Version control with Git

Martin Constant

A cautionary note

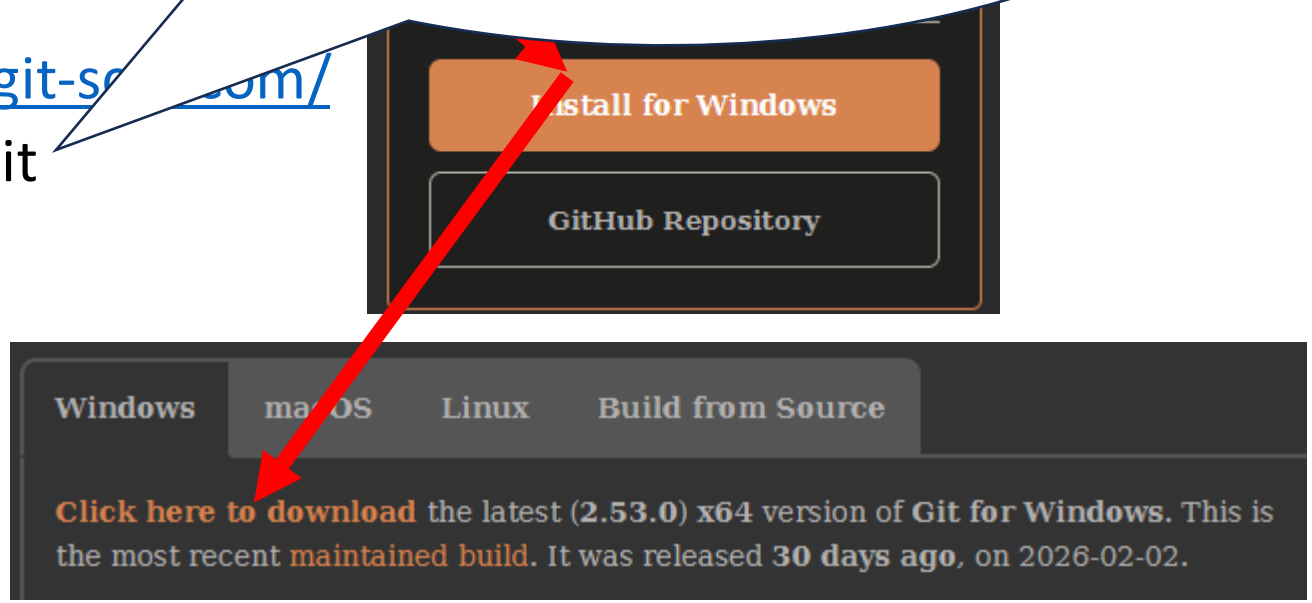
> My first instinct is to sell all my computers, fake my own death, move to another planet, and reinvent computing from scratch, rather than try to understand Git.
>
> I rarely actually do that, mind you. But the urge is there.

— Lars Wirzenius (Linux kernel developer)

Installing Git

- If you want to follow along.
- Windows:
  - Google : « Git » and go to https://git-sc[illegible]om/
  - Download the installer and install it
- MacOS, you need homebrew:
  - https://brew.sh/
  - Then, open terminal and:
  - >>> brew install git

- Select “Git from the command line and also from 3rd-party software”
- Enable file system caching
- Select “Use external OpenSSH”
- Enable symbolic links

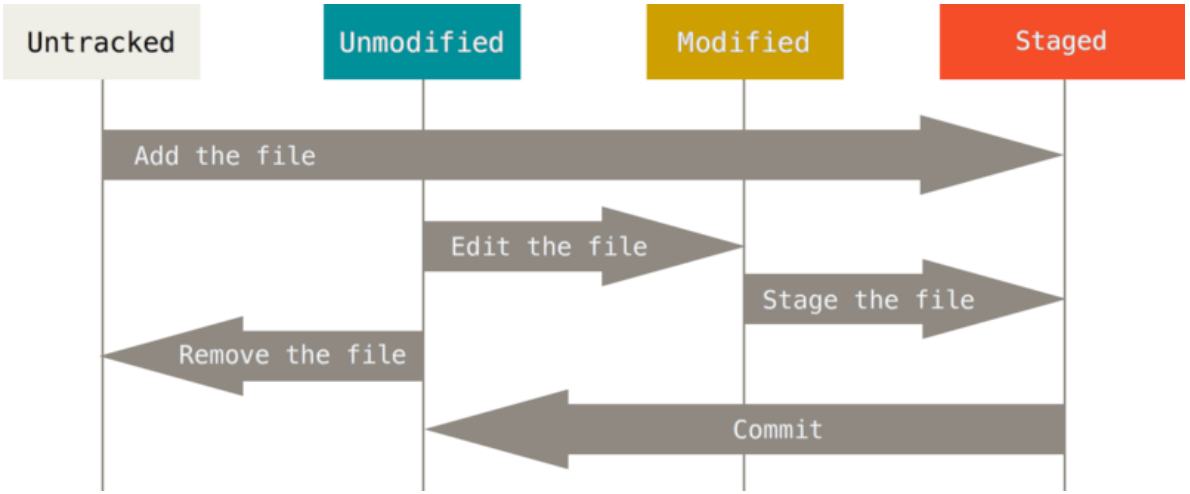


What is Git?

- Git is a **version control** tool.
- Git is **not** GitHub (more on that later).
- Git creates a hidden folder where changes are tracked.
- Git works best with text files (.txt, .csv, .py, .R, **but not .docx**)
  - But can track any file.

Git file states

- 4 different file states in git
  - Untracked: git knows nothing about this file's history
  - Unmodified: Tracked, umodified since the last commit
  - Modified: Tracked, modified since the last commit
  - Staged: Tracked, will be added to the next commit



Ok... And?

Why would I need git?

Version Control

- Have you ever tried to make a quick change to a piece of code, only to realize an hour later that you've broken everything?
- There's only so many times you can press UNDO before you go insane.
- You could try creating a copy of the file before you work on it.
- But that's how you end up with files called "doc_v1_final_amended_FINAL". We've all seen it. If you keep it up, you're left with a spider web of folders each containing almost identical files with almost identical names. Which one do you want?
- Git prevents this.
- Git keeps a record of every change made to every file in your codebase. There is only ever one copy of your file at a time. But if you need to, you can access every previous iteration of your files by rewinding your changes.

Credit: https://www.gitscientist.com/

Ok... And?

Why would I need git?

- If you write code, you can benefit from git.
  - You know which file(s) changed, how and when; and you can revert changes.
- If you share code, you can benefit from git.
  - Several git-based platforms to share code: GitHub, GitLab, git-forge, etc…
- If you review code, you can benefit from git.
  - Much easier to understand why your master's student code now doesn't work if you can easily see what they changed since the last time it worked.
- If you manage data, you can benefit from git.
  - DataLad is a tool built on git that can be used to manage (large) data.
  - https://www.datalad.org/

And that's why

Virtually every large code is managed by git

- PsychoPy: https://github.com/psychopy/psychopy
- BayesFactor (R library): https://github.com/richarddmorey/BayesFactor
- JASP: https://github.com/jasp-stats/jasp-desktop
- Praat: https://github.com/praat/praat.github.io

- But also… Analysis code for a project of mine: https://github.com/Martin2Constant/EEGManyLabs_Eimer1996
- Every Pavlovia experiment is managed by git (Pavlovia is "just" a fancy GitLab).

Git commands

- Git is, in essence, just a command line program:

  >>> git init is used to initialize the repository and creates a hidden .git folder

Top-most folder of your project

How to track files

- To tell git to start noticing a file:

  >>> git add yourFile

  - git add stages a file, making it tracked and ready to be committed.
- To commit (i.e., save) the changes:

  >>> git commit -m “Commit message”

  - Take all the changes you staged with “git add” and commit (save) them to the repository

Example – EEGML

Branches

- I want to implement a feature, but I’m afraid to mess up my main code.
- I can create a branch

*Until you want to merge it to main

Branches

- Once I’m satisfied with my feature, I can merge it into my main code. (main is actually just the default branch).
- Show sketch 1

Remotes a.k.a. “What the heck is GitHub?”

- To save my code in the cloud, I can upload to a “remote” location.
- It is usually initially private, but if I want to share it, I can add collaborators, or make it public.

- Show sketch 2

- GitHub is a platform to host remotes.
  - It is owned by Microsoft, so may not be suitable for more sensitive stuff.
  - You can enable the Education benefits (more features) https://github.com/education
- UNIGE also provides a GitLab (probably better suited for sensitive stuff):
  - https://gitlab.unige.ch/

Automating some steps

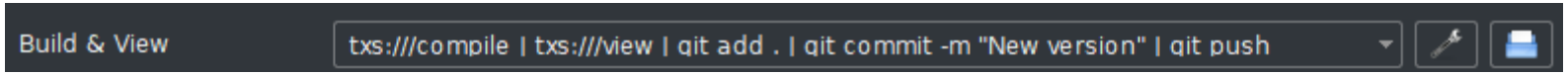
- You can automate some steps (though be mindful about how much you want to automate).
- At the end of a script:

```python
import os

os.system('git add Results')
os.system('git commit -m "New results"')
os.system('git push')
```

```
if IsLinux
    !git add Data
    !git commit -m "Add new participant"
    !git push
end
```

- When compiling your .tex file.



GUI – GitHub Desktop

- GitHub Desktop is a Graphical User Interface for git (despite its name, it can also be configured for other remotes such as GitLab).

- It can be useful, as long as you don’t forget the underlying mechanisms of git.

GitHub Desktop – Example

Datalad and GIN

- Hosting files on: https://gin.g-node.org/
  - German-hosted free git-based platform for hosting (neuroscience) data
  - No size limits
  - Private or public (with DOI if requested) repositories
  - e.g., https://doi.org/pmg5

- Properly configured, Datalad can be used to interface with GIN
  - https://www.datalad.org/