Versioning of data and code using Git

Introduction to Data Management Practices course

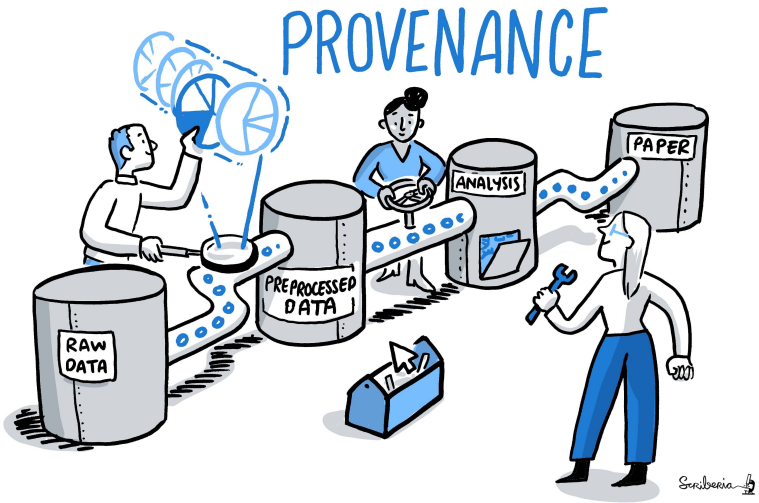
NBIS DM Team

data@nbis.se

https://nbisweden.github.io/module-versioning-dm-practices/

Outline

1. The fundamentals of versioning
2. Distributed versioning on the web using GitHub.com
3. Local versioning on your computer using GitHub Desktop
4. Collaborative versioning using GitHub.com and GitHub Desktop



Data storage and processing

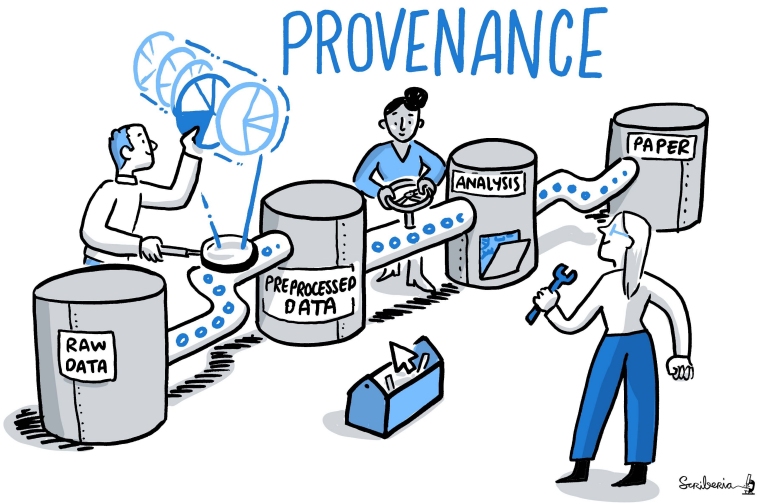
- Data has a life cycle

  Raw (experiment) data – produce, collect, license, get access, …

  Processed – generate, clean, aggregate, label, transform, analyse, …

  Archived – document, select, convert, package, submit, …

  Published – FAIRify, promote reuse, …
- Maintain data integrity and authenticity
- Plan a storage strategy
- Plan a backup and disaster recovery strategy



Data integrity and authenticity

- **Keep original (raw) versions** of data files, or keep documentation that allows the reconstruction of original files
- **Track the location of files** if they are stored in a variety of locations
- Establish **terms and conditions of data use** within the project team and beyond
- Keep a ‘master file’ of the data and take measures to preserve its authenticity
- Decide **how many and which versions to keep** for how long
- **Document changes** that were made in any version
- **Record relationships between items** where needed, for example between code and the data file it is run against

(UK Data Service, 2017a; Krejčí, 2014)

Local storage

- When working on different (local) workstations, e.g. laptop at home and the desktop in the office:
  - **always make sure that you are working on the most current version**, for example with the help of versioning software or guidelines
  - make sure that the most **current version is always backed up somewhere else**
- Only suitable as a primary storage for projects involving very few people
- Avoid if data will be moved back and forth between personal computers frequently

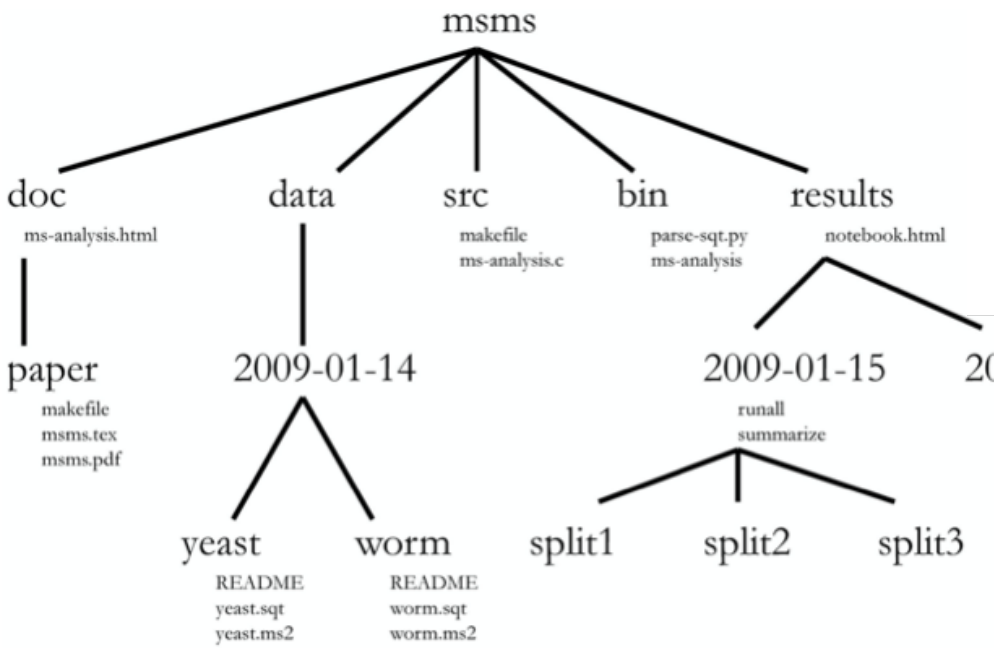
Cloud storage

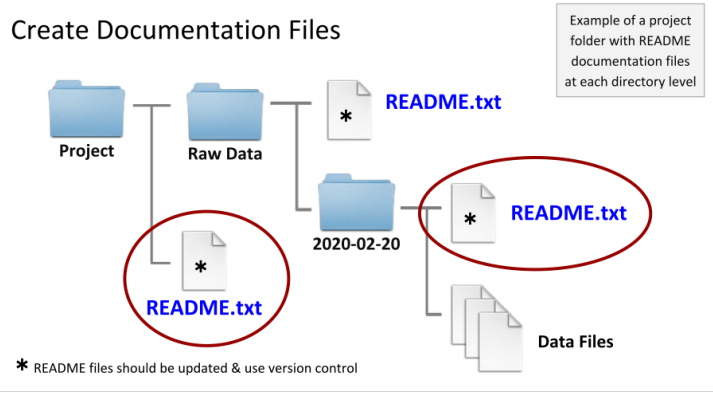
- Granting **shared, remote and easy access to data and other files** to all involved in the project
- **Read the terms of service**. Especially focus on rights to use content given to the service provider
- **Opt for European, national, or institutional** cloud services which store data in Europe if possible

- **Not your only storage and backup solution**
- **Not for unencrypted (sensitive) personal data**

Also be careful with passwords and other secrets!

Organising files and folders





Folder structure, docs and naming conventions are important!

Example from: Noble WS (2009) A Quick Guide to Organizing Computational Biology Projects. *PLoS Comput Biol* 5(7): e1000424. https://doi.org/10.1371/journal.pcbi.1000424

Fundamentals of versioning

- Snapshot projects and files

  Infinite undo, traceability, reproducibility

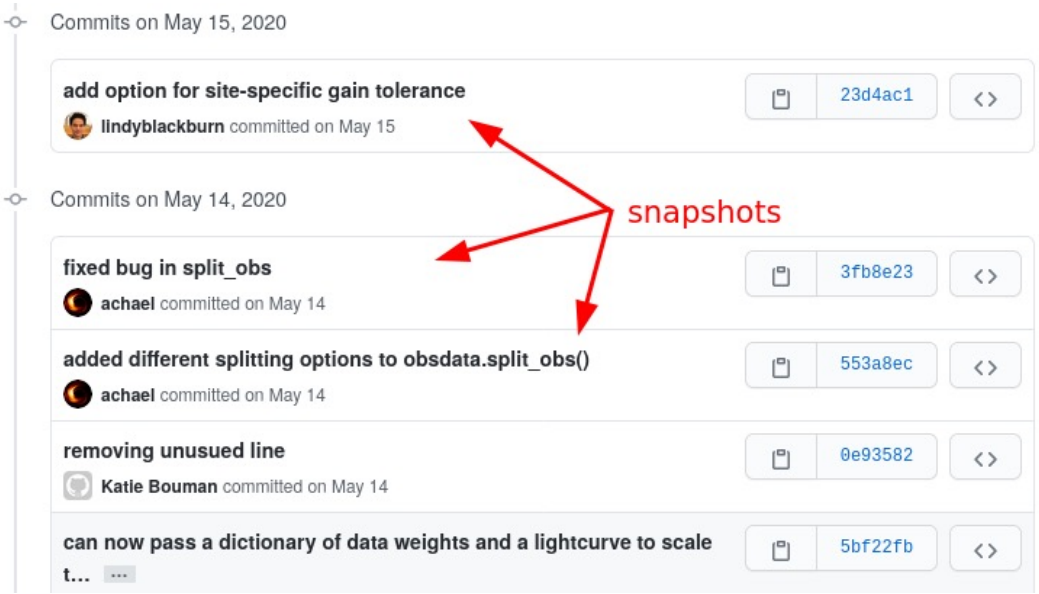
  Software, data, documents, scripts, manuscripts, …
- Copy–rename–describe

  Duplicate / rename files and folders

  Changes.txt
- Software assisted

  Projectplace, Google Docs, Sharepoint, Dropbox, but there is more…
- Collaborative versioning

  On your computer, on the web, from a single line of text to a complete project



https://stackoverflow.com/questions/1408450/why-should-i-use-version-control

Git is free and widely used

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

https://git-scm.com/

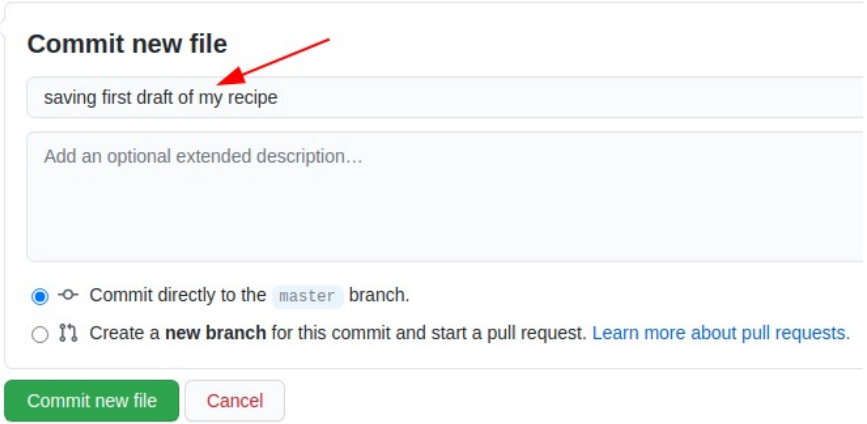
Git is just one component

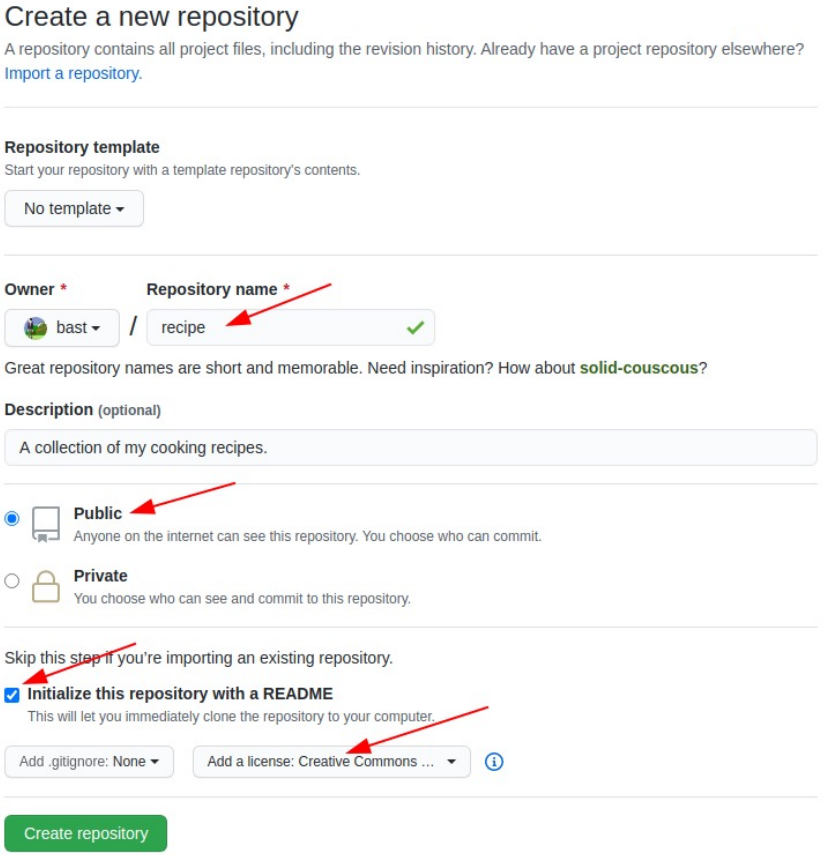
https://twitter.com/phdcomics/status/826861642507882496

https://twitter.com/phdcomics/status/826861642507882496

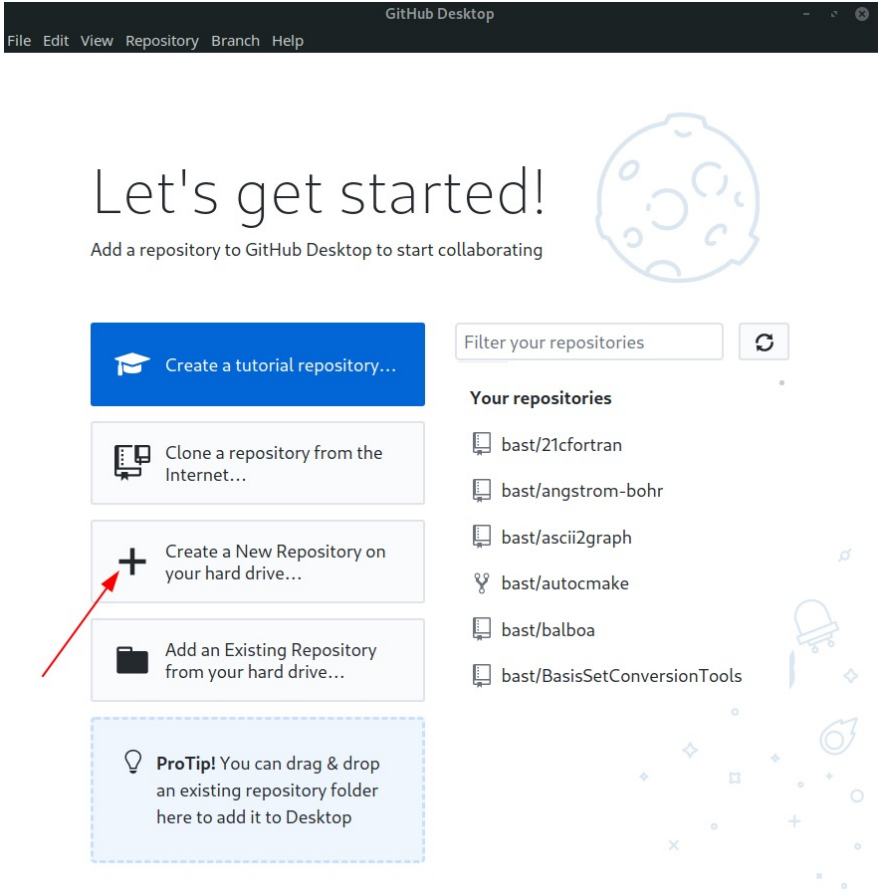
Versioning on the web

- **GitHub**, GitLab, Bitbucket, etc…
- Initialize a **repository** for storing versions of your files
- Create and **commit** a new file
- Edit and **commit** a changed file
- Make a new **branch** of history

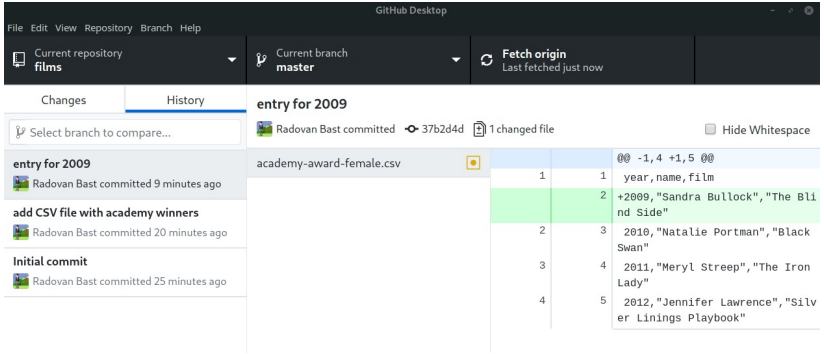




Versioning on a local PC

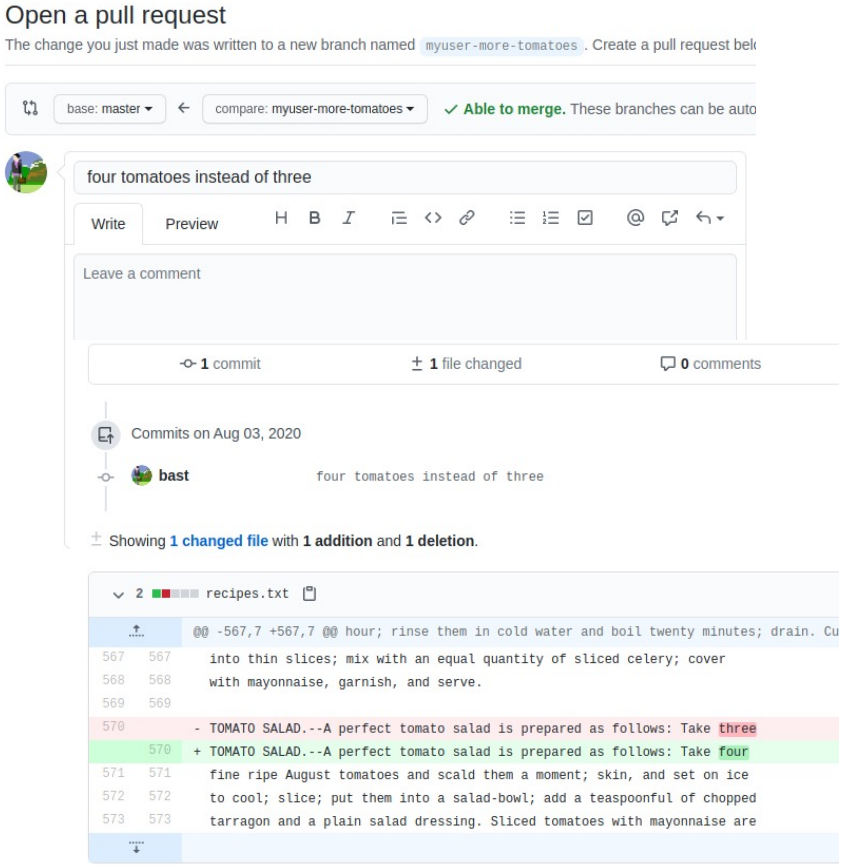


- Git, **GitHub Desktop**, Tower, …
- **Initialize** a repository for storing versions on your own computer
- **Add** files from your working directory to the **staging area**
- **Push** changes to the web
- **Pull** changes from the web



Collaborative versioning

- Curate a “**master**” branch of your project’s version history
- **Request** changes from one branch to be pulled into another on the web
  Merge changes from one branch into another
- **Resolve** conflicting changes



Managing parallel versions

- Not a replacement for back-ups
- Write informative commit messages
- Establish conventions for using and naming branches
- Best out-of-the box experience with text-based files (lines)
- Use tags / hashes to reference a specific revision
- Use data archives to preserve important revisions, e.g., Zenodo or Figshare