Version Control with the GitHub App and GitHub[1]

BITSS Research Transparency and Reproducibility Training (RT2)

Akash Shaji, BITSS
May 22, 2025

Before we begin

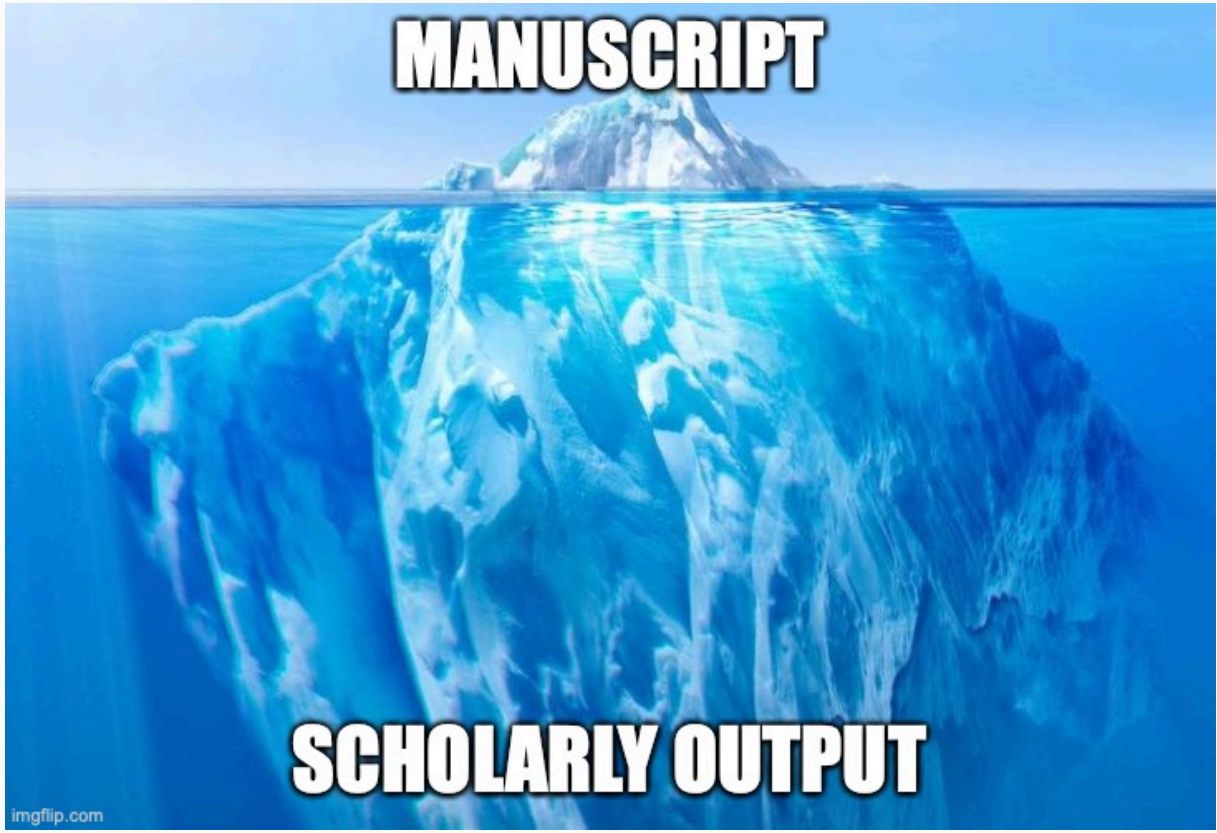
- GitHub desktop app installed?
- Create an account at GitHub.com?
- If no to any, please do it now

Motivation: Computational Reproducibility

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)



What is Version Control?

Version control (also called “revision control” or “source control”) is a system for tracking and managing changes to files—most commonly source code—over time. It lets multiple people collaborate, keeps a history of who changed what (and why), and enables you to revert or compare versions whenever needed.

Git/Github for Version Control

- Version Control Software 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, and gaining popularity in research.

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, Atom, 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.

Bonus track: get you excited about using open source statistical software (R, Python, Julia, etc)

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.

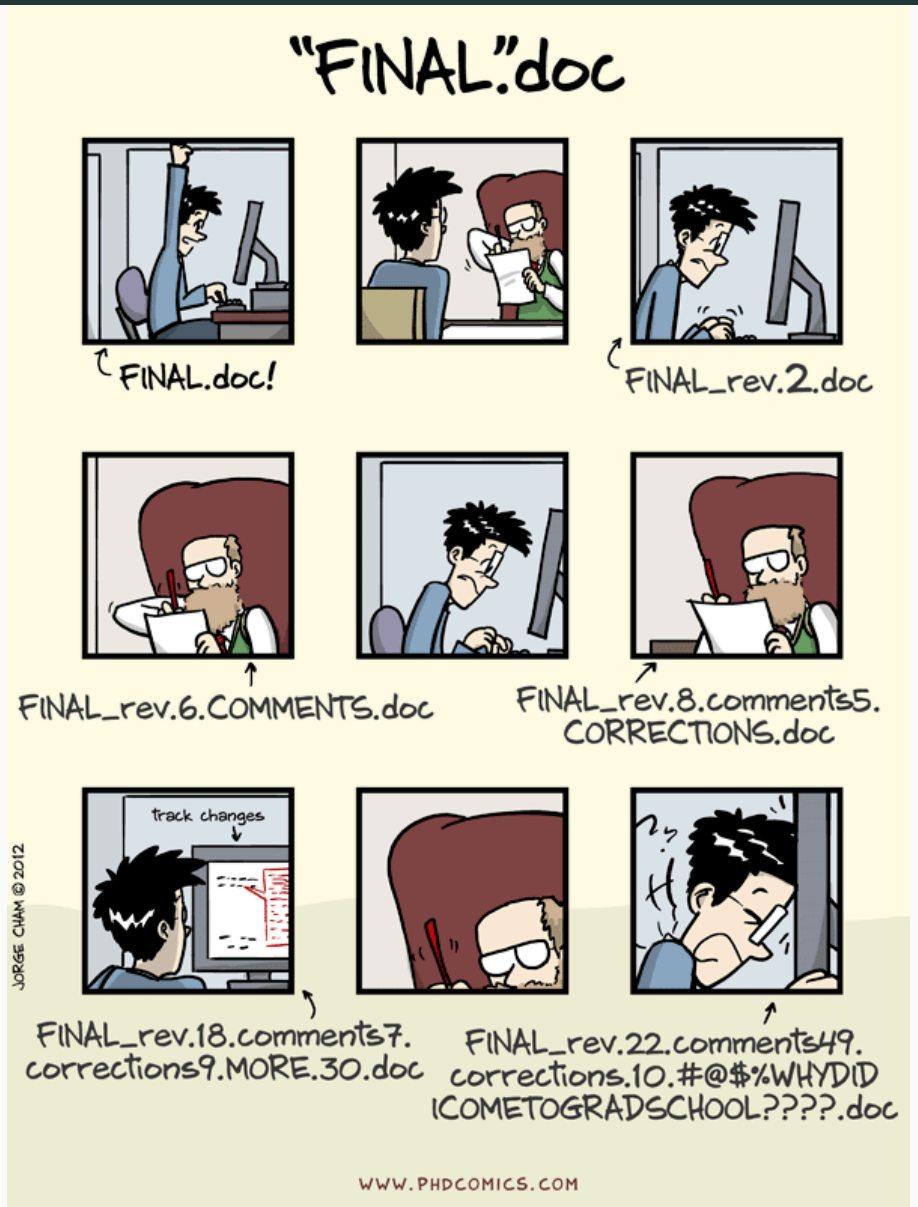
Strategy 2:

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.

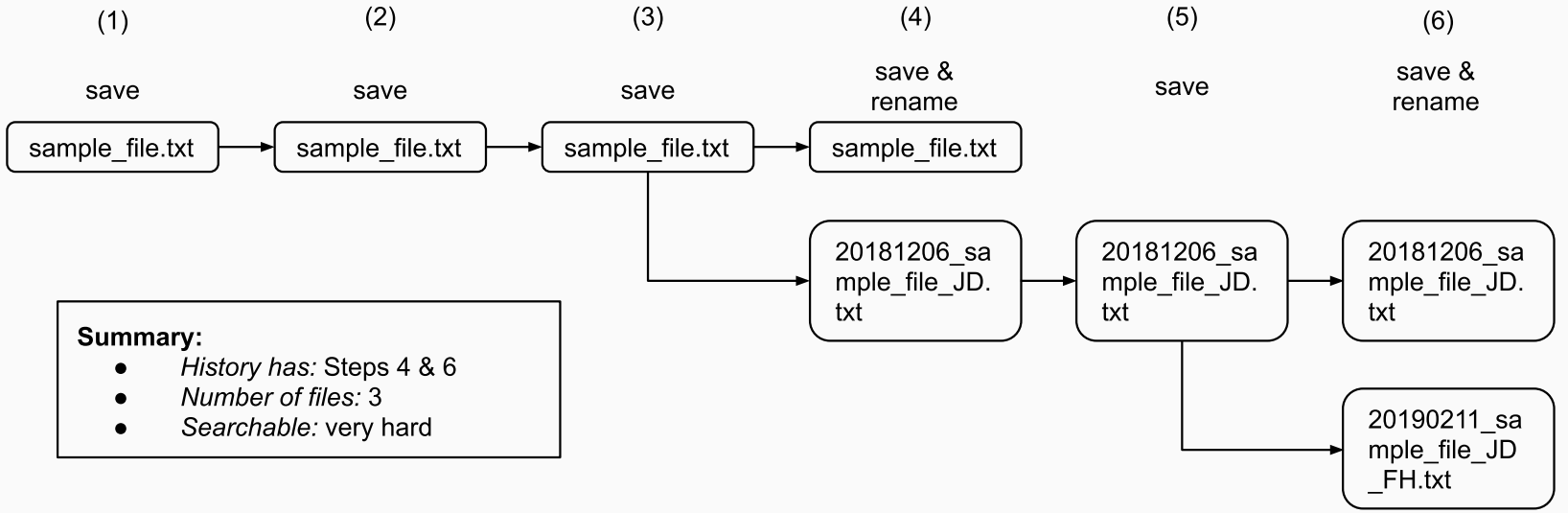
Cons: Harder adoption.

We want to avoid this situation:



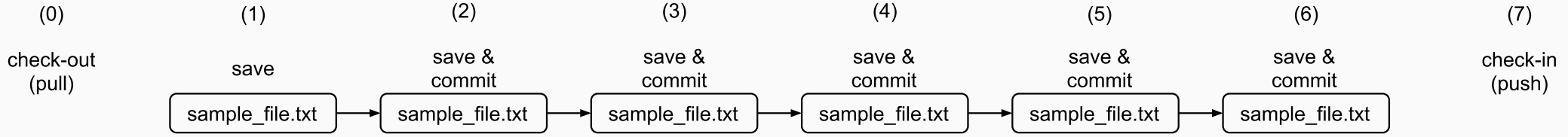
Comparison of Workflows

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

Hands on training:

1 - **Simple but instructive.**

2 - Repeat with collaboration.

4 - Explore a real life repo.

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

Demo

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**

3 - Explore a real life repo.

Two formats of collaboration

- One owner, many pull requests.
  - Easier to control, requires constant updating of forks.
- Many owners, all can push.
  - **Very** important to pull at the beginning and at before each push.

Demo #2: Pulling from the cloud

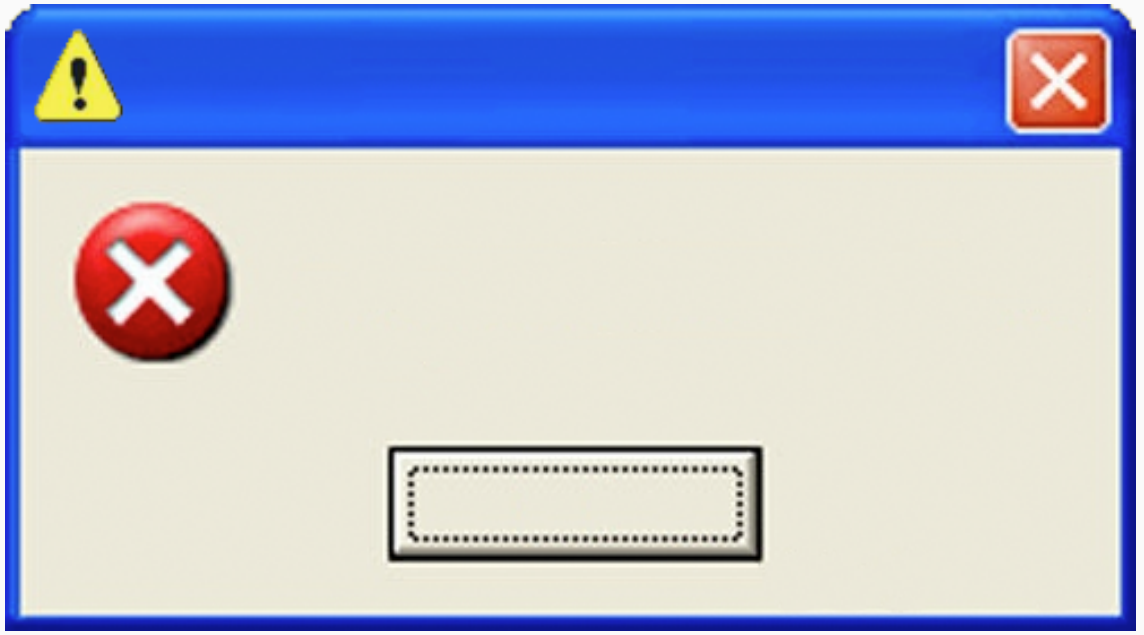
1 - Go back to our Demo Repo

2 - In the Github **web** make some changes (Assume that the change is made by a collaborator)

3 - Save, commit and push.

4 - **pull from the cloud**.

Fatal Error!



Burn it and start with a fresh copy!

Jenny Bryan's Advice

Demo #3: Look inside a real-life project (and collaborate!)

1- Find the following repo: `github.com/BITSS/opa-wealthtax`.

2- Fork it and clone it.

3- Open it in your computer: `opa-wealthtax.Rproj` (needs RStudio), look around and execute `code/dynamic_doc/wealth_tax_dd.Rmd`.

4- Find elasticities at `rawdata/edits/research.csv`, modify one elasticity, document, execute again.

5 - Find `code/interactive_visualization/server.R` and in line `1561` change `red` to `blue`

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!)