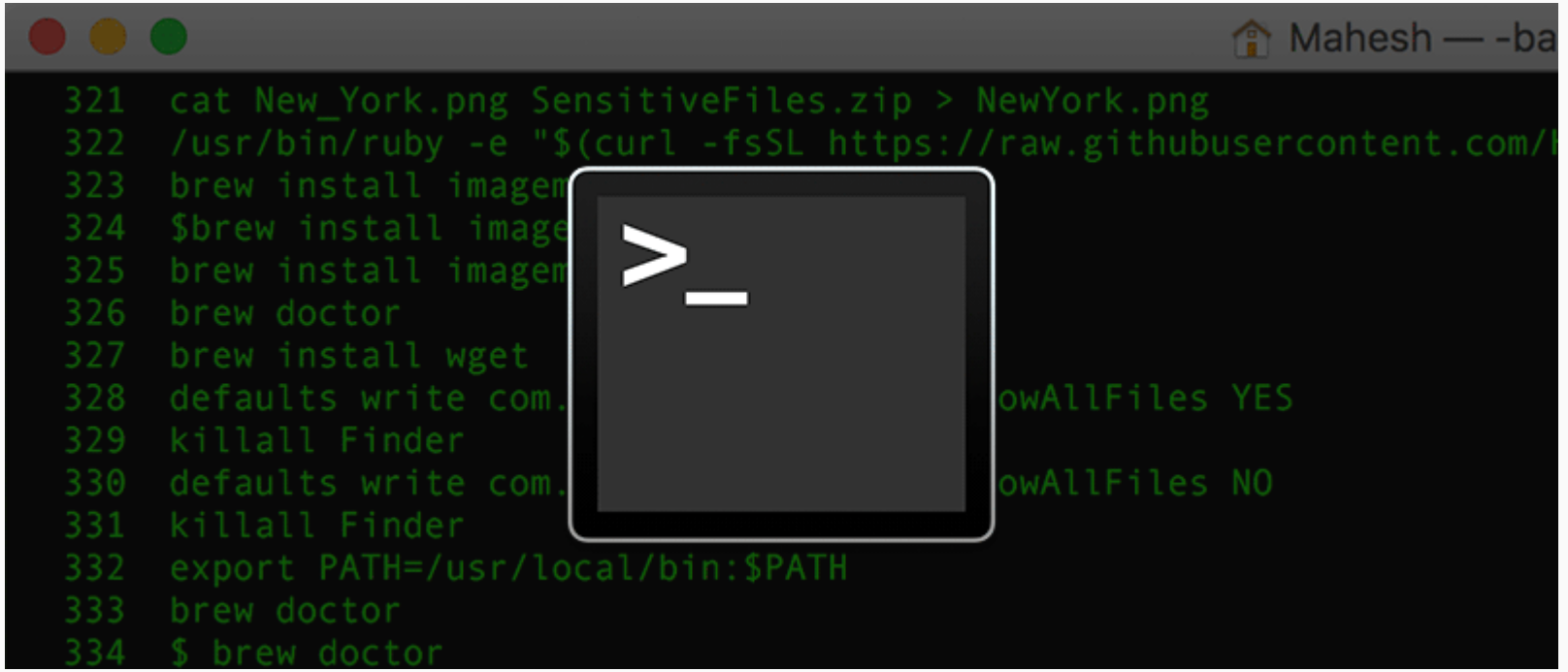
What is a GUI and why the bad reputation

Graphical **U**ser **I**nterface

- For most of us (non-programmers): *GUI = Software*.
- GUIs are behind the popularization of personal computers.
- Unfortunately GUIs are pretty bad at keeping a record of actions taken (bad for reproducibility).

What is not a GUI?

- Any software that is run in the command line (aka terminal, shell, bash, etc).



- Git was designed to run in the command line.

- Today we will learn Git **without** the command line.

What is Git 1/2

- Git is a software designed to track the **entire** history of the code of a project.
- Designed originally for software development, it has gained important traction in the research community.
- Main appeal: facilitates full reproducibility and collaboration.
- Git is mainly meant to work as a non-GUI (in the command line) software.
  However: most of the key features can be used through a GUI.

What is Git 2/2

- By code Git understands any type of plain text file (`myfile.R`, `myfile.do`, `.tex/.md/.txt/.csv/.etc`).
- This type of file can be understood as "human readable" as machine and human see the same fie.
- Files that are "non-human readable" are called binary files (`myfile.docx`, `myfile.xlxs`, `.pdf/.exe/.dta/.etc`).
- Git can also detect changes in binary files, but it cannot show those changes.

What is Github

- Github is a company that provides two services (that we care of):
  - A web hosting service for all our files track with Git (public free/private $ or free if academic).
  - A GUI software (Desktop App) that provides user friendly access to Git.
- Others hosting ss include: Bitbucket, GitLab, Gitkraken, etc.
- Other GUIs include: SourceTree, Gitkraken, VS Code, RStudio.

The Primary Goal of Version Control (for us)

The Goal: keep track of any *potentially meaningful* modification to your code.

Secondary Goal: learn how to collaborate with others using Github.

Strategy 1:

1. Agree on a naming convention with your co-authors (eg: YYYYMMDDfilename_INITALS).

2. Begin working from the last saved version (eg: `20180325demo_FH.do`).

3. At the end of the day, save on a new version (eg: `20180327demo_FH.do`).

Pros: Easy adoption.

Cons: Error prone, hard to document, lots of files for each document.

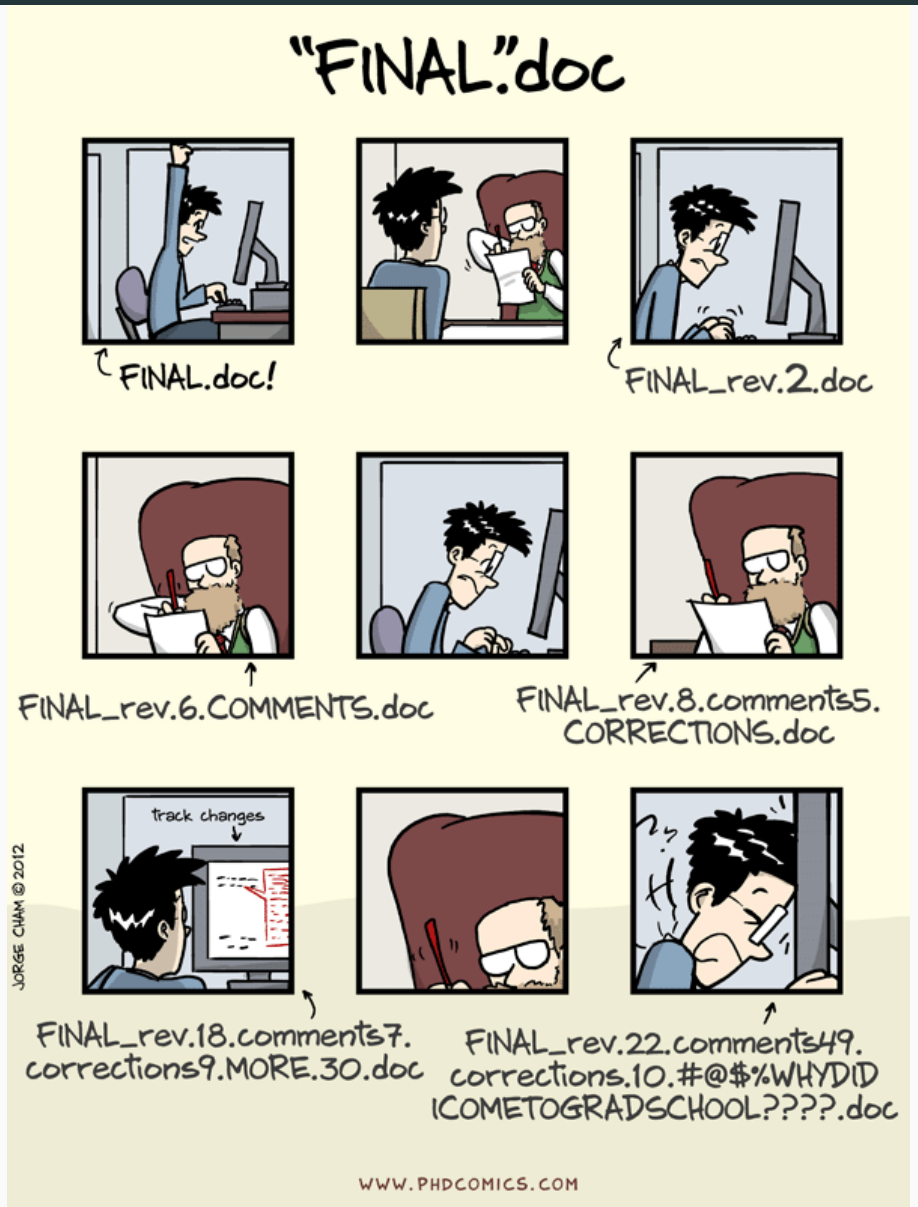
Strategy 2:

1. Name your file `filename` (ideally `01_filename`)
2. Take a snapshot of your work every time you complete relevant change (day, hour or minutes).
3. Update your entire working folder to the cloud.

Pros: Error proof, seamless documentation, one file per document, track differences across all versions, meant to work with the cloud.

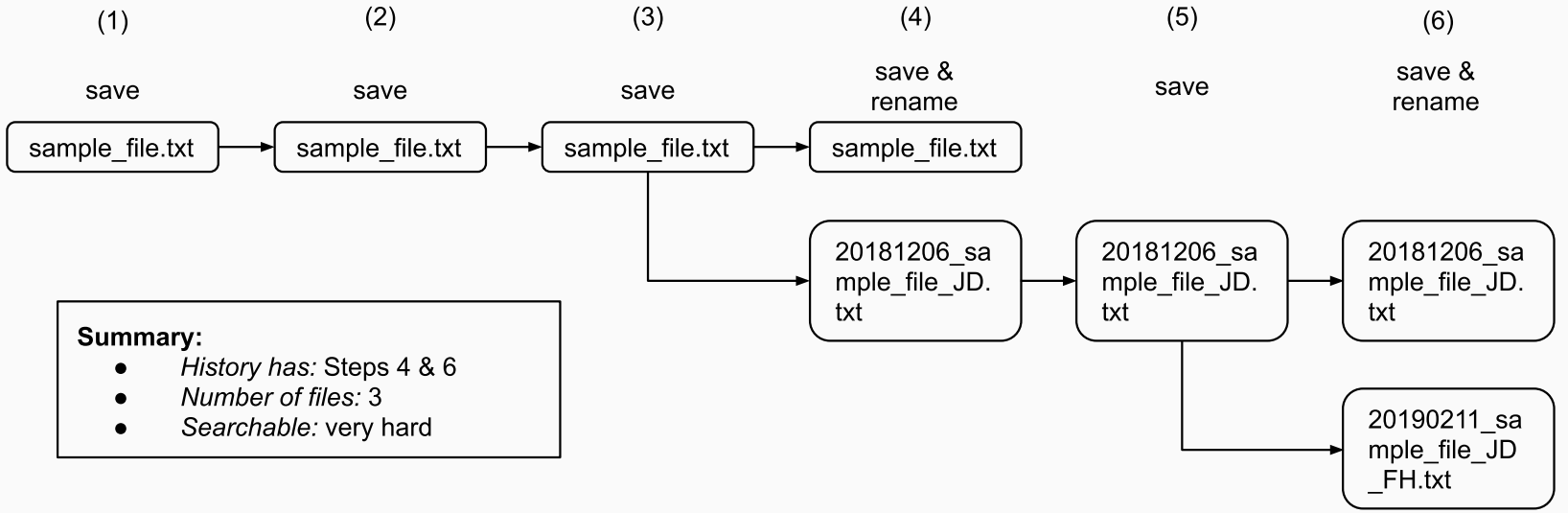
Cons: Harder adoption.

We want to avoid this situation:



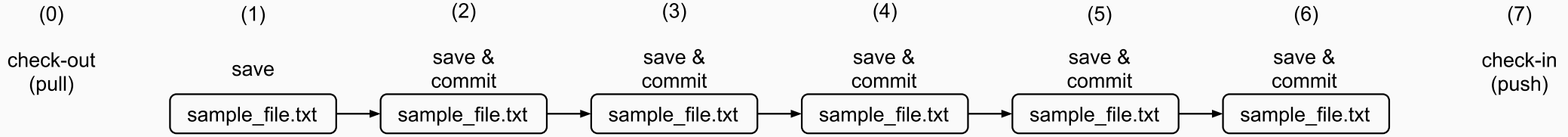
Comparison of Workflows

Strategy 1: Renaming



Summary:
- *History has:* Steps 4 & 6
- *Number of files:* 3
- *Searchable:* very hard

Strategy 2: Version Control Software



Summary:
- *History has*: Steps 2,3,4,5,6
- *Number of files*: 1
- *Searchable*: very easy

Other reasons to use Git

- To access a whole new world of knowledge!
- Great tool for collaboration.
- Easier to test all sorts of ideas/models.

Demos

Three Demos:

1 - **Simple but instructive.**

2 - Repeat with collaboration: shared ownership.

3 - Repeat with branches.

Demo #1: We Start in the Cloud

1 - Create github.com account and sign in.

2 - Let's look at some **repos:**

- Papers: Labor, Health, Public Finance
- Courses: So, many, methods, courses
- Covid: JHU, Imperial, NYT, The Economist, EconTracker
- and more: nice diagrams, meta-guides, books, guaguas!

3 - First way to access content: download.

4 - What if you want to have your own copy of the repo? **Fork** it!

Demo #1: We move to our local computer

5 - Now create your own repo. Initiate readme and make some edits.

6 - **Clone** the repo. Explore the files and location.

7 - Create new files, edit. And **commit**. Edit again, and commit again.

8 - **Push.** Edit on github.com, and **pull.**

9 - For this tutorial, best way to access previous version: explore in github.com and download.

here is some help with the lingo

Three Demos 2/3:

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete, search repo, download old version.

2 - **Repeat with collaboration: shared ownership.**

3 - Repeat with branches.

Two formats of collaboration

- Many owners, all can push.
  - **Very** important to pull at the beginning and at before each push.
- One owner, many pull requests (not covered here).
  - Easier to control, requires constant updating of forks.

Demo #2: Many owners

1 - Let's pretend that each of you represents two individuals: one working on your computer, and an alter-ego working on a different computer.

2 - Only for the purpose of this tutorial, we will edit the content of a repo on GitHub.com, as-if it was done by your alter-ego in a different computer.

3 - You-you: edits, commits and **does not push.**

4 - Alter-You commits and pushes in **different lines** (in GitHub.com).

5 - Now you-you (the one on your local machine) pushes.

6 - Repeat 3 - 5 but now both of you in the same lines.

Fatal Error!

Burn it and start with a fresh copy!

Jenny Bryan's Advice

Three Demos: 2/3

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete, search repo, download old version.

2 - Repeat with collaboration: shared ownership.
Review: Simulate collaboration, resolve conflicts.

3 - Repeat with branches.

Demo #3: Branches and collaboration (we wil be here a while)

1 - Create a branch from previous repo.

2 - Add new content (do not replace), commit a few times, and go back and forth to the main branch.

3 - Go back to main branch (master), observe file, merge.

4 - Look at the history of the main branch.

5 - Push and inspect visual tool from GitHub

6 - Repeat 1-3 but now replace instead of adding content.

Three Demos: 3/3

1 - Simple but instructive.
Review: def repo, github.com, download, clone, destination folder, fork, create repo, commit, push, pull, delete, search repo, download old version.

2 - Repeat with collaboration: shared ownership.
Review: Simulate collaboration, resolve conflicts.

3 - Repeat with branches.
Review: All of the above, plus: branch, merge, resolve conflicts.

Now go and explore!

Some good habits:

- Commit often (<1hr)
- Always pull before you start a new session of work. Also good to pull before pushing.
- Think of your remote as the most important set of files. Get used to deleting things in your local machine.

Want to Learn More: Version Control

Tutorials

- Great 20 min intro to Git by Alice Bartlett
- Great 2hr tutorial to Github by Jenny Bryan (git ninja)
- Software Carpentry's step-by-step tutorial (command line).

Documentation

- Jenny Bryan's Happy Git
- Documentation from Matthew Gentzkow and Jesse Shapiro
- Karthik Ram's paper on Git for Research

Economists Doing Highly Reproducible Work[1]

People

- Nick Huntingon
- Shoshana Vasserman
- Lars Vilhuber
- Grant McDermott
- Tyler Ransom
- Ed Rubin
- Luiza Andrade
- Max Kasy
- Matt Jensen
- Jason DeBacker
- John Horton
- Cora Kingdon
- Chandler Lester
- Alvaro Carril
- Andrew Heiss
- Lisa Rennels
- Michael Stepner
- Lachlan Deer
- Rebekah Din

Organizations

- LOST
- Opportunity Lab
- Congressional Budget Office
- Policy Simulation Library
- Gentzkow & Shapiro Lab
- Urban Institute

[1]: Non-exhaustive list of people and organizations doing amazing reproducible work on github (other than us!)