Research Software Engineering

DAY 1

Dr. Johanna Bayer

Dr. Johanna Bayer

Postdoctoral Fellow at the Donders Institute for Brain, Cognition and Behaviour, the Netherlands

DRA Trainer (focus on software)

Repronim Fellow

Editor at **JOSS**

The Carpentries instructor and maintainer

The Turing Way core contributor, maintainer and governance member

Datatalks.club podcast producer

Cat mom

Who are you and why are you here?

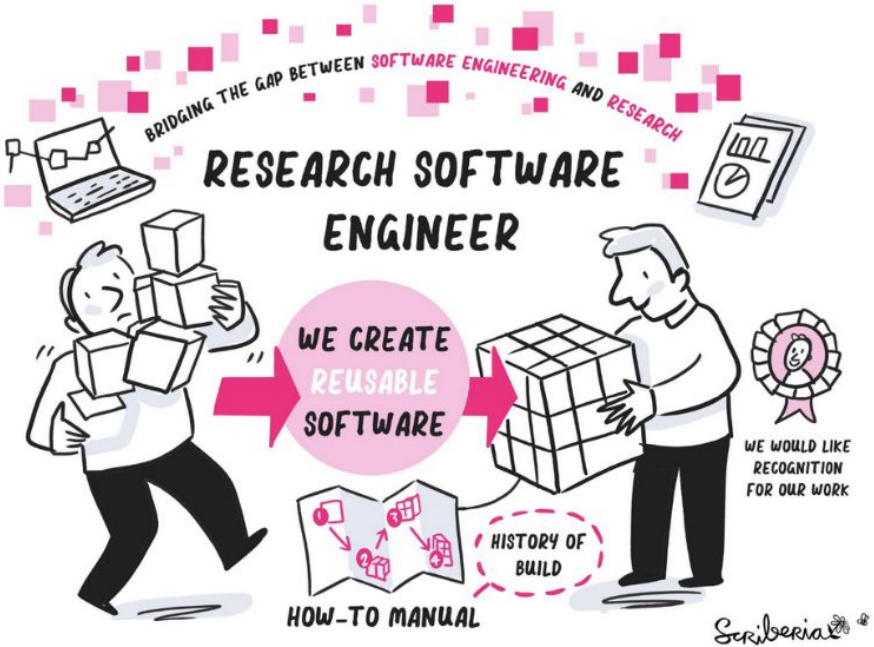
Outline

DAY 1				
Topic	**Time**			**Duration**
Welcome + Introduction: Why are we here?	9:00	-	9:15	0:15
What is a research software engineer? Setting up a research code project repository structure, naming, README **Activity 1 (20 min)**	9:15	-	10:05	0:50
Break	10:05	-	10:20	0:15
Activities with Git Theoretical Background (30 min) **Activity 2 (15 min)**	10:20	-	10:55	0:35
Break	10:55	-	11:10	0:15
Activity 3 (15 min) Theory and activities: collaborating effectively using issues, branches, forks, pull requests, merges and code reviews	11:10	-	11:45	0:35
Wrap-up	11:45	-	12:00	0:15

Discussion 1

What is a Research Software Engineer?

What is a Research Software Engineer?



What is a Research Software Engineer?

Definition is quite imprecise and notes from the original meeting where the term was coined (UK, 2012), include more a description of what it NOT is:

Roles: “research officer” – considered support don’t want to be servants

NOT permanent postdoc

Not a lab technician

Calling them software engineers is not good enough – domain specialised

“Researcher” has pressure to publish

https://www.software.ac.uk/blog/not-so-brief-history-research-software-engineers-0

What is a Research Software Engineer?

… t**he research software engineer**. These individuals combine a *professional attitude* to the exercise of software engineering with a *deep understanding of research topics*.

Computational work must reflect the committed attitude of experimentalists towards caring about precise, professional, repeatable, meticulous work – ***no-one with the same casual attitude to experimental instrumentation as many researchers have to code would be allowed anywhere near a lab.*** *This is striking considering how often research results now depend on software.*

Baxter et al. 2012

What is a Research Software Engineer?

F1000Research F1000Research 2024, 13:1429 Last updated: 27 NOV 2024

OPINION ARTICLE

Foundational Competencies and Responsibilities of a Research Software Engineer

[version 1; peer review: awaiting peer review]

Florian Goth[1], Renato Alves[2], Matthias Braun[3], Leyla Jael Castro[4], Gerasimos Chourdakis[5,6], Simon Christ[7], Jeremy Cohen[8], Stephan Druskat[9], Fredo Erxleben[10], Jean-Noël Grad[11], Magnus Hagdorn[12], Toby Hodges[13], Guido Juckeland[10], Dominic Kempf[14], Anna-Lena Lamprecht[15], Jan Linxweiler[16], Frank Löffler[17], Michele Martone[18], Moritz Schwarzmeier[19], Heidi Seibold[20], Jan Philipp Thiele[21,22], Harald von Waldow[23], Samantha Wittke[24]

What is a Research Software Engineer?

Training/teaching

Work in research environment

Produce software

Communicators/Translators

self-identify

Produce papers

Research Software Engineer

funding

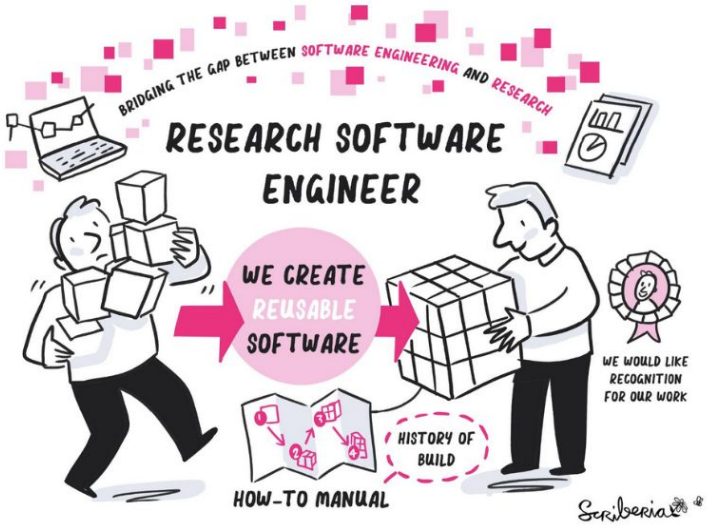
Research

Software Developer

Non-technical complexities of the research environment

What is Research Software

Foundational algorithms, the software itself, as well as scripts and computational workflows that were **created during the research process or for a research purpose**, across all domains of research.



Setting up a (research) code project

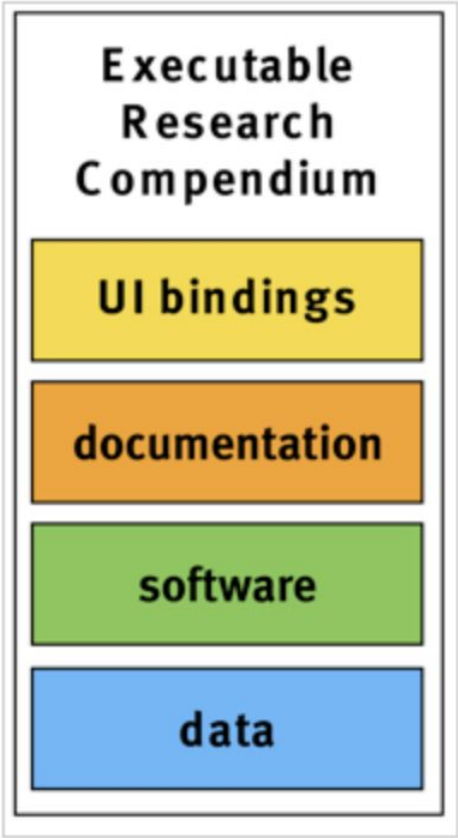
DIGITAL RESEARCH
ACADEMY

The Research compendium

The Research Compendium

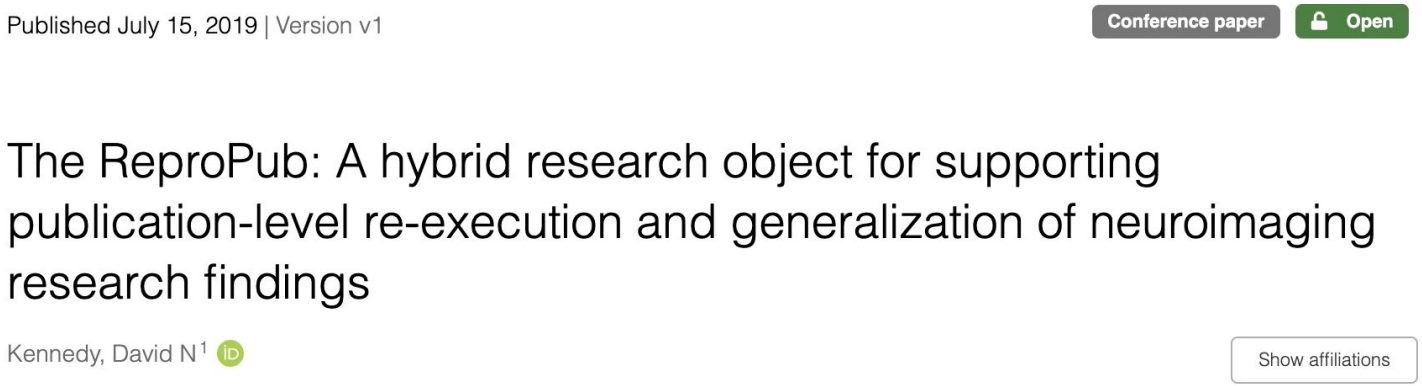
- A research compendium is a collection of all digital parts of a research project including data, code, texts (protocols, reports, questionnaires, meta data). The collection is created in such a way that reproducing all results is straightforward.

- A research compendium accompanies, enhances, or is a scientific publication providing data, code, and documentation for reproducing a scientific workflow.



Nuest, D., Konkol, M., Pebesma, E., Kray, C., Schutzeichel, M., Przibytzin, H., & Lorenz, J. (2017). Opening the Publication Process with Executable Research Compendia. *D-Lib Magazine*, *23*(1/2). 10.1045/january2017-nuest

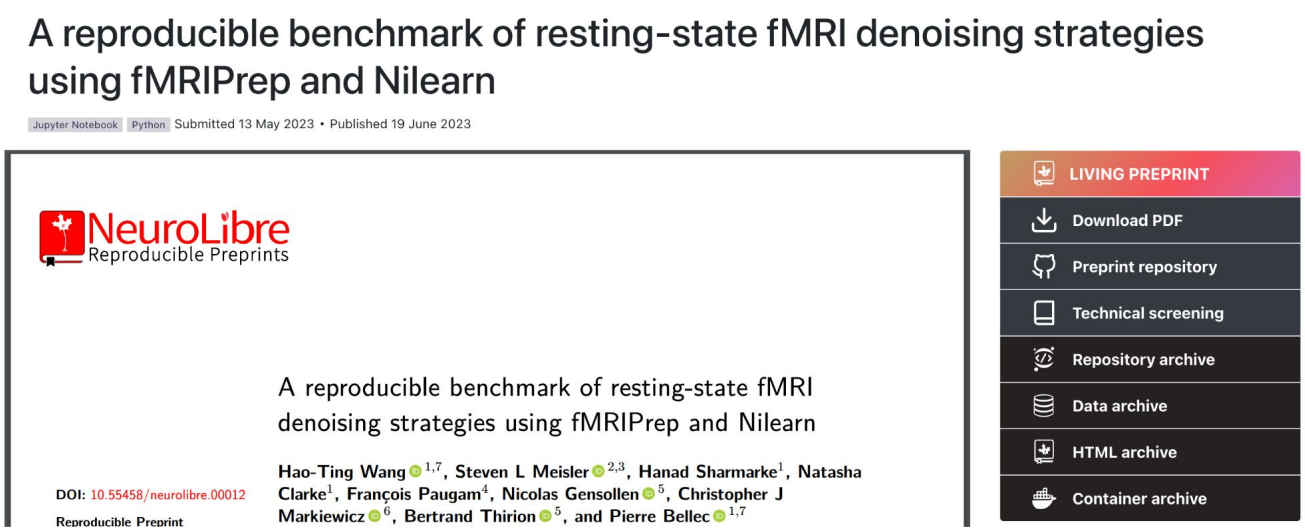
The research compendium/the repro pub



- Idea: create a workflow that re-creates a publication from raw data to pdf

Neurolibre

- FULLY reproducible paper
- Living preprint: reviewer comments are visible; interactive document

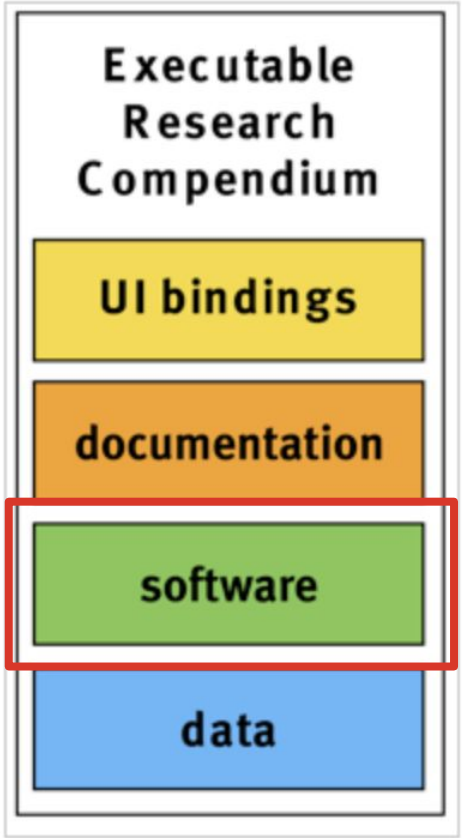


overview

The Research Compendium

A very comprehensive website on research compendia can be found here

An example template to document your research can be found here



Reproducible science includes a good organisation of the software/code

Setting up a code project - as part of a research compendium

The folder structure

```
compendium/
├── data
│   ├── my_data.csv
├── analysis
│   └── my_script.R
├── DESCRIPTION
└── README.md
```

```
compendium/
├── CITATION
├── code
│   ├── analyse_data.R
│   └── clean_data.R
├── data_clean
│   └── data_clean.csv
├── data_raw
│   ├── datapackage.json
│   └── data_raw.csv
├── Dockerfile
├── figures
│   └── flow_chart.jpeg
├── LICENSE
├── Makefile
├── paper.Rmd
└── README.md
```

project folder structure

https://the-turing-way.netlify.app/reproducible-research/compendia.html

Setting up a code project

```
my-research-project/
├── .vscode/           # vs code settings
│   ├── launch.json        # Cleaned, derived data
│   └── settings.json      # Data documentation
├── src/               # Code (modules, scripts)
├── notebooks/         # Jupyter or MATLAB Live Scripts
├── data/
│   ├── raw/           # Untouched raw data
│   ├── processed/     # Cleaned, derived data
│   └── README.md      # Data documentation
├── results/
│   ├── figures/       # Plots/images generated
│   └── tables/        # CSV, LaTeX tables, etc.
├── paper/
│   ├── manuscript.md     # Or .tex, .docx, etc.
│   ├── bibliography.bib  # References
│   └── figures/       # Final figures for the paper
├── tests/             # Automated tests
├── requirements.txt       # or environment.yml / setup.py
├── Makefile           # Automate workflows (optional)
├── LICENSE
├── README.md      # What is your project about?
└── .gitignore
```

project folder structure

Cookiecutter project template

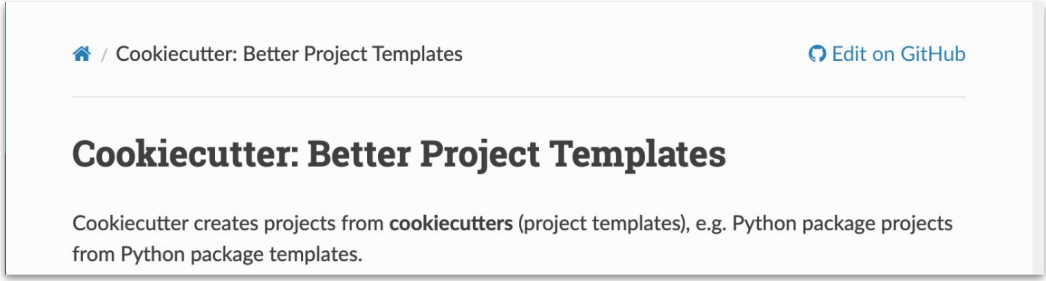
- Code and data version for good file organisation

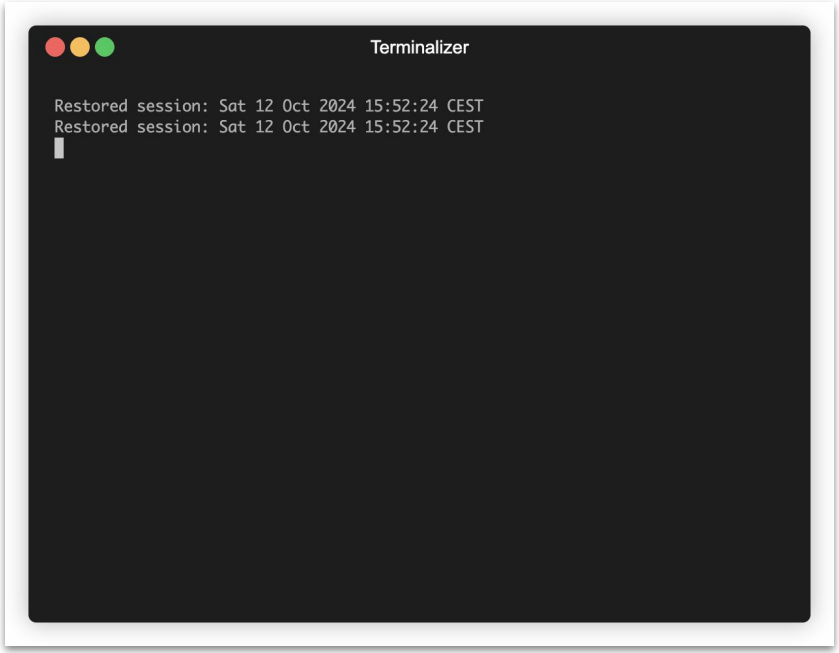
```
johannabayer@Hanna my_project % tree
.
├── LICENSE
├── Makefile
├── README.md
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── docs
├── models
├── my_project
│   ├── __init__.py
│   ├── config.py
│   ├── dataset.py
│   ├── features.py
│   ├── modeling
│   │   ├── __init__.py
│   │   ├── predict.py
│   │   └── train.py
│   └── plots.py
├── notebooks
├── pyproject.toml
├── references
├── reports
│   └── figures
├── requirements.txt
└── setup.cfg
```

project folder structure

Project folder structure

- Cookie cutter project templates
- Cookie cutter data science (ccds)
  - Python-based project structure





project folder structure

https://cookiecutter.readthedocs.io/en/stable/

Setting up a coding project - naming files

Files need to be

- Human readable
- Machine readable
- Consistent
- Descriptive
- Optional: Play with default ordering (i/e start your file with creation date `YYYY-MM-DD`).

Naming files

(BAD) Examples:

- Myabstract.docx
- Joe’s Filenames Use Spaces and Punctuation.xlsx
- figure 1.png
- fig 2.png
- JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt

Naming files

Better examples

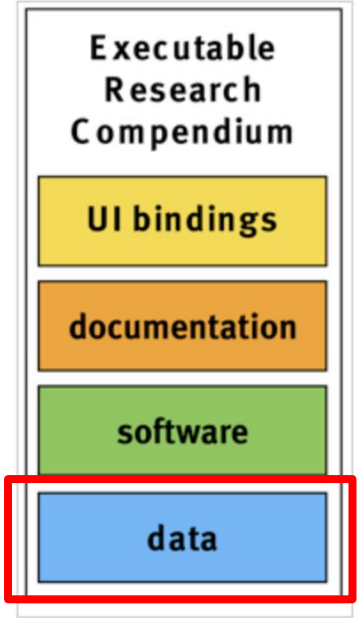
- YYYYMMDD_OHBM_abstract.txt
- Step_1_step_description
- Step_2_step_description
- Unique filenames:
  - Sub_1
    - sub_1_fileA.txt
    - Sub_1_fileB.txt
  - Sub_2
    - sub_2_fileA.txt
    - Sub_2_fileB.txt

Follow the software example: versioning

- Generic version is the CURRENT/IMPORTANT version (eg. OHBM_abstratc.txt)
- All older/other versions are versioned (e.g OHBM_abstract_Sept_2024.txt)
- Similar to JAVA/Git - > main method/master is the one that compiles, not main_2025, even if that's the latest version.

https://berd-nfdi.github.io/BERD-reproducible-research-course/2-1-naming.html

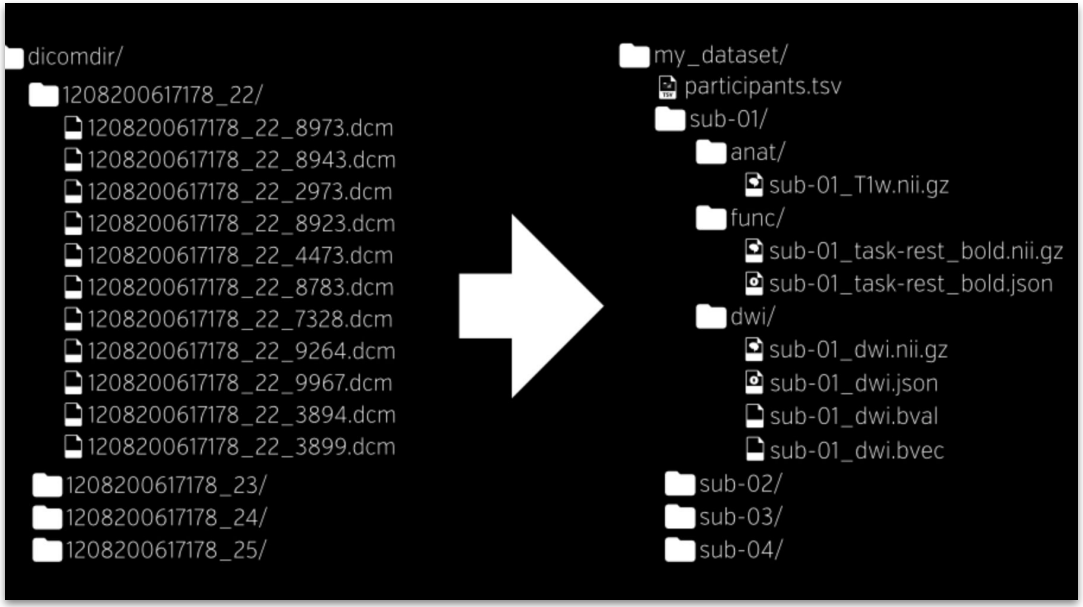
Data organisation - does your field have a file standard?




```
my-research-project/
├── .vscode/          # vs code settings
│   ├── launch.json       # Cleaned, derived data
│   └── settings.json     # Data documentation
├── src/              # Code (modules, scripts)
├── notebooks/        # Jupyter or MATLAB Live Scripts
├── data/
│   ├── raw/          # Untouched raw data
│   ├── processed/    # Cleaned, derived data
│   └── README.md     # Data documentation
├── results/
│   ├── figures/      # Plots/images generated
│   └── tables/       # CSV, LaTeX tables, etc.
├── paper/
│   ├── manuscript.md     # Or .tex, .docx, etc.
│   ├── bibliography.bib  # References
│   └── figures/      # Final figures for the paper
├── tests/            # Automated tests
├── requirements.txt      # or environment.yml / setup.py
├── Makefile          # Automate workflows (optional)
├── LICENSE
├── README.md         # What is your project about?
└── .gitignore
```

Does your field have a file standard?

Brain Imaging Derived Structures



standards (https://bids.neuroimaging.io/):

Slides by courtesy of Dr. Remi Gau, adapted

File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures

- modularizes data
- names files in a human AND machine friendly way
- specifies a folder structure
- uses standard interoperable file formats
- documents metadata
- minimize duplication

Slides by courtesy of Dr. Remi Gau, adapted

File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures

Key1-value_key2-value_suffix.extension

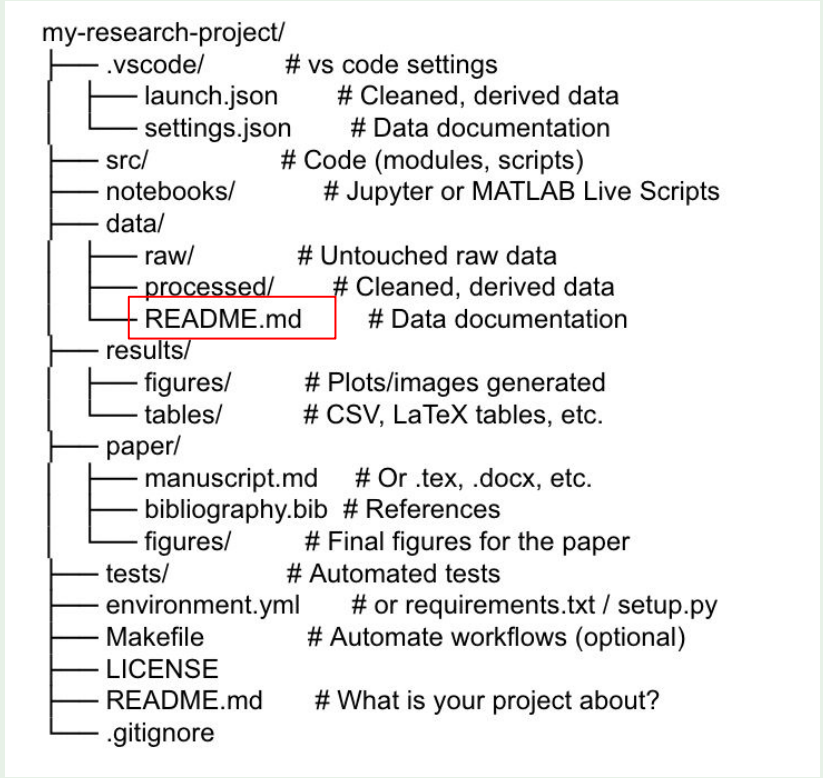
sub-001_population-cross_mri.nii

- **Suffix** preceded by an **underscore**
- Entities are composed of **key-label** pairs separated by **underscores**
- **Key** and **label** separated by **hyphen**
- **Keys**, **labels**, **suffixes** can only contain letters and / or numbers.
- Key-value combination can be repetitions from higher order folders to form unique filenames

Slides by courtesy of Dr. Remi Gau, adapted

Adding a Readme.md

- Entry point/first item that visitor reads
- Follows Markdown format
- There are several (slightly differing) markdown formats, but we are going to focus on git markdown.



Readme.md

Readme.md

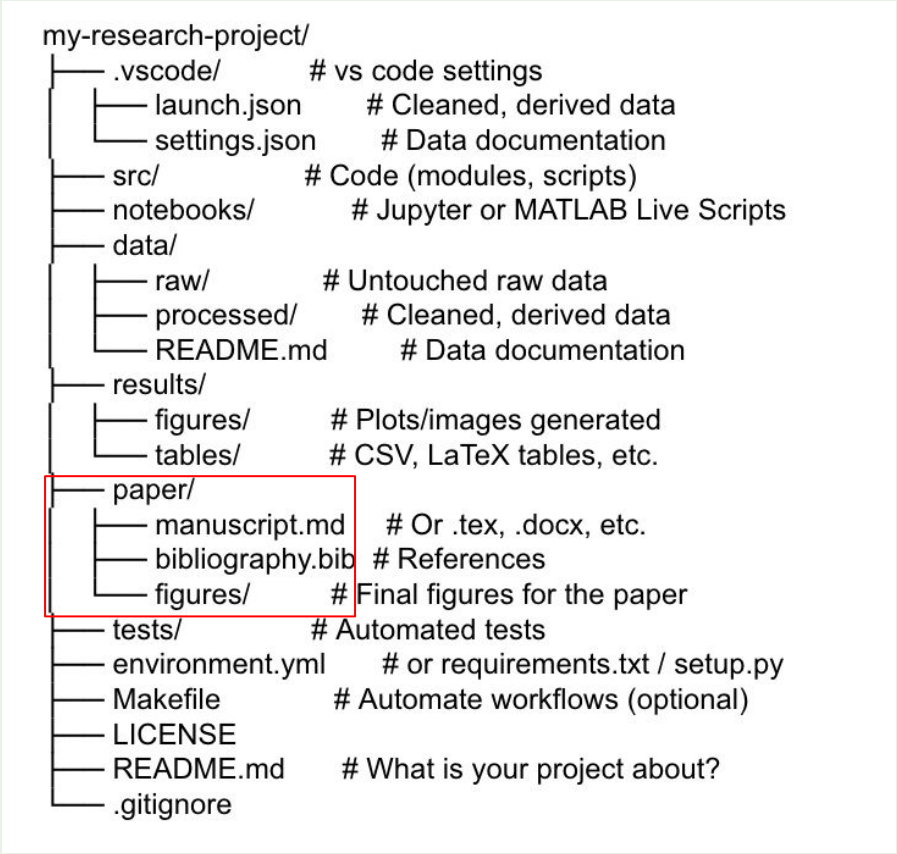
Files ending in .md indicate the Markdown style. We are going to follow Github Markdown.

A Readme.md should contain:

- Title and description of the project
- Author of the project and contact details
- Information about the license of the project - can everyone reuse your code?
- Structure of the project
- Anything that a user of the project should know

Examples: pydantic

Thinking about the writing early on



The repro pub - text editor and reference management system (based on reproducibility criteria)

Text editor	pro	con
MS Word/Google docs	- Plugins for most reference managers - good (?) platform for getting feedback/review - (Version controlled)	- Figures need to be imported each time
Markdown	-Integration with R (markdown, knitr) -In theory, fully reproducible (code and paper)	- No review option
Overleaf (online)	- Bibtex/Latex - Pro version has direct git and Dropbox link - Math support, templates - review option	- steeper learning curve
LaTeX (Desktop)	- Free - Bibtex file - Entirely version controllable - Math support - Create a figure folder that you save your figs in	-Very steep learning curve -No review option

writing

Collaborative publishing using overleaf

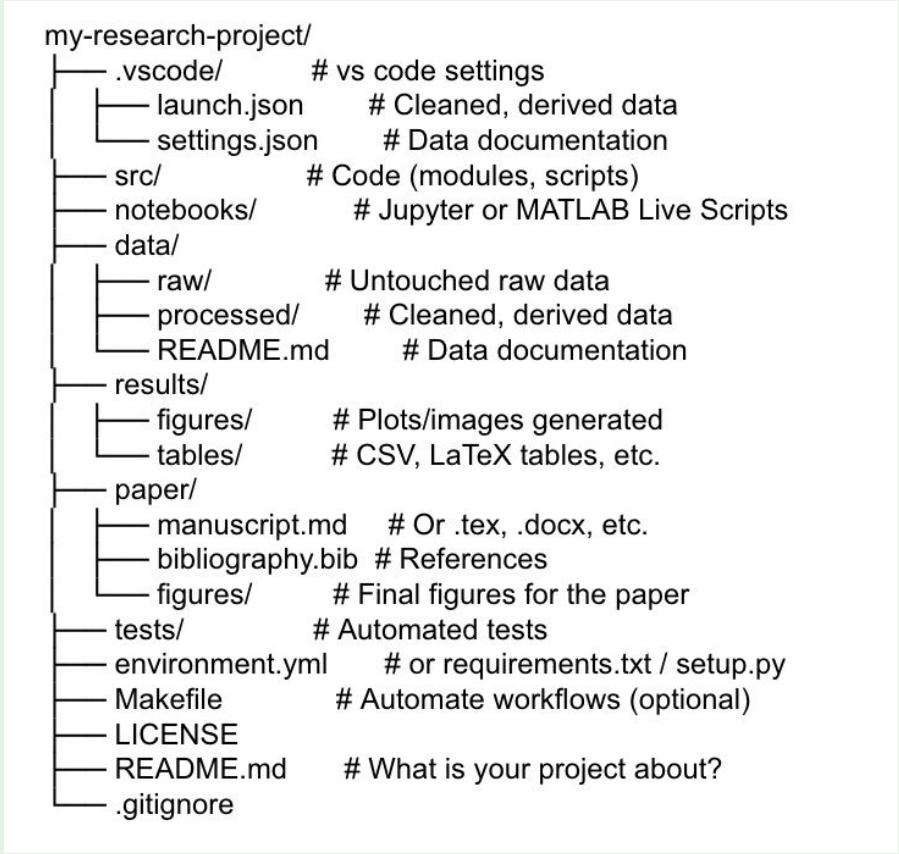
- All benefits of pure latex but
  - Working on a document together & review mode (suggest changes)
  - Import citations from paperpile, zotero, mendeley and endnote
  - Templates for publications, journals, conference posters etc
  - Link an image folder (i.e in Dropbox)
  - Git integration

Downside

- Internet dependency (except offline mode)

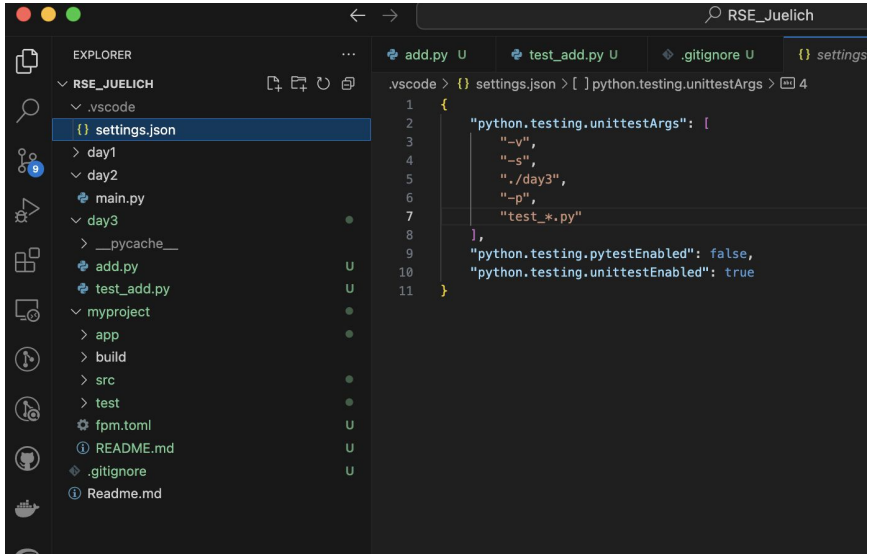
writing

Your code editor



Setting up your code editor

- VS code
  - Get the extensions for you code base



Important files:

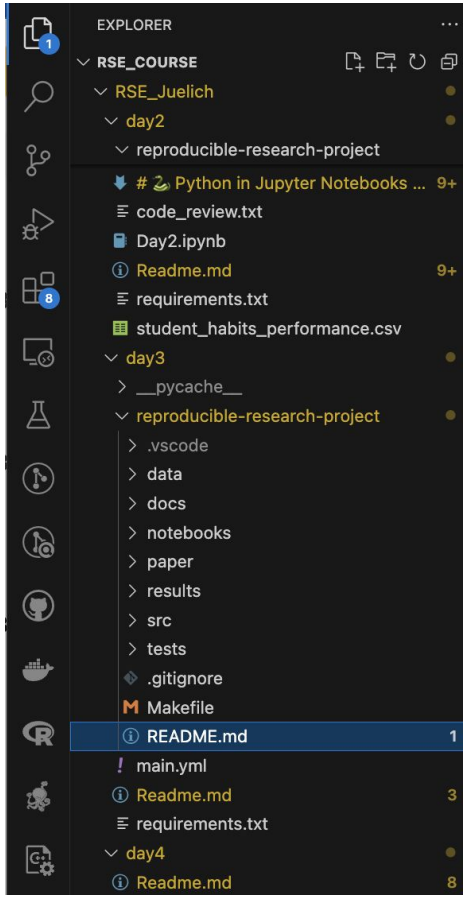
- Settings.json
  - Sets the behaviour of your vscode (linting, formatting etc)
- Launch.json
  - Specifies the behaviour of vs code running you programs
    - Debugging, which interpreter, which language

Important VS code extensions

Extensions for your programming language;

- Python, Julia, etc IDE.

- GitLens
- (Docker)
- Jupyter notebook extension



settings.json

Purpose: Configure **editor behavior**, **extensions**, and general **environment settings**

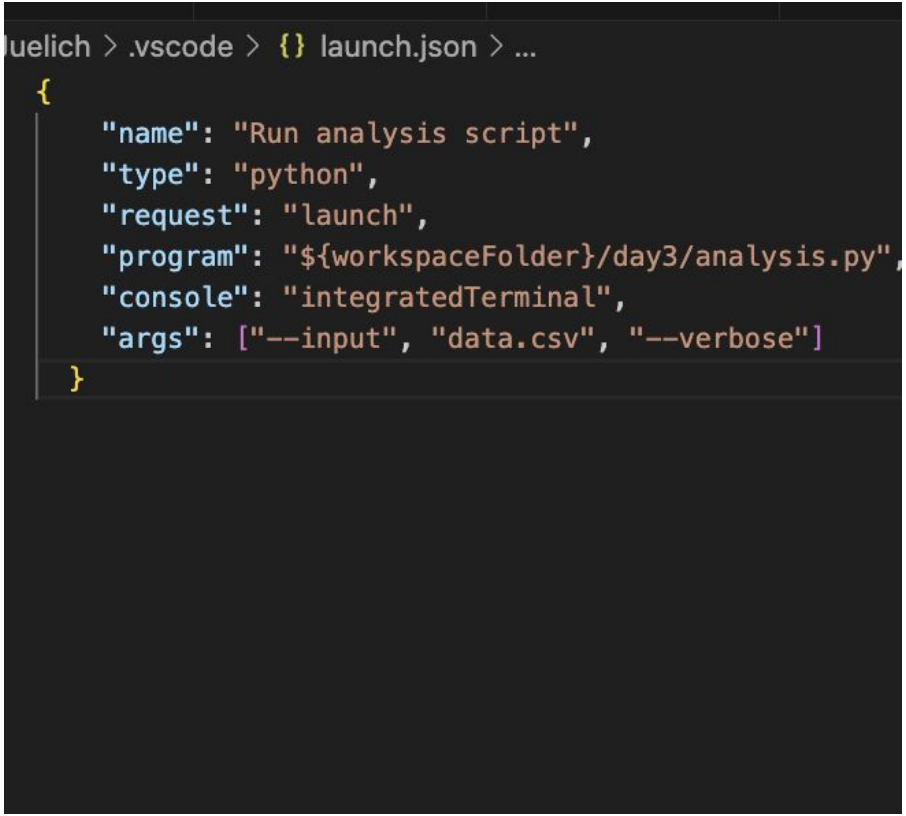
Scope: User-wide or workspace-specific
Examples:

- Tab size, font, auto-save, linting rules
- Default Python interpreter or Fortran compil path
- Format-on-save, excluded files, theme

launch.json

Specifies:

- Which script or executable to run
- What **command-line arguments** to pass
- Whether to **launch** a new process or **attach** to an existing one
- Where to show the output (terminal or debug console)
- Environment variables, working directory, debugging behaviour



Jupyter notebooks

- Can be installed via `pip install notebook` and the launched via `jupyter notebook notebook.ipynb` or `jupyter notebook`
- Allows to run notebooks in the browser
- Not super great for large research projects, but:
  - Operating system agnostic
  - Can be shared and opened in the browser via Google Colab
  - Can be run via vscode
  - A variety of kernels available

For research:

- Better: https://marimo.io/

Code editor

Matlab environment

Matlab projects

Use MATLAB Projects to manage files, paths, and settings.

- Go to **Home > New > Project > From Folder**
- Choose your folder (e.g., your project-name/ directory)
- MATLAB creates a .prj file to manage paths, startup code, etc.
-

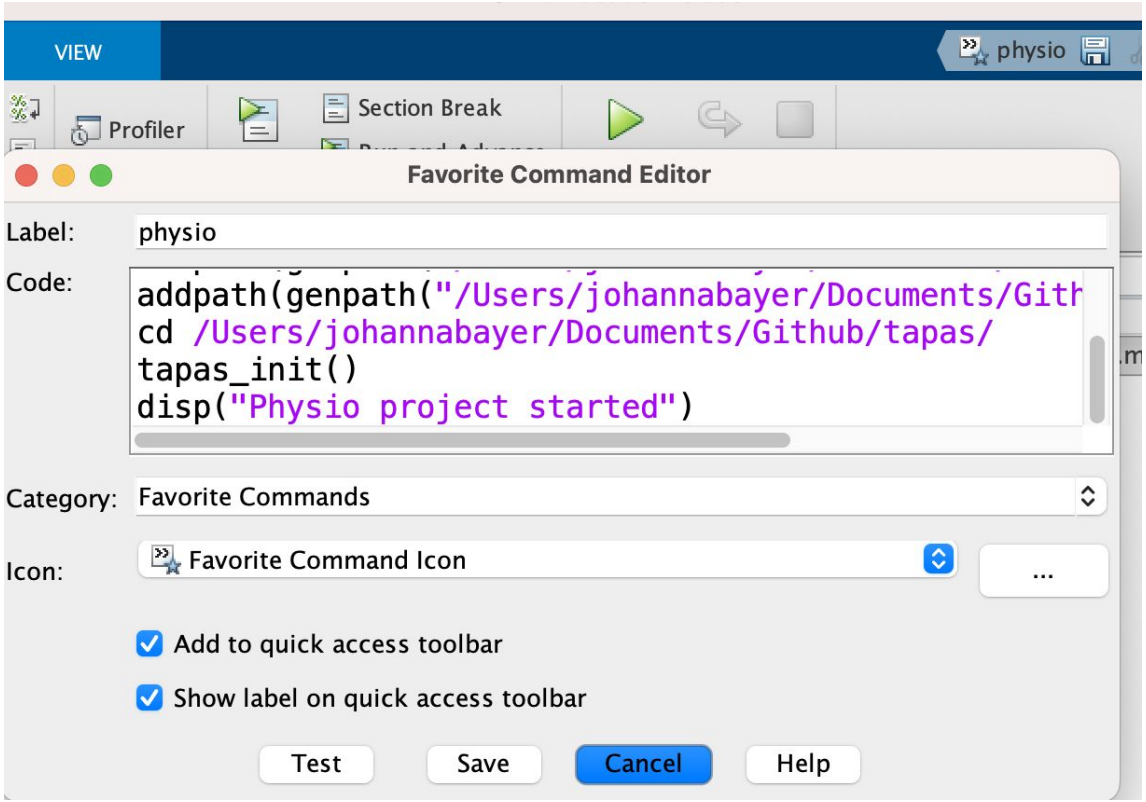
Configure startup paths:

- **Add a startup.m file** to automatically configure your environment on project open:

```
% startup.m
addpath(genpath('src'));
addpath('scripts');
disp('Project environment initialized.');
```

Code editor

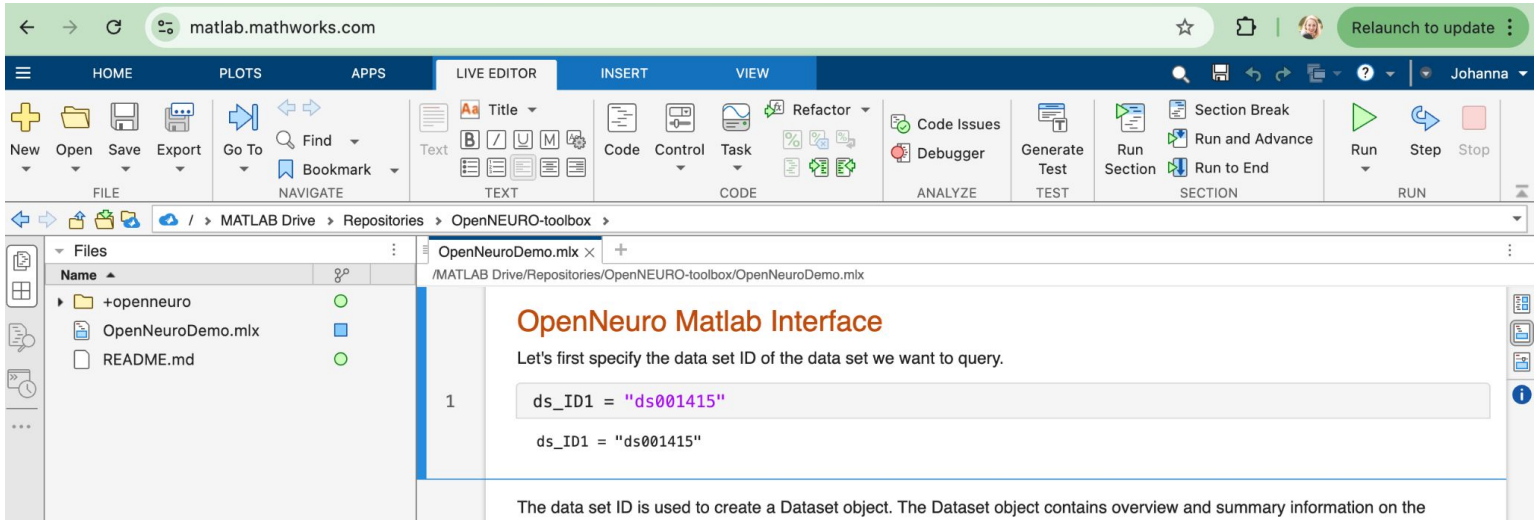
Set favourite commands



Code editor

Matlab Live Scripts

- Cell based matlab scripts (.mlx)
- Can be opened in the browser (Matlab online)
  - Allows to share and execute code
  - Still license needed



Set up how to compile/run your code

Die Qual der Wahl ….

- Run via VS code
- Run via the command line/terminal
- Run via Jupyter notebook
- Run via a (other) IDE
- Run on a cluster/cloud …

Setting up a coding project on a cluster

- Rsync of folders between cluster and local (images, tables etc)
- Ssh into cluster via vscode
- Sbatch

More resources on Research Compendia

- https://github.com/topics/research-compendium/
- https://the-turing-way.netlify.app/reproducible-research/compendia
- https://research-compendium.science/
- https://selinazitrone.github.io/YoMos2020/slidy_presentation.html#(1)

15 min break

(Software) version control

Version control

Git and Github

GitHub is a web-based platform designed for version control, collaboration, and project management. It uses **Git**, a distributed version control system, to enable developers, researchers, and teams to track changes in code, manage projects, and collaborate on software and other digital projects.

GitHub is widely used in software development, open-source projects, research, and even non-code workflows due to its robust collaboration tools and integration capabilities.

Githubs main purpose:

- Version control
- Collaboration

History of version control

Git Github Gitlab and Co

Git: the Software:

Git is a distributed version control system (VCS) designed to track changes in files and coordinate work among multiple people. It allows developers to work on projects locally, create branches for experimentation, and merge changes back into the main codebase. Git is primarily a command-line tool, and it provides a set of commands for managing repositories and collaborating with others.

Github: the Platform:

GitHub is a web-based platform built around Git that provides a centralized hosting service for Git repositories. It adds a graphical user interface (GUI) on top of Git, making it easier to perform common Git operations like creating repositories, managing branches, and collaborating with others. GitHub also offers additional features such as issue tracking, pull requests, code review tools, and project management capabilities. It's widely used for open-source projects, personal projects, and team collaboration.

Gitlab: a different version

GitLab is another web-based platform that provides hosting for Git repositories, similar to GitHub. It offers similar features such as repository management, issue tracking, and pull requests. However, the focus is more ob DevOps It integrates Git repository hosting with CI/CD pipelines, project management, and other DevOps tools. Gitlab also allows to host independent servers

History of version control

A history of version control

Pre-version control (-1990)

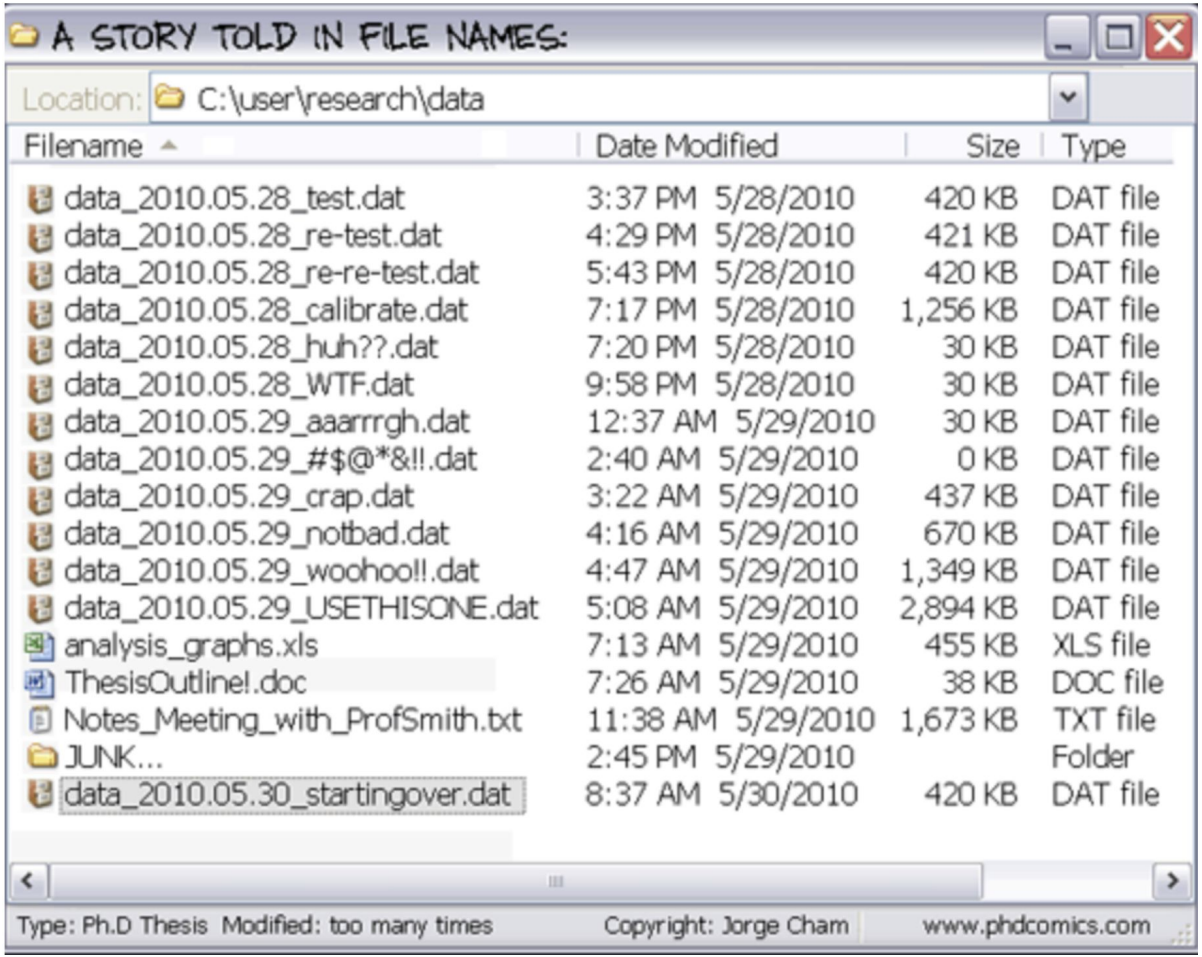
File based systems

Features

- Versions of files were kept individually

Challenges:

- Obvious



A history of version control

Centralized Version Control Systems (CVCS) (1990s)

Features:

- Centralized repositories to coordinate changes between team members.
- Developers checked out files to work on them and checked them back into the central server.

Challenges:

- Reliance on central server
- No ability to work offline

Examples:

- Concurrent Versions System (CVS) and Subversion (SVN)

A history of version control

Distributed Version Control Systems (DVCS) (2005–Present)

Features:

- Every developer has a full copy of the repository, including history.
- Allows offline work and faster operations.
- Resolves many limitations of CVCS.

Examples:

- Git
- Mercurial
- Bazar (Ubuntu)

A history of version control

Web-Based, collaborative Version Control Systems:

With the rise of the internet and collaborative development, web-based version control systems emerged. They provided a platform for hosting repositories and added web interfaces to simplify collaboration and code management. GitHub (2008), GitLab (2011), and Bitbucket (2008) are popular examples of such platforms.

Examples:

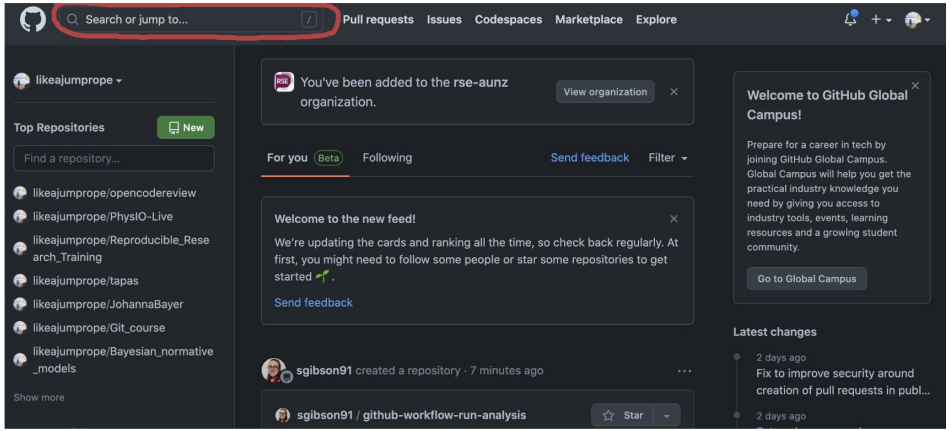
- Github
- Gitlab
- Bitbucket

Version control using Github/Gitlab

Community, T. T. W., & Scriberia. (2020). *Illustrations from the Turing Way book dashes*. Zenodo. 10.5281/ZENODO.3695300

Getting to know the Github interface

1. Log into your github account
2. The login page will get you to your dashboard. Here you find:
   a. An overview of your dashboard
   b. Your profile
   c. An overview of your repositories

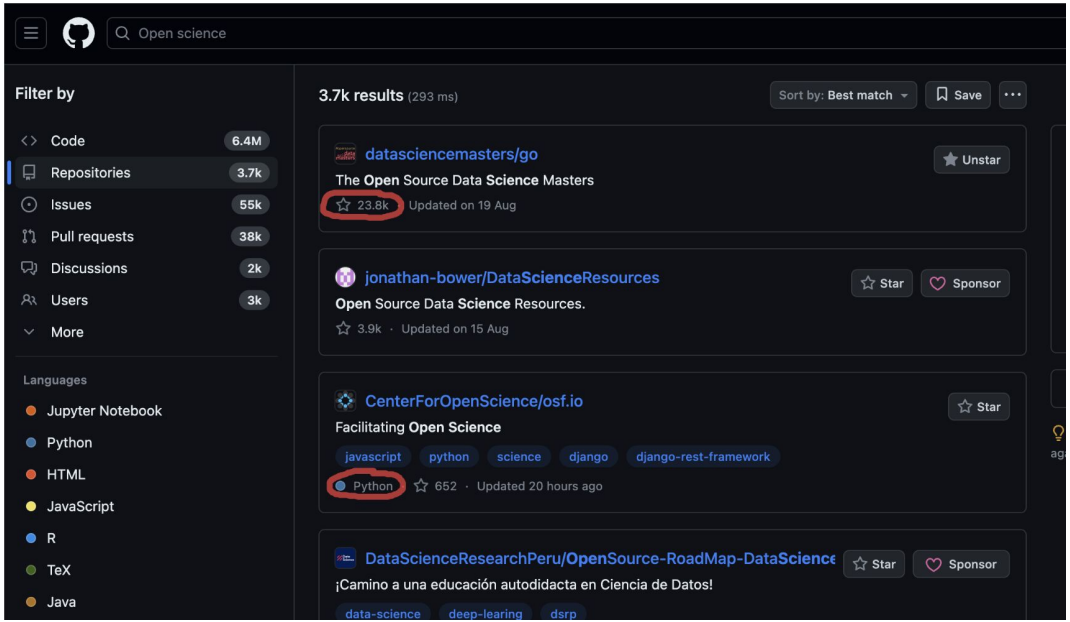


Getting to know the Github interface

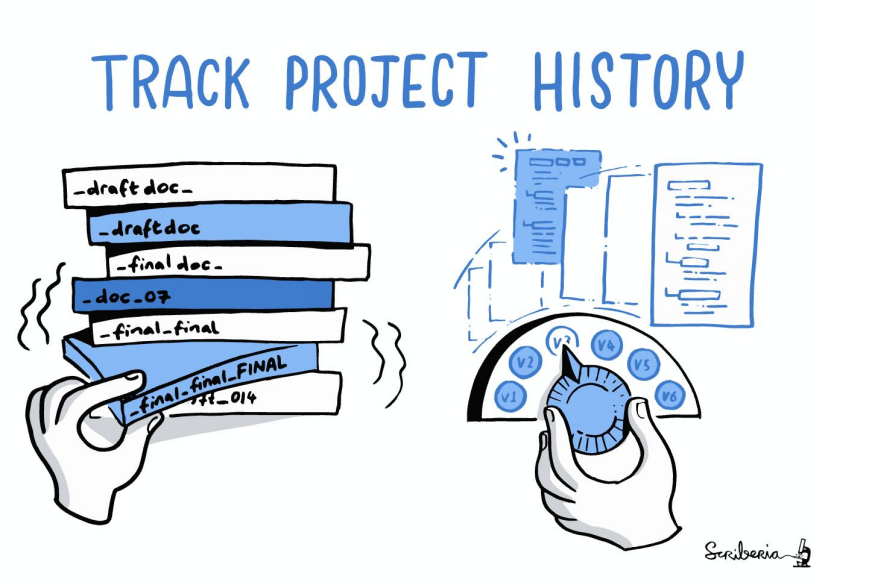
Repositories on Github are divided into:

- Those that you have write access to
- Those you can only view (the great majority)

You can use the search function to query all public repositories on Github

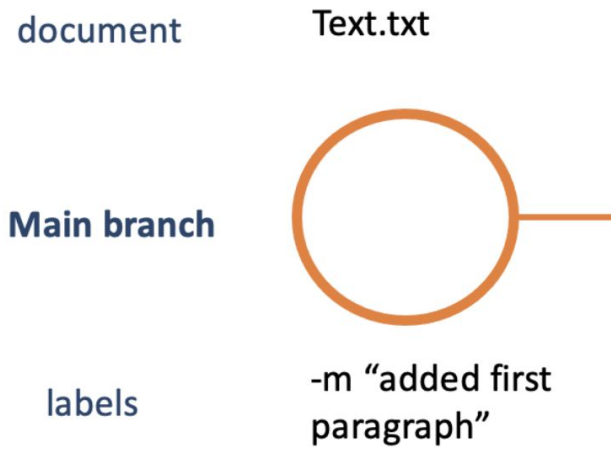


Committing



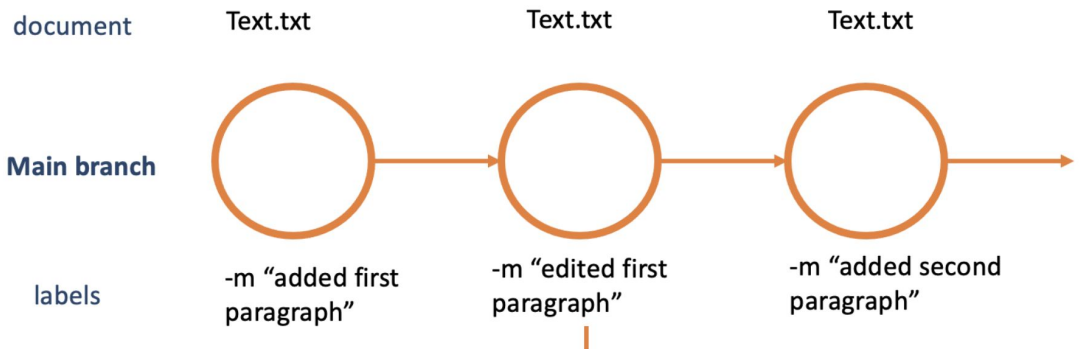
What is a commit?

- **Commits**: annotated snapshot of changes in a document



What is a commit?

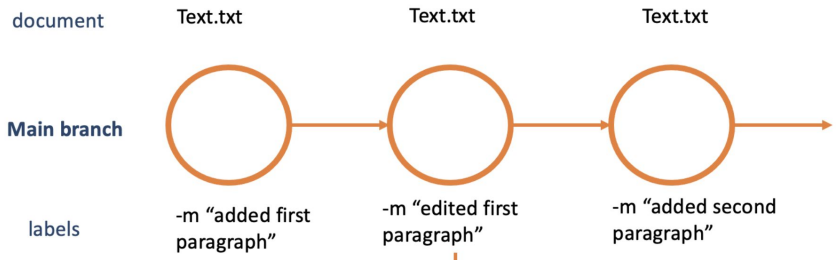
- **Commits**: annotated snapshot of changes in a document

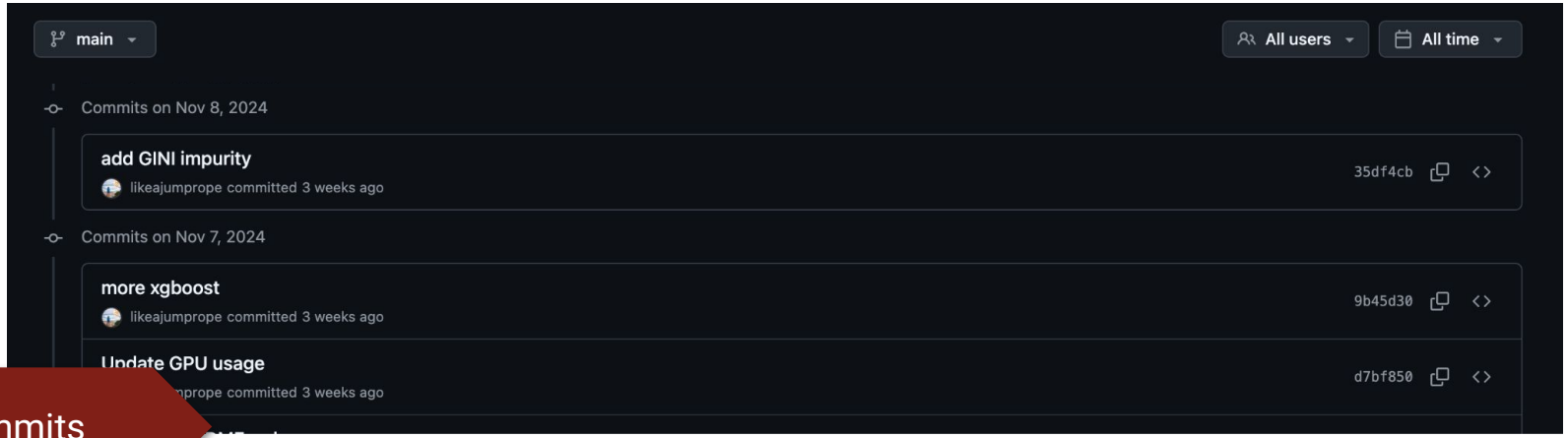


Git and Github: tracking your files, scripts and code

- Version control for files and scripts

- Ability to revert back to a previous commit

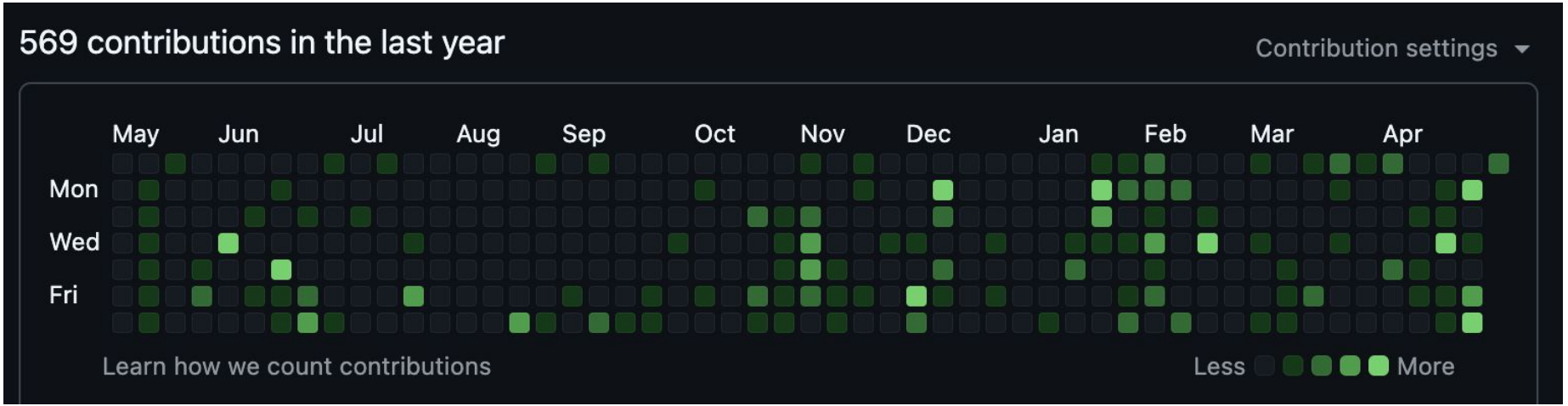




Commits

Git and Github: tracking your files, scripts and code

- Commit history

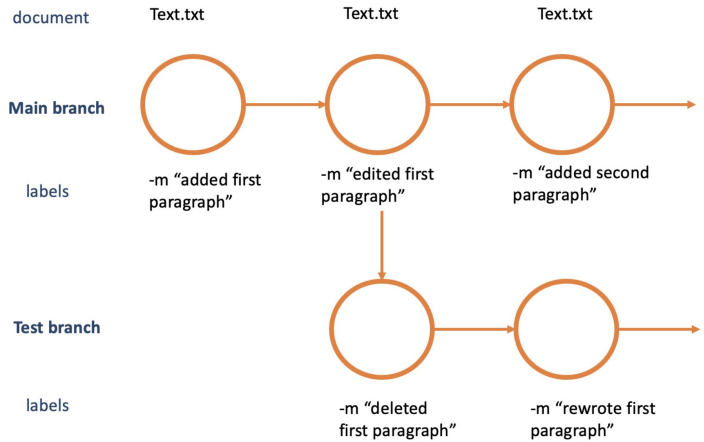


Commits

Git and Github: tracking your files, scripts and code

Collaboration (github online)

- **Branch**: Diversion point of repository and history

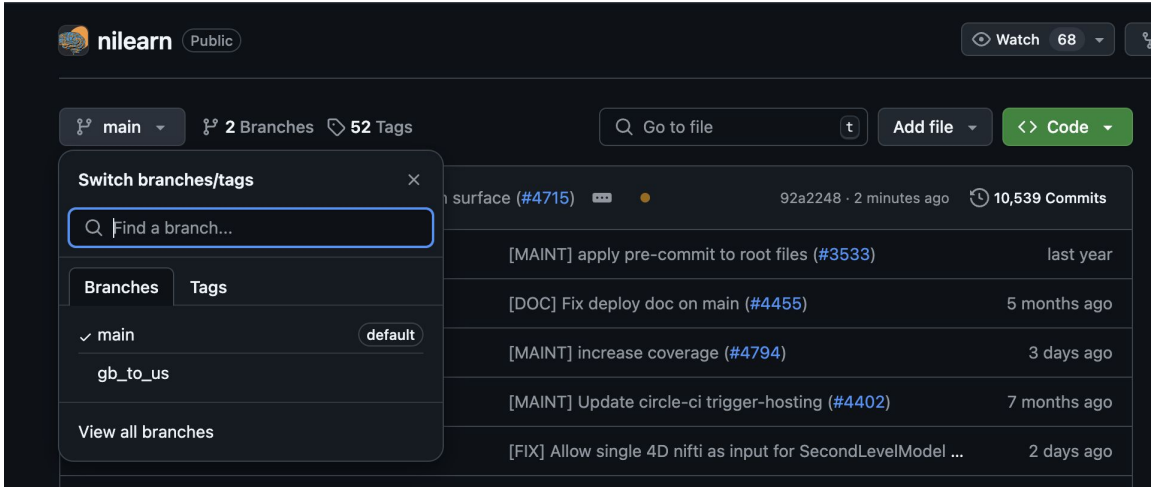


Branched

Git and Github: tracking your files, scripts and code

Collaboration (github online)

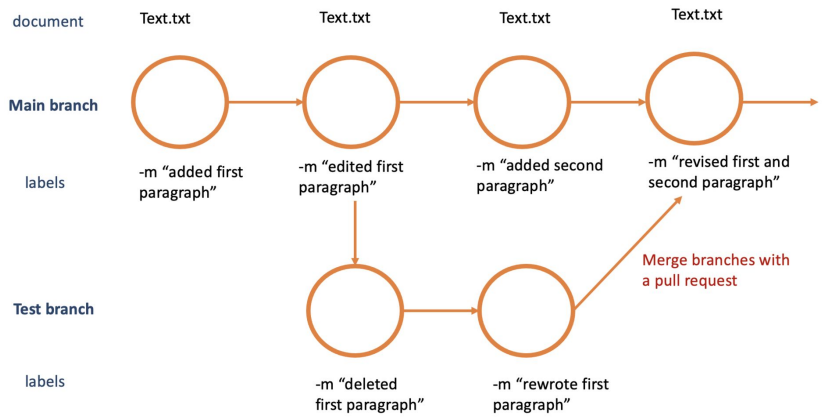
- **Branch**: Diversion point of repository and history



Git and Github: tracking your files, scripts and code

Collaboration (github online)

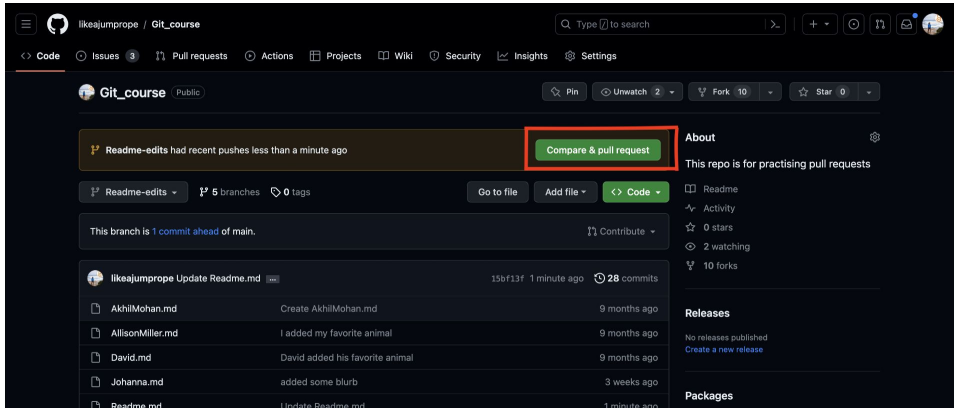
- **merging:** Unifying divergent branches, integrating changes
- Merging between branches is performed using a **pull request.**



Git and Github: tracking your files, scripts and code

Collaboration (github online)

- **merging:** Unifying divergent branches, integrating changes
- Merging between branches is performed using a **pull request.**



Merging

Git and Github: tracking your files, scripts and code

Collaboration (github online)

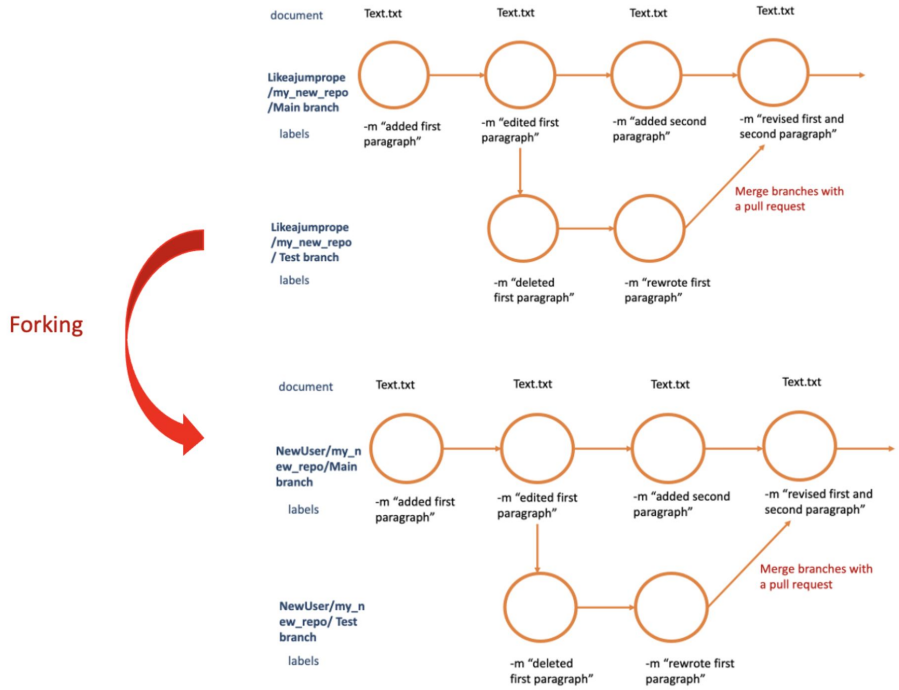
- **Fork**: make a copy of someone else's repository under your account.
- Repositories on Github are divided into two ca
  - Yours (you have writing permission)
  - Other people's (you don't)

- **Forking** creates a copy of someone else's repository that you own (aka you have writing permission)

Git and Github: tracking your files, scripts and code

Collaboration (github online)

-



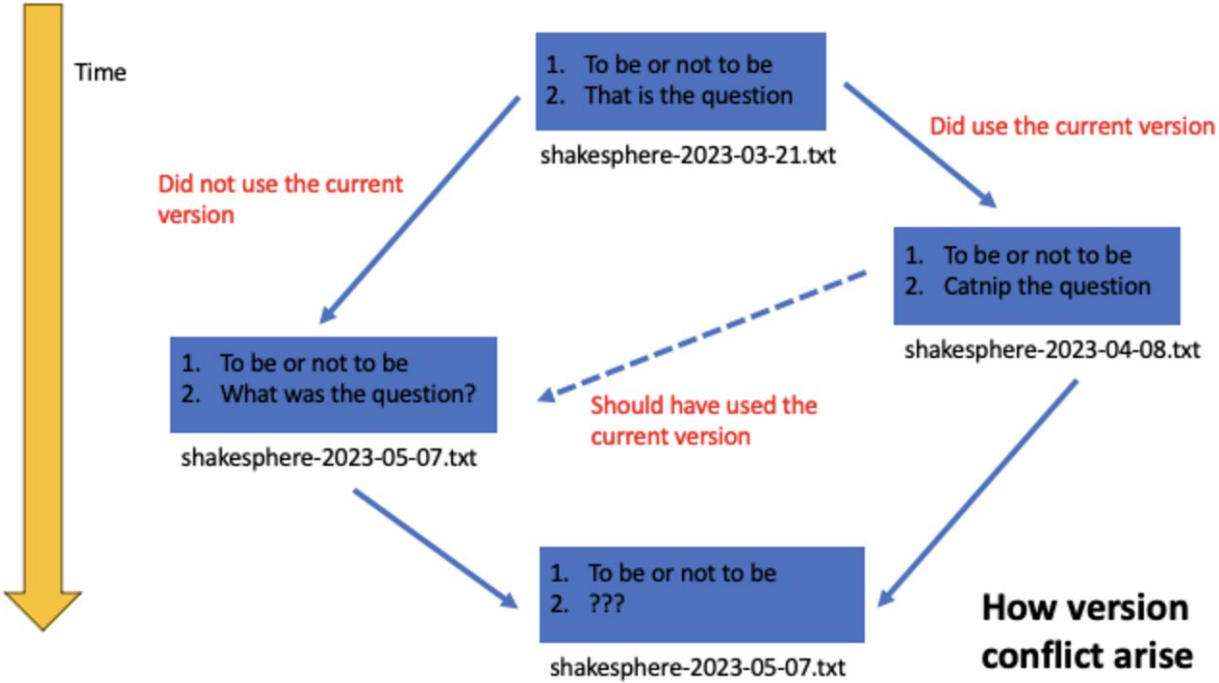
Forking

Why is it called pull request?

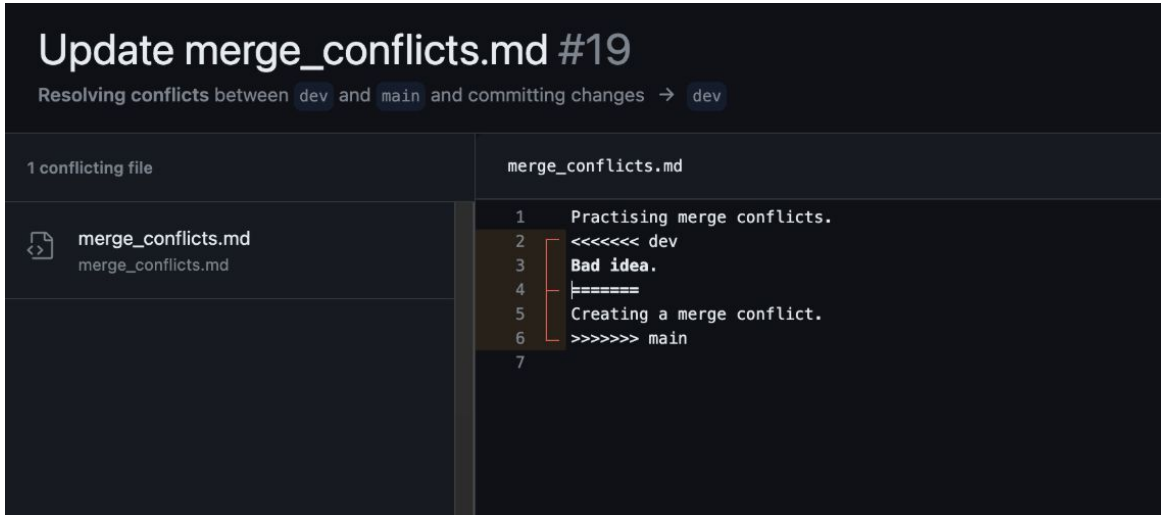
- Remember the separation between your and other people's repositories.
- In order to get your changes into someone else's repository, you have to request for them to be pulled in. This action is not up to you.
- You request a pull.
- The other person accepts, and pulls.

Fig. 191 Making your first pull request on GitHub. The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807.

Merge conflicts

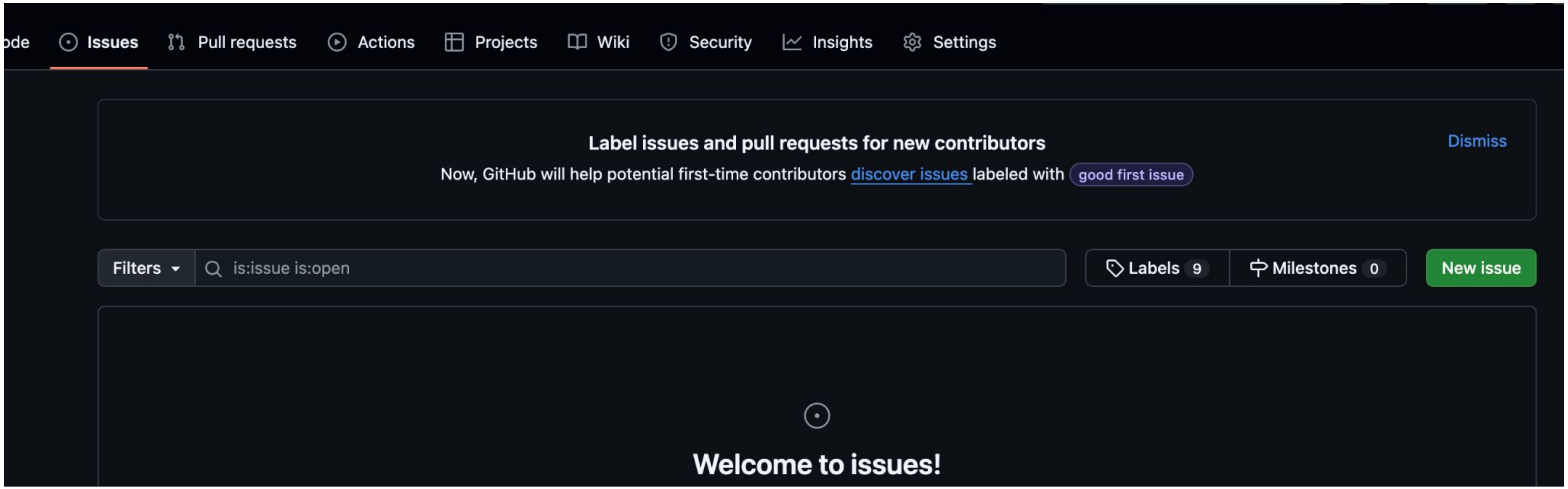


Resolving a merge conflict



Issues on Github

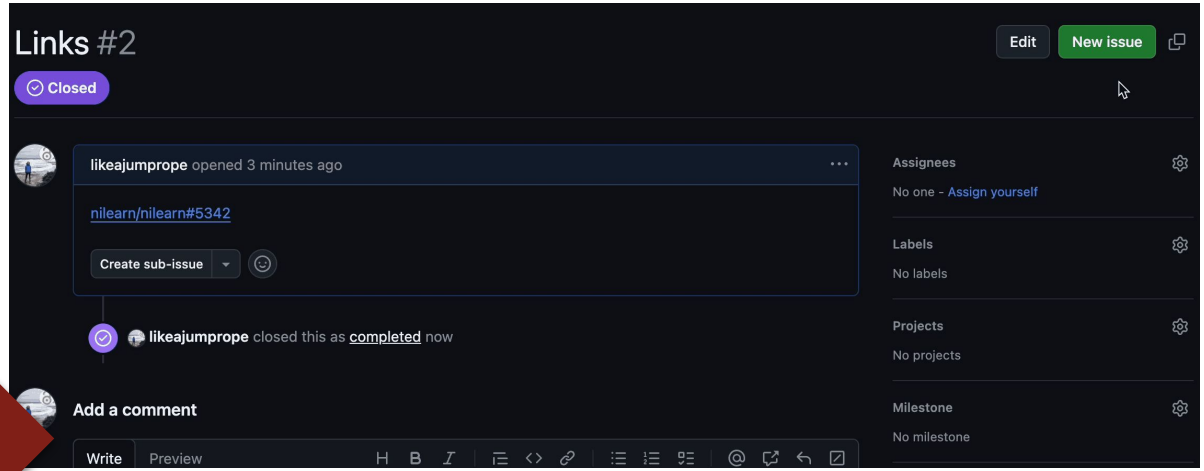
Issues on GitHub are a built-in tool for tracking tasks, enhancements, bugs, and other project-related activities. They provide a way for developers, collaborators, and even external users to communicate and manage the work associated with a GitHub repository.



Issues and collaboration on Github online

Issues are the way of communicating on Github

- Issues follow markdown format
- PRs can be referenced in Issues (and vice versa)
- Usually one PR responds to an issue



Activity 1 (20’)

1.

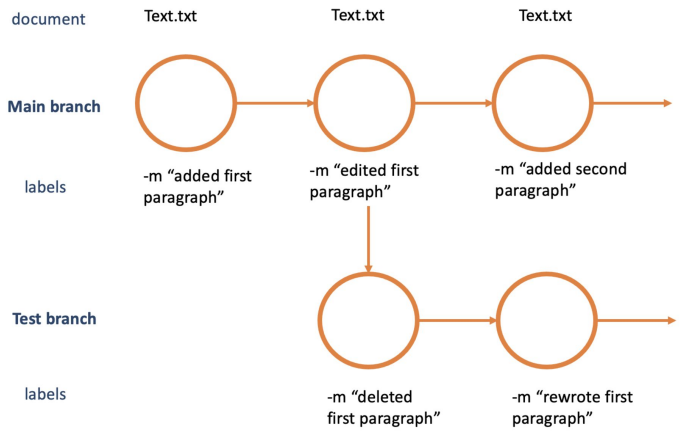
Working on Github online

1. Log into your Github account.
2. Fork the course repo: https://github.com/likeajumprope/RSE_Juelich
3. In the folder for day1, you’ll find a Readme.md.
4. Do Activity 1:
   - Create a file, add your favourite animal and make a PR.

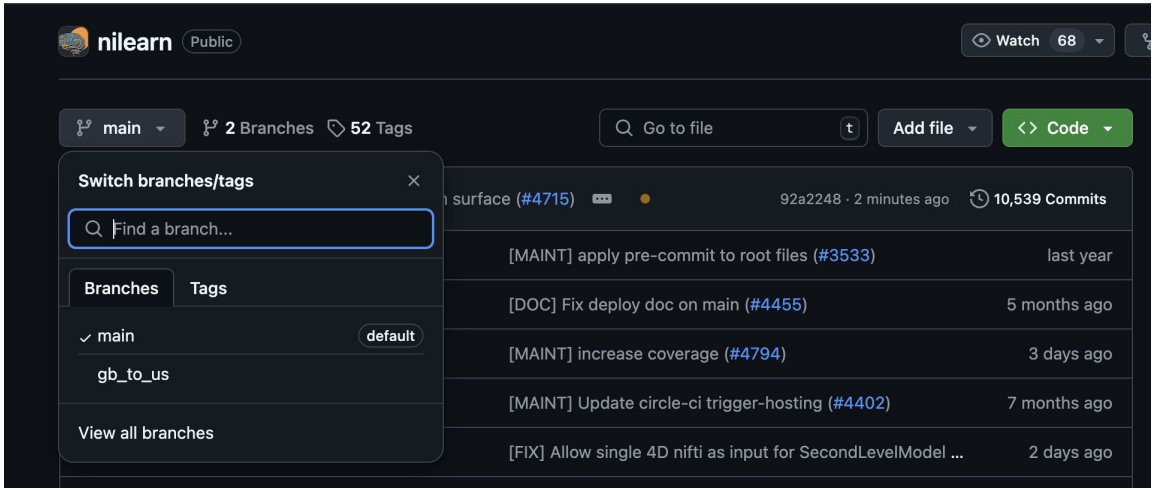
Branches

Collaboration (github online)

- **Branch**: Diversion point of repository and history



Make branch and merge back into your code



Branches

Activity 2 (15’)

Branching

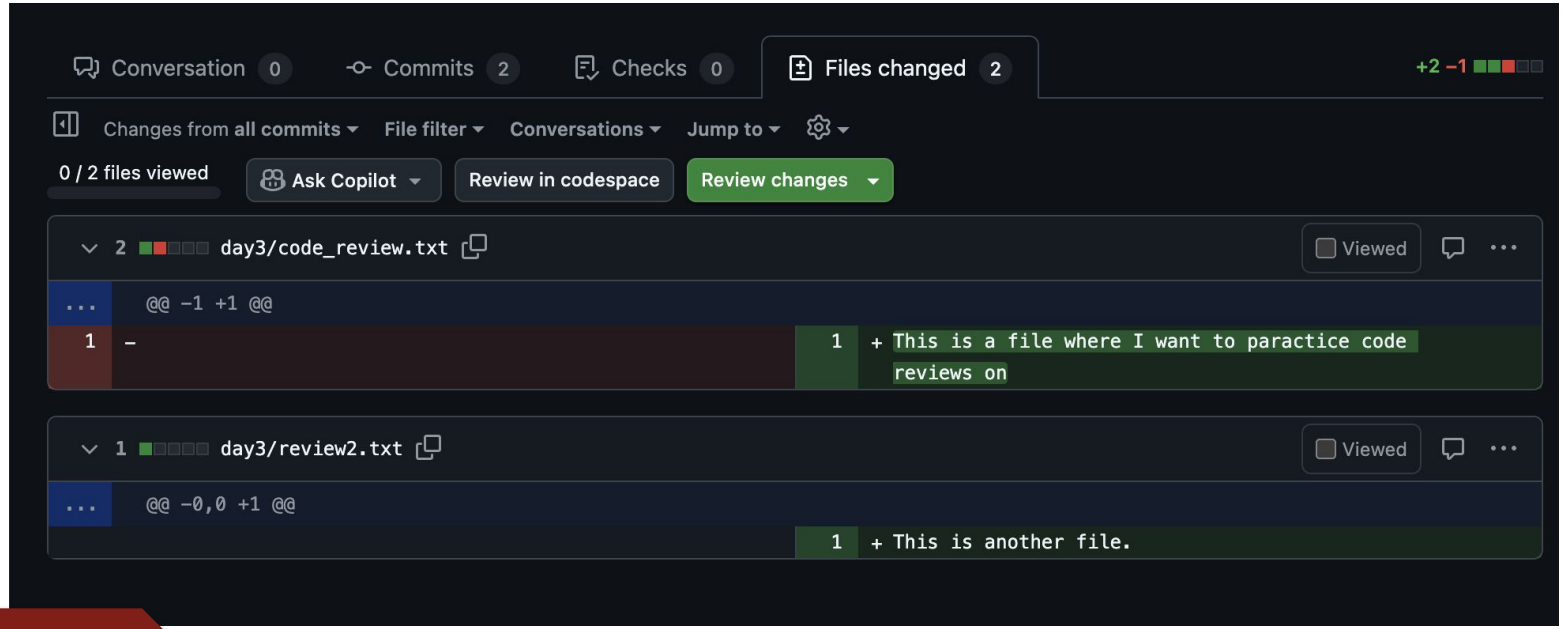
Branches

- Under the Readme.md section of day1, do Activity 2:
- Create a branch, make a change, merge it back into main

Break (15 min)

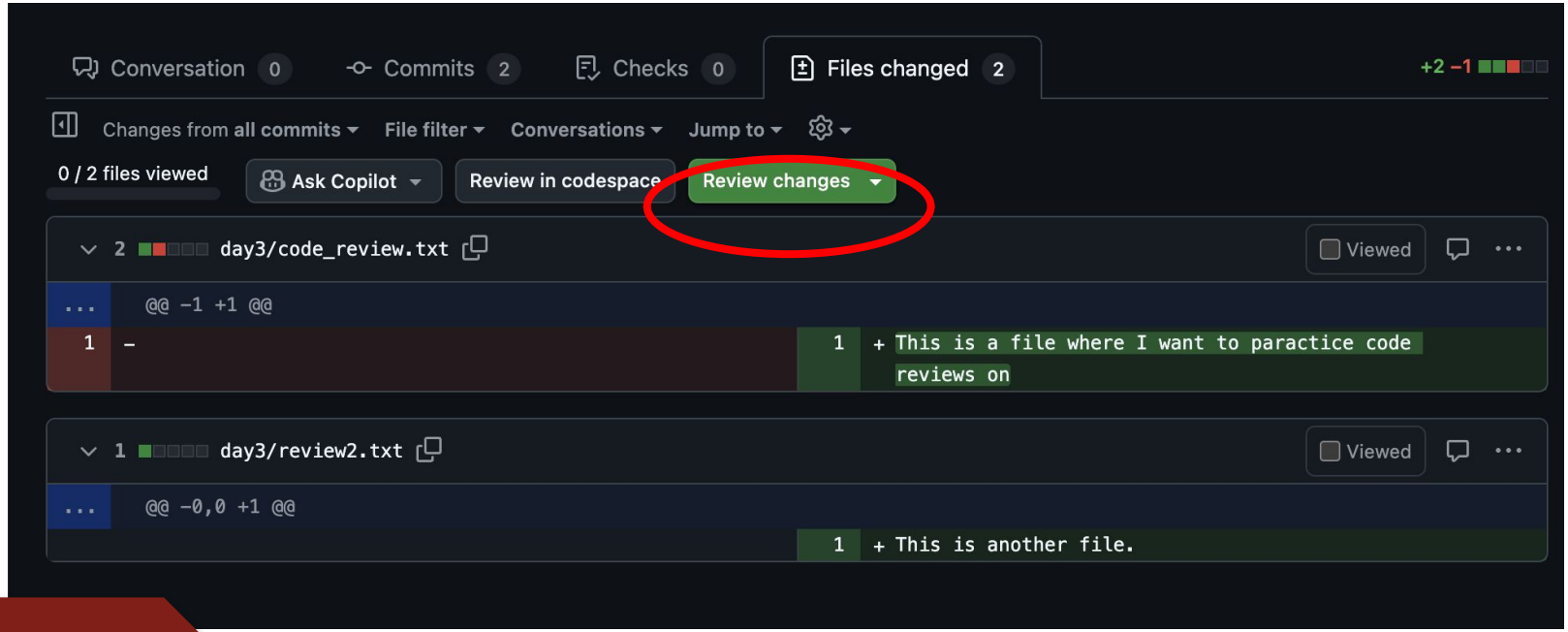
Review a PR online

Github allows for a very efficient code review process



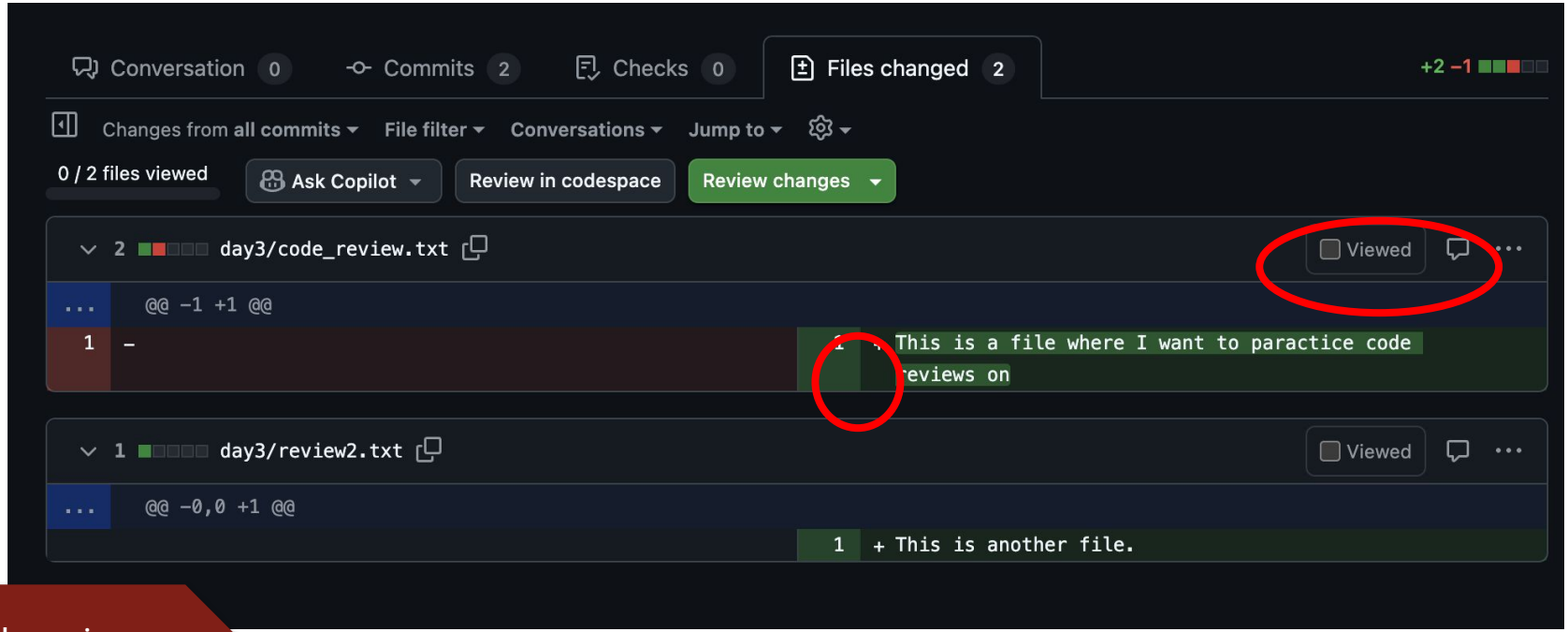
Review a PR online

Click on “review changes”



Review a PR online

Click on “review changes”



Activity 3 (15’)

Forking, PR’s and code review

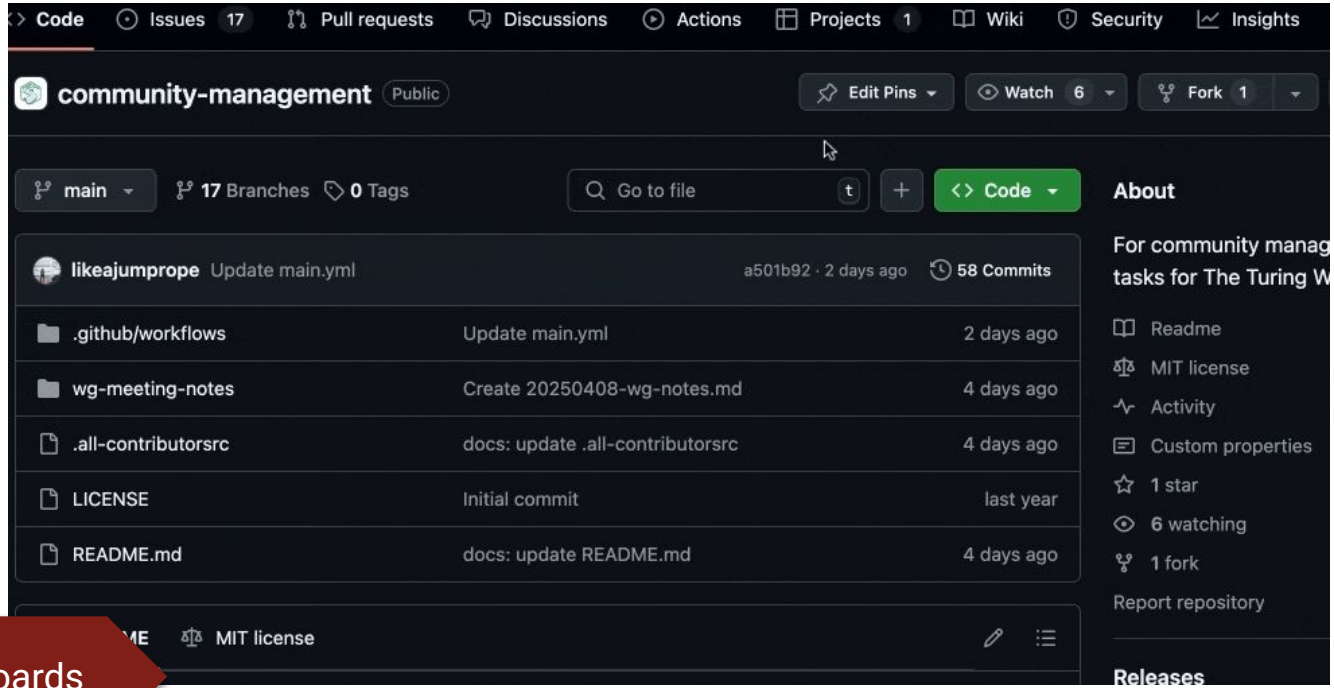
Under the Readme.md section of day1, do Activity 3

- Fork each other’s repo, make a PR, assign each other as reviewers and perform a code review

Some more Github features

Project management - boards on github

Project boards are a great way to organize a project and issues on github.

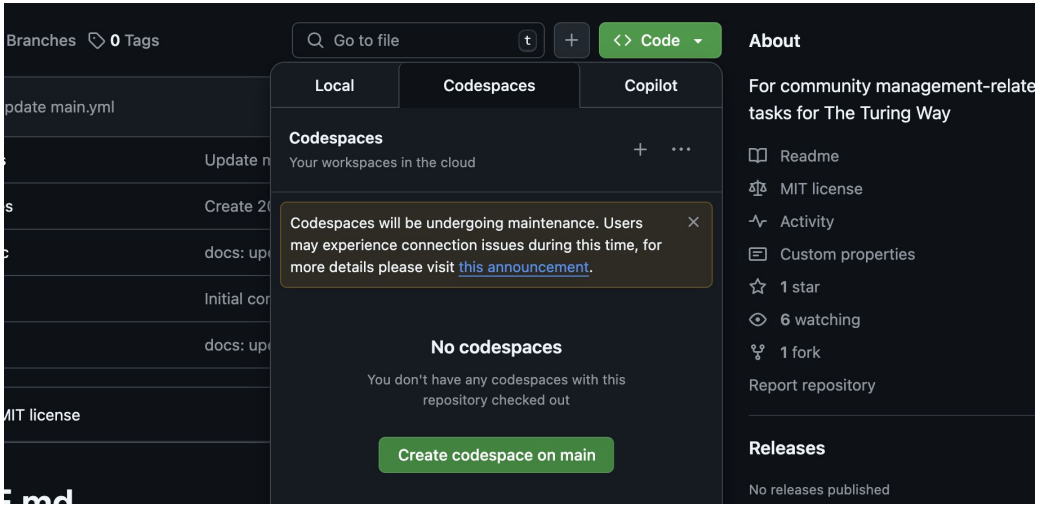


Project boards

Codespaces

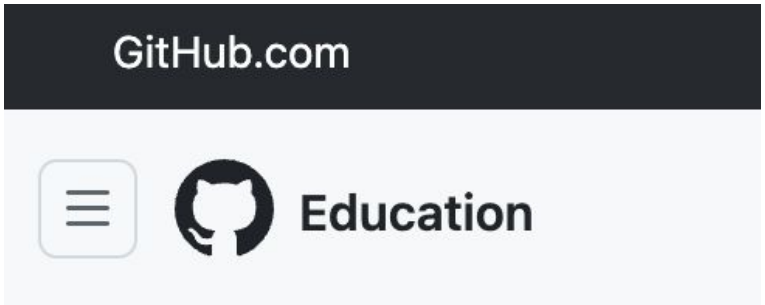
Github codespaces allows you to run files in Github repository in VS code instance in your browser.

- Good for quickly checking things
- CAREFUL: You have some free quota (as student etc), but then it's a paid service
- Close your instance!!



Github Education

- As a registered student, you can gain access to Github education
  - Free perks, such as:
    - Free Co-Pilot and Codespace
    - Full Notion AI plan
    - Licenses to Jetbrains IDE etc
    - Certifications



GH Education

Github Copilot

- Chat-GPT for CODE
- Integratable with VS code

- If you are a registered maintainer of an open source repository, you also get free access to codespaces and co-pilot.

What are your questions?

Ressources

Baxter, R, Chue Hong, N, Gorissen, D, Hetherington, J & Todorov, I 2012, 'The Research Software Engineer', Digital Research 2012, Oxford, United Kingdom, 10/09/12 - 12/09/12.

The Missing Semester of Your CS Education: https://missing.csail.mit.edu/

Nuest, D., Konkol, M., Pebesma, E., Kray, C., Schutzeichel, M., Przibytzin, H., & Lorenz, J. (2017). Opening the Publication Process with Executable Research Compendia. *D-Lib Magazine*, *23*(1/2). 10.1045/january2017-nuest

BIDS standard: https://bids.neuroimaging.io/index.htm

Goth, F. *et al.* Foundational competencies and responsibilities of a Research Software Engineer. *F1000Res.* **13**, 1429 (2024).

https://book.the-turing-way.org/collaboration/github-novice

https://book.the-turing-way.org/reproducible-research/vcs