

Before we begin

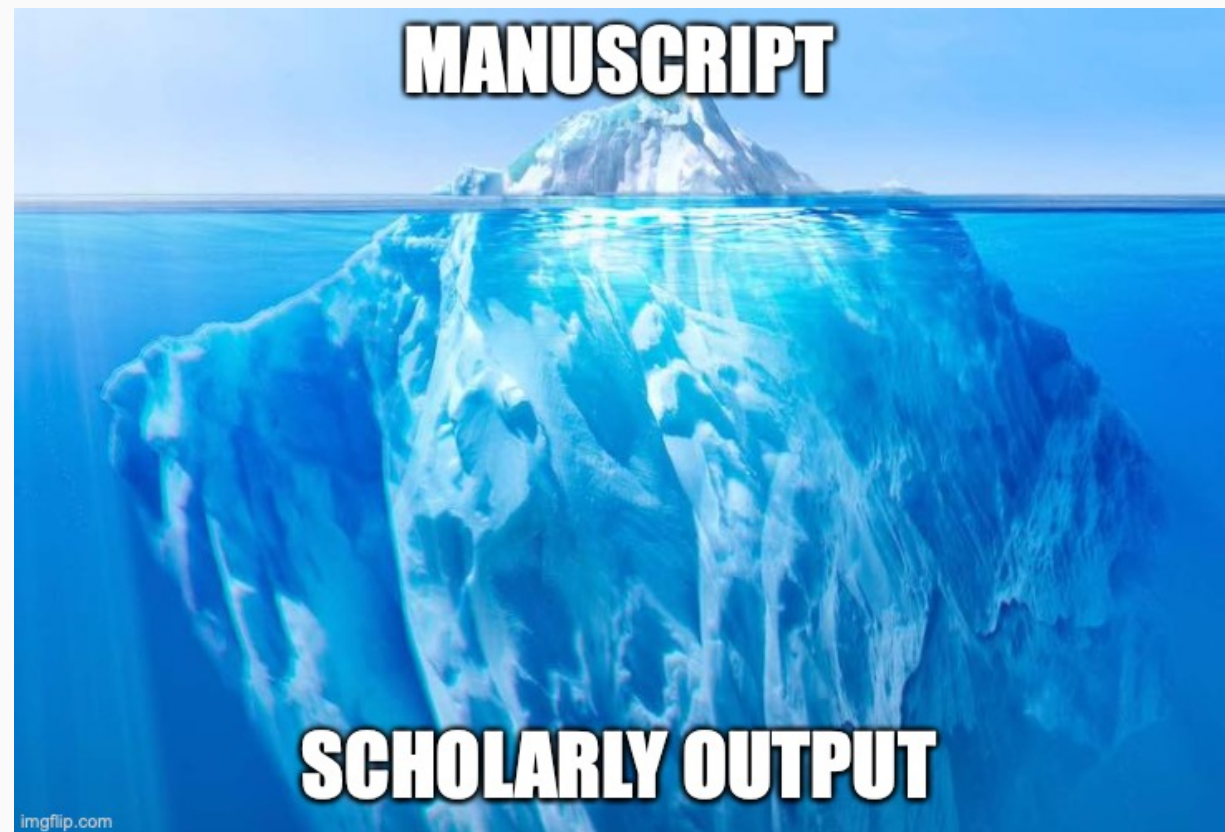
- [GitHub desktop app](#) installed?
- Create an account at [GitHub.com](#)?
- If no to any, please do it now (recommend to exit zoom temporarily to allow for internet bandwidth)

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)



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, but not so much among non-programmers.
- Why? I believe it has to do with GUIs.

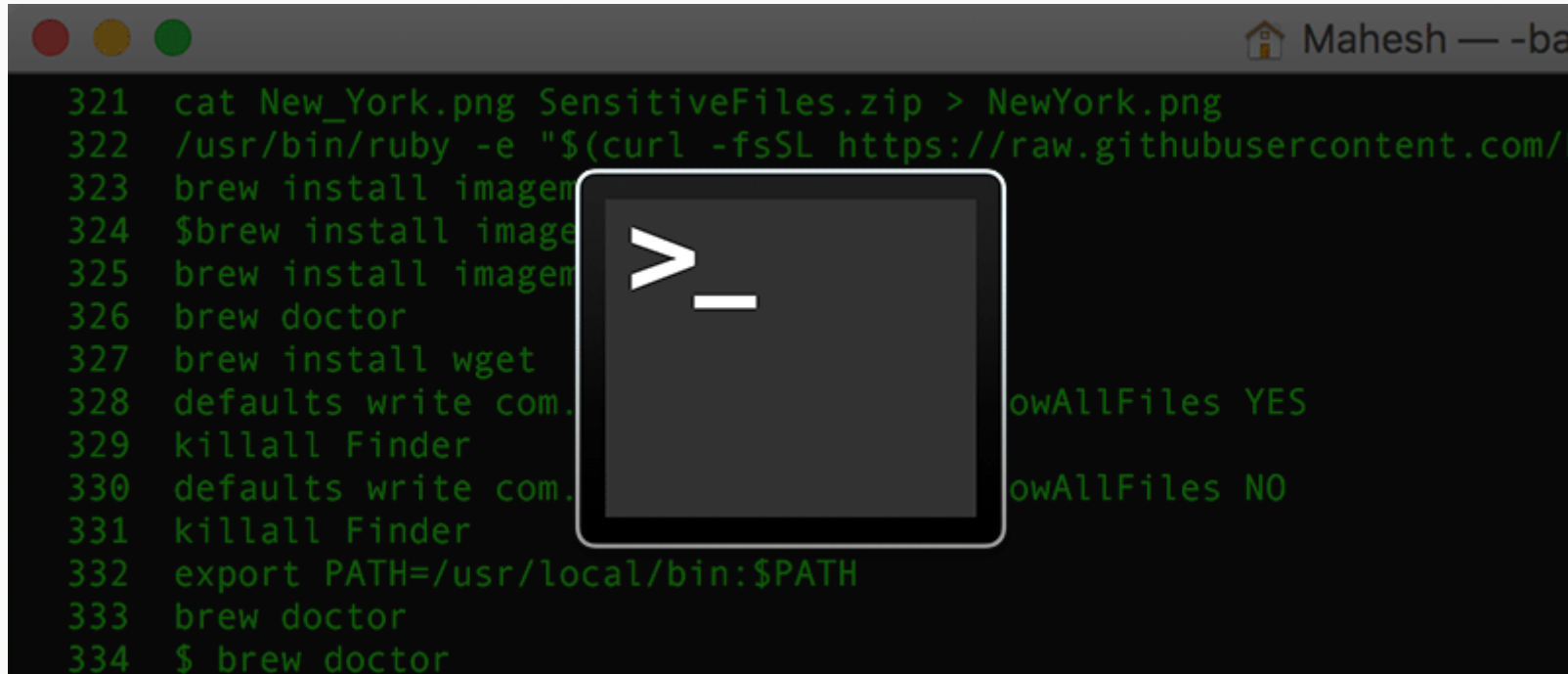
What is a GUI and why the bad reputation

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).



```
321 cat New_York.png SensitiveFiles.zip > NewYork.png
322 /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/l
323 brew install imagemagick
324 $brew install imagemagick
325 brew install imagemagick
326 brew doctor
327 brew install wget
328 defaults write com.apple.finder FXShowAllFiles YES
329 killall Finder
330 defaults write com.apple.finder FXShowAllFiles NO
331 killall Finder
332 export PATH=/usr/local/bin:$PATH
333 brew doctor
334 $ brew doctor
```

- 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 file.
- Files that are "non-human readable" are called binary files (`myfile.docx`, `myfile.xls`, `.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.

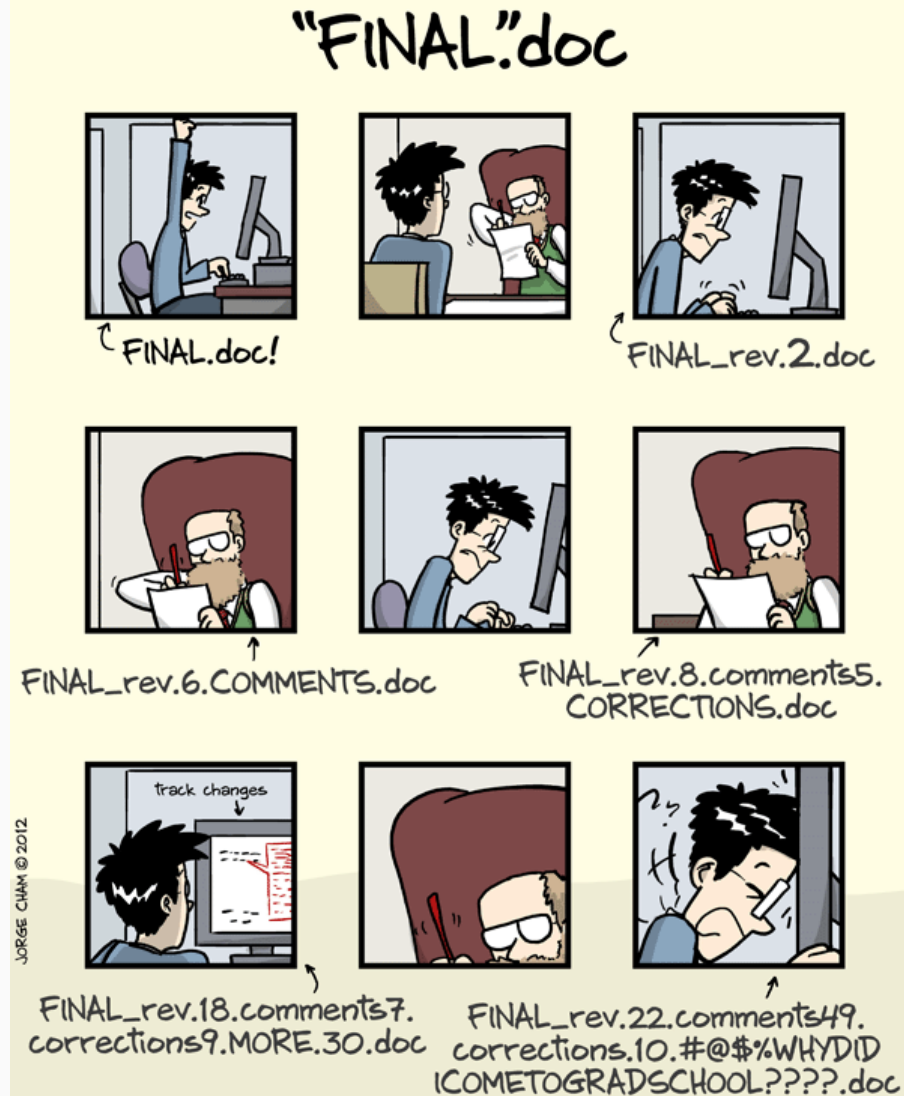
Strategy 2:

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.

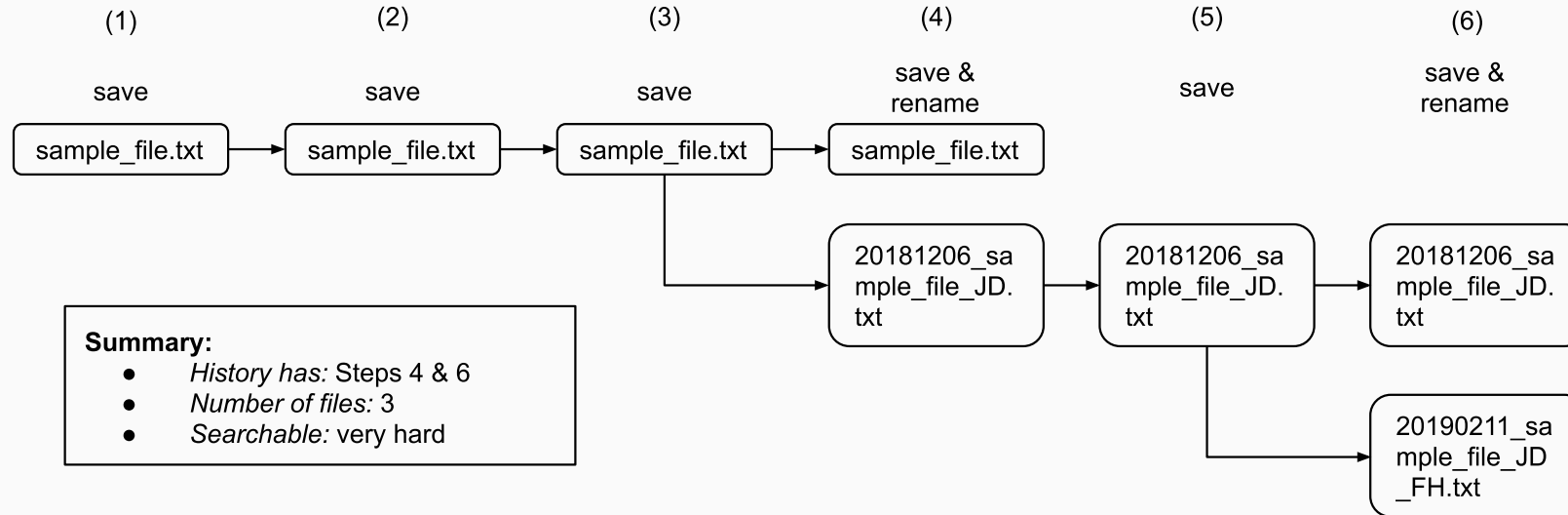
Cons: Harder adoption.

We want to avoid this situation:

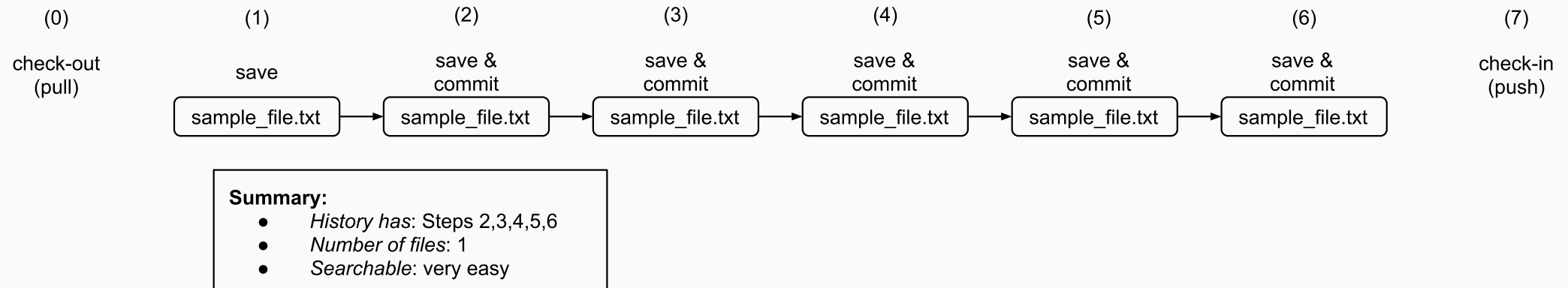


Comparison of Workflows

Strategy 1: Renaming



Strategy 2: Version Control Software



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

Five Demos:

- 1 - **Simple but instructive.**
- 2 - Repeat with branches.
- 3 - Repeat with collaboration: pull requests.
- 4 - Repeat with collaboration: shared ownership.
- 5 - 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.

Five Demos 2/5:

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 branches.**

3 - Repeat with collaboration: pull requests.

4 - Repeat with collaboration: shared ownership.

5 - Explore a real life repo.

Demo #2: 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 - Repeat 1-3 but now replace instead of adding content.

Fatal Error!



Burn it and start with a fresh copy!



Jenny Bryan's Advice

Five Demos: 3/5

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 branches.

Review: All of the above, plus: branch, merge, resolve conflicts.

3 - **Repeat with collaboration: pull requests.**

4 - Repeat with collaboration: shared ownership.

5 - 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 #3: Pull requests

- 1 - Fork repo github.com/BITSS/test_birthday, and clone it into your machine.
- 2 - Edit fields of name, and birth date.
- 3 - Save, commit and push.
- 4 - Create your first **pull request**.
- 5 - Let's see if I can manage all those pull requests very quickly (maybe illustrate issues).
- 6 - Now find your neighbors repo of Demos 1 & 2, fork it, clone it, make a change, save, commit, and...

Five Demos: 4/5

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 branches.

Review: All of the above, plus: branch, merge, resolve conflicts.

3 - Repeat with collaboration: pull requests.

Review: collaborate via fork + PR

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

5 - Explore a real life repo.

Demo #4: Many owners

- 1 - Half of you (#1): go back to the repo of demo 1 & 2 and invite a collaborator.
(Suggestion: the "forker" finds the repo, the "forkee" is invited, edits, commits, and push/pull)
- 2 - The other half (#2): clones, commits and pushes.
- 3 - #1 commits and pushes in **different lines**.
- 4 - Switch and repeat 2 & 3: #2 commits first and pushes, then #1.
- 5 - Repeat 2 - 4 but now both of you in the same lines.
- 6 - Repeat now but with branches (optional).

Five Demos: 5/5

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 branches.

Review: All of the above, plus: branch, merge, resolve conflicts.

3 - Repeat with collaboration: pull requests.

Review: collaborate via fork + PR

4 - Repeat with collaboration: shared ownership.

Review: collaborate via share ownership.

5 - **Explore a real life repo.**

Demo #5: 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`

Five Demos: 5/5

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 branches.

Review: All of the above, plus: branch, merge, resolve conflicts.

3 - Repeat with collaboration: pull requests.

Review: collaborate via fork + PR

4 - Repeat with collaboration: shared ownership.

Review: collaborate via share ownership.

5 - Explore a real life repo.

Review: All of the above, plus: how does a real-life example looks like.

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 Work¹

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!)