

Open Science for HPC

Digital Research Academy



DIGITAL RESEARCH
ACADEMY

Dr. Johanna Bayer

Slides are licensed under CC-BY 4.0
[digital-research.academy](https://creativecommons.org/licenses/by/4.0/)

Code of Conduct

DO

- ✓ Be respectful
- ✓ Give everyone a chance to contribute
- ✓ Use inclusive language
- ✓ Appreciate and accommodate differences
- ✓ Lead by example
- ✓ Everyone is learning!

DON'T

- ✗ Repeatedly interrupt
- ✗ Bully, discriminate, or harass
- ✗ Make fun of personal appearance or choices

Have a concern?

- Report by telling the instructor.
- Report by contacting Joyce at DRA:
joyce.kao@digiresacademy.org

Outline

- Data version control
- Finding, citing, managing & licensing <insert research output here > openly
- Sharing <insert research output here> openly
- Good coding practices
- Computational reproducibility - stabilizing your computing environment
- Open and reproducible science in complex research projects

Data version
control

Finding, citing,
managing &
licensing

Sharing
openly

Good coding
practices

Computational
reproducibility

Complex
projects

Data version control

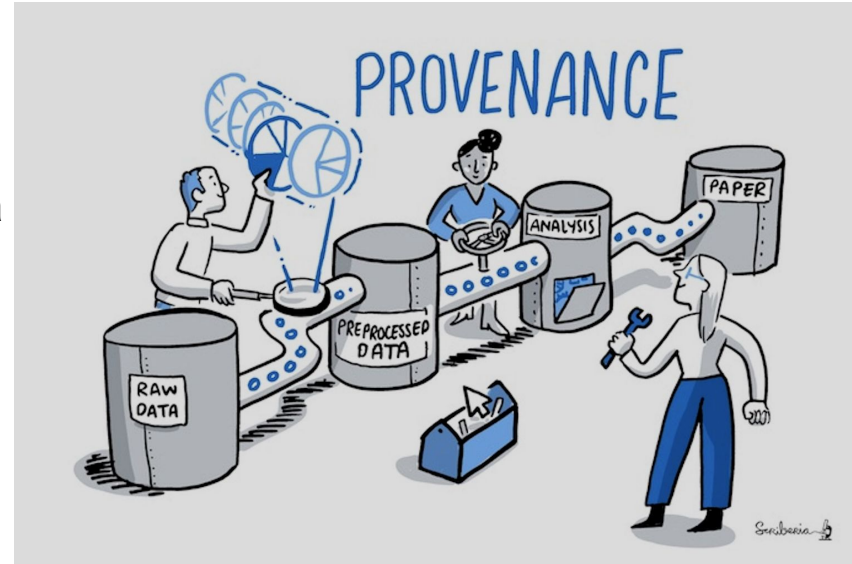
Provenance on which data in which version was underlying which computation is crucial for reproducibility. The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: [10.5281/zenodo.3332807](https://doi.org/10.5281/zenodo.3332807).

Data version control

Data is a living creature

- Throughout the research process, data can be:
 - datasets can be extended with new data
 - adapted to new naming schemes
 - reorganised into different file hierarchie
 - updated with new data points
 - modified to fix any errors.
- > threatens reproducibility

Provenance: describing the origin of an object



Provenance on which data in which version was underlying which computation is crucial for reproducibility. The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: 10.5281/zenodo.3332807.

File organisation: Version control

Product	Short description	
Git/Github	Scripts and files (up to 100 MiB per file/ 50MB per repo)	https://github.com/
Git Large file Storage (LFS)	Creates pointers to large files that are stored elsewhere (i.e git server [paid], Dropbox)	https://git-lfs.com/
Datalad	Data sets and pipelines tracking, uses git annex & direct integration with git	https://www.datalad.org/
Data Version Control (DVC)	Data sets and (ML) pipeline tracking, ignores git	https://dvc.org/
ML flow (et al.)	(ML) Experiment tracking (python)	https://mlflow.org/

Git LFS

Git Large File Storage (Git LFS) is an open-source Git extension that improves how large files are handled in Git repositories. Instead of storing the content of large files directly in the repository, Git LFS replaces them with **lightweight pointers** and stores the actual file content in a separate storage location.

Git LFS comes with a **command-line extension to Git** and allows you to treat files of any size alike, using standard Git commands.

A major shortcoming, however, is that Git LFS is a centralised solution. Large files are not distributed but stored on a remote server. This usually requires setting up your server **or paying for a service** - which can make it very inaccessible.

Git LFS

How Git LFS Works

1. Pointers in the Repository:

- Git LFS replaces large files in the repository with small pointer files. These pointer files are tracked in Git instead of the actual file.

2. External Storage:

- The actual large files are stored in a separate Git LFS server or Git hosting service that supports Git LFS (e.g., GitHub, GitLab, Bitbucket).

3. Efficient Fetching:

- When you clone or pull a repository, Git LFS downloads the large files only when needed, reducing the size of the repository.

Data Version Control (DVC)

DVC (<https://dvc.org/>)

- A tool for version-controlling large datasets, machine learning models, and pipelines, specifically geared toward **machine learning and data science workflows**.
- DVC guarantees reproducibility by consistently maintaining a **combination of input data, configuration, and the code that was initially used to run an experiment**.
- Seamless integration with git



Data Version Control (DVC)

- Similar to LSF, uses pointers
- Tracks changes in data sets **using .dvc files.**
- Pipeline management:

```
### DVC
$ cat dvc.yaml
stages:
  prepare:
    cmd: python src/prepare.py
    deps:
      - data/raw
      - src/prepare.py
    outs:
      - data/prepared/test.csv
      - data/prepared/train.csv
```

- Experiment tracking

```
### DVC
$ dvc metrics show
Path          accuracy
metrics/accuracy.json 0.67934
```

Data version
control



Data Version Control (DVC)



Strengths:

1. Handles Large Files Efficiently

- Tracks large files (e.g., datasets, models) without storing them directly in Git, using .dvc pointer files to manage metadata.

2. Pipeline and Workflow Management

- Provides built-in tools (dvc.yaml) to define and track pipelines, ensuring reproducibility of data science and machine learning workflows.

3. Git Integration

- Integrates tightly with Git, enabling version control for both data and code in a single repository.

Weaknesses:

- **Learning curve:** Using DVC effectively requires understanding both Git workflows and DVC-specific commands for managing pipelines and datasets.

Git submodules

- **Git submodules** allows to include one Git repository (the **submodule**) as a subdirectory within another Git repository (the **parent repository**).
- useful when you want to incorporate **another repository's code** into your project while keeping it as a separate entity.
- Issue: Git submodules **do not update** with the parent library

For example:

- You might include a shared library or module from another repository in your project without copying the files directly.
- Submodules allow you to **track a specific version** of the included repository while maintaining a connection to its original repository.

Git submodules

Strength

1. **Version Control:** Submodules allow you to pin dependencies to specific commits.
2. **Modularity:** Keeps projects modular by separating the submodule's codebase.
3. **Avoid Code Duplication:** Submodules link to the original repository without copying its files.
4. **Works with Git Workflow:** Submodules integrate seamlessly with Git, leveraging Git's tools and workflows.

Weaknesses:

1. **Complexity:** Submodules can complicate workflows, especially for new Git users.
2. **Manual Updates:** Updating a submodule requires manual intervention (e.g., `git submodule update`).
3. **Nested Submodules:** Managing nested submodules can be particularly challenging.
4. **Merge Conflicts:** Merge conflicts involving submodules can be harder to resolve.
5. **Dependency Awareness:** All team members need to understand how submodules work to avoid issues during collaboration



Git-annex

- [Git Annex](#) is an open-source tool that extends Git's functionality to manage large files or collections of files across multiple locations. Instead of storing large files directly in the Git repository, Git Annex stores **symlinks (symbolic links)** in the repository while managing the actual files in a separate annex.
- It is especially useful for managing large datasets, media files, or files that need to be distributed across multiple devices or storage systems.

Strengths:

1. **Distributed and Decentralized:** Ideal for teams with varying levels of connectivity or where central hosting is impractical.
2. **Customizable Storage:** Can use multiple storage backends simultaneously.
3. **Data Redundancy:** Ensures data integrity by storing files across multiple locations.
4. **Offline First:** Files and changes can be queued for syncing when online.
5. **Advanced Rules:** Configure rules for where files should be stored or downloaded based on system requirements.



Git-annex

Weaknesses:

1. **Complexity:**

- Configuration and usage can be more complicated than Git LFS.
- Requires knowledge of symlinks and Git Annex-specific commands.

2. **Not Seamlessly Integrated with Git Hosting Services:**

- Unlike Git LFS, Git Annex does not natively integrate with platforms like GitHub or GitLab.

3. **No Built-in CI/CD Integration:**

- While it works well for distributed storage, it lacks native integration with DevOps pipelines.

Datalad



DataLad

DataLad, builds upon git and git-annex. Like [git-annex](#), it allows you to version control data and share it via third-party providers but simplifies and extends this functionality.

- In addition to sharing and version controlling large files; it allows recording, sharing, and using software environments, recording and re-executing commands or data analyses, and operating seamlessly across a hierarchy of repositories.

```
$ datalad containers-run -n software \  
-m "Evaluate SGD classifier on test data" \  
--input 'data/raw/val/n03445777' \  
--input 'data/raw/val/n03888257' \  
--output 'accuracy.json' \  
"python3 code/evaluate.py"  
[INFO] Making sure inputs are available (this may take some time)  
[INFO] == Command start (output follows) =====  
[INFO] == Command exit (modification check follows) =====  
{'accuracy': 0.7363751584283904}  
run(ok): /home/me/usecases/ml-project (dataset) [singularity exec  
save(ok): . (dataset)  
action summary:  
  add (ok: 2)  
  get (notneeded: 4)  
  run (ok: 1)  
  save (notneeded: 1, ok: 1)
```

<https://www.datalad.org/>

Data version
control

Datalad - strength



Handles Large Datasets Efficiently

- **Uses Git-Annex:** Tracks large files without bloating the repository by storing content in external locations and maintaining lightweight metadata in Git.
- **Remote Backends:** Supports multiple storage backends (local storage, SSH servers, cloud storage like AWS S3, and more).
- **Partial Data Access:** Allows downloading specific parts of a dataset without needing the entire dataset (on-demand access).

Reproducibility

- **Version Control for Data:** Enables tracking of changes to datasets, ensuring reproducibility across different research stages.
- **Snapshots:** Provides a history of dataset versions, making it easy to revert to a previous state.

Integration with Git

- Fully integrates with Git workflows, allowing you to track and version both datasets and code.
- Works well in collaborative environments using Git-based tools (e.g., GitHub, GitLab).

FAIR Principles Compliance

Metadata Management

- Enables adding rich metadata to datasets, improving their discoverability and usability.

Dataset Discovery and Sharing

- Provides tools for discovering and sharing datasets, particularly useful in scientific research.
- Supports distributed sharing via Git-Annex, enabling dataset collaboration.

Automation-Friendly

- Designed for use in both interactive and automated environments.
- CLI and Python API support enable integration into pipelines and workflows.

Datalad -weaknesses



1. Steep Learning Curve

- The combination of Git, Git-Annex, and Datalad concepts can be challenging for new users, especially those unfamiliar with Git workflows.
- Requires understanding both Git and Git-Annex to fully utilize Datalad's capabilities.

2. Dependency on Git-Annex

- Relies heavily on Git-Annex, which itself has a learning curve and specific limitations (e.g., requires additional setup for certain remotes).
- Git-Annex symlinks may cause confusion for users not familiar with its architecture.

3. Limited Mainstream Adoption

- While powerful, Datalad is not as widely adopted as other tools (e.g., DVC or Git LFS) in some domains, which might limit its community support and available resources.

4. Performance on Very Large Repositories

- Handling metadata for extremely large repositories with many files can become slow, as Git and Git-Annex are not optimized for such scenarios.

5. Complex Setup for Non-Technical Users

- Installing and configuring Datalad, especially in multi-user or multi-institution setups, can be challenging.
- Requires additional configuration for integrating with cloud storage or other remotes.

ML flow (python)

MLflow is an open-source platform for managing the **end-to-end machine learning (ML) lifecycle**. It provides tools to streamline **experimentation, reproducibility, and deployment of machine learning models**. MLflow integrates well with existing ML libraries and frameworks, making it a versatile tool for practitioners.

The screenshot displays the MLflow 2.18.0 web interface. The top navigation bar includes the MLflow logo, version 2.18.0, and tabs for 'Experiments' and 'Models'. The 'Experiments' tab is active, showing a sidebar with a search bar and a list of experiments. The main content area shows the 'Default' experiment with tabs for 'Runs', 'Evaluation', 'Experimental', and 'Traces'. A search bar and filters are present, along with a table of runs.

Run Name	Created	Dataset	Duration	Source	Models
luminous-flea-485	29 seconds ago	-	1.3s	mlflow_...	sklearn
grandiose-loon-828	2 minutes ago	-	1.6s	mlflow_...	sklearn
gentle-stork-276	4 minutes ago	-	1.8s	mlflow_...	-

Data version
control

Comparison

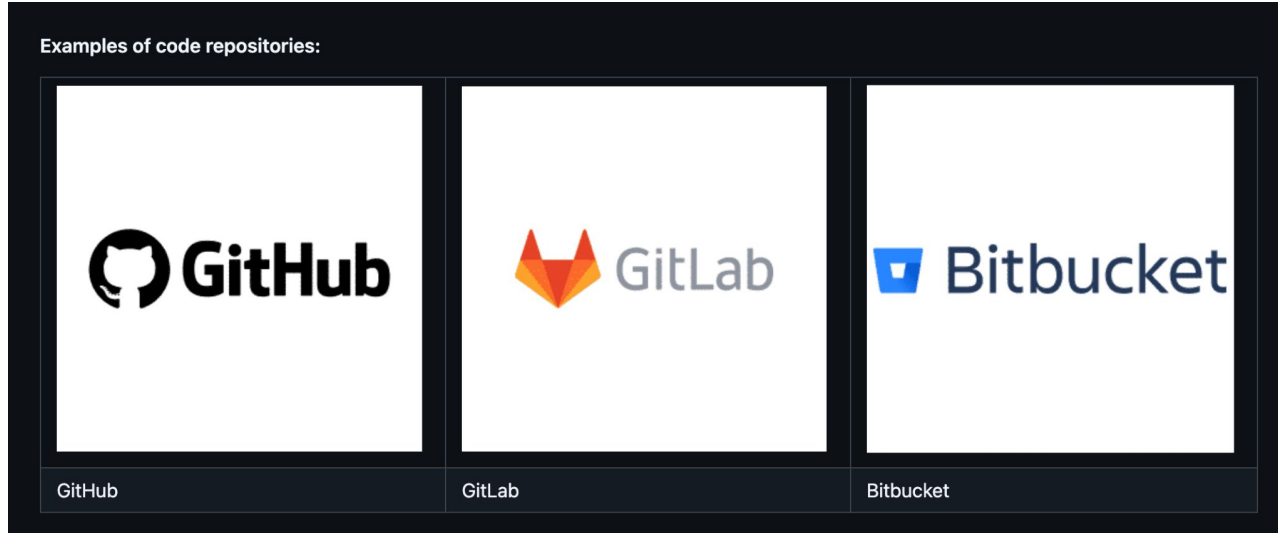
- **DVC:** Best for machine learning workflows where data versioning, pipeline management, and reproducibility are needed.
- **Git LFS:** Simplest for versioning large files directly with Git.
- **Git-Annex:** Flexible for managing large datasets with multiple storage options (i.e Dropbox).
- **Datalad:** Research-focused, built on Git-Annex for scientific data management. Pipeline management, retrieving and sharing data
- **MLflow:** Focused on tracking machine learning experiments and managing models.
- **Git Submodules:** Ideal for managing reusable Git-based code dependencies.

**Finding, citing, managing & licensing
<insert research output here> openly**

The background of the slide features two overlapping rectangular blocks. A large, light green rectangle occupies the lower-left and central portions of the slide. Overlapping the bottom-right corner of this green rectangle is a smaller, solid pink rectangle.

Open Source (Software)

Finding open source software



<https://github.com/nasa/Transform-to-Open-Science/tree/open-science-101>

Finding, citing,
managing &
licensing



Software Heritage



Open Source Development Network (OSDN)



SourceForge



Free and Open-Source Software Hub (FOSSHUB)



Googlecode



Comprehensive Perl Archive Network



PyPI



CRAN

Cool repositories

- Original Apollo 11 guidance computer (AGC) source code for Command Module (Comanche055) and Lunar Module (Luminary099).
<https://github.com/chrislgarry/Apollo-11>
- The linux kernel: <https://github.com/torvalds/linux>
- Awesome repos: <https://github.com/pawelborkar/awesome-repos>
- Secret book of knowledge:
<https://github.com/trimstray/the-book-of-secret-knowledge>

Citing open source software

How should open software be cited?

- Cite the software DIRECTLY, if possible.
- Sometimes there is an associated paper – this can alternatively be cited.
- Adding a link to the download page WILL NOT give the authors credit.

Citing open source software

Software purchased off-the-shelf:

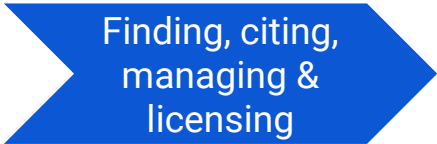
ProductName. Version. ReleaseDate. Publisher. Location.

- SuperScience. 1.2. December 2012. ResearchSoftware. Edinburgh, UK.

Software downloaded from the web:

ProductName. Version. ReleaseDate. Publisher. Location. DOI or URL. DownloadDate.

- OGSA-DAI REST. 4.2.1. December 2012. OGSA-DAI Project. <http://sourceforge.net/projects/ogsa-dai>. 27/04/2012.
- UltimateFFT. 2.4. December 2012. Fred Bloggs, EPCC, The University of Edinburgh, UK. <http://www.epcc.ed.ac.uk/ultimate-fft>. 27/04/2012.
- C implementation of Wu's color quantizer. 2. 1991. Xiaolin Wu, Department of Electrical & Computer Engineering, McMaster University, Hamilton, Ontario. <http://www.ece.mcmaster.ca/~xwu/cq.c>. 27/04/2012.



Finding, citing,
managing &
licensing

Citing open source Software

Software checked-out from a public repository:

ProductName. Publisher. URL. CheckoutDate. RepositorySpecificCheckoutInformation.

- OGSA-DAI REST. OGSA-DAI Project. <http://sourceforge.net/projects/ogsa-dai>. 27/04/2012.
Check-out: ogsa-dai/branck/ogsadai4.1/, revision 1657.

Software provided by a researcher:

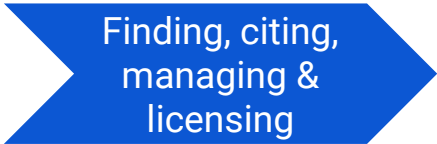
ProductName. Author. Location. ContactDetails. ReceivedDate.

- BestFFTroutine ever file. Fred Bloggs, EPCC, The University of Edinburgh, UK.
Fred.bloggs@epcc.ed.ac.uk. 27/04/2012.

Software checked out from Github:

Author/Organization. (Year). Title of the repository (Version if applicable). GitHub. URL

- OpenAI. (2024). ChatGPT repository (v1.2). GitHub. <https://github.com/openai/chatgpt>

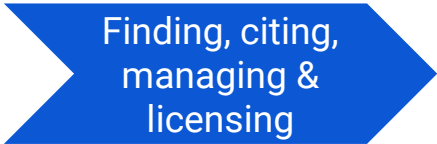


Finding, citing,
managing &
licensing

Citing research software: case studies

Creating and managing open (research) source software

- **Research Software** includes source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose
- Multiple initiatives such as the [Research Software Alliance \(ReSA\)](#), and the European [Open Science Cloud \(EOSC\)](#) are looking into improving research software management, reusability, and sustainability.
- Research software should be as open as possible but as closed as necessary



Finding, citing,
managing &
licensing

Software Management Plans

- All types of research software, open or closed, benefit from a Software Management Plan

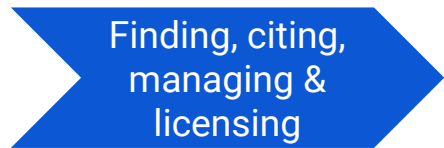
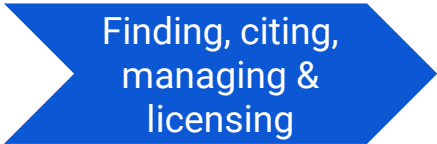


Figure 1. Software Management Plan requirements grouped by their focus.

Software management - Github repository

A good github repository should have:

- A good and descriptive Readme.md
 - Purpose
 - Installation guide
 - Documentation
 - User instructions
- A Contributing.md
 - How can and should people contribute?
 - What are the conditions for contributing?
- Version releases
- A license.md



Finding, citing,
managing &
licensing

Licenses

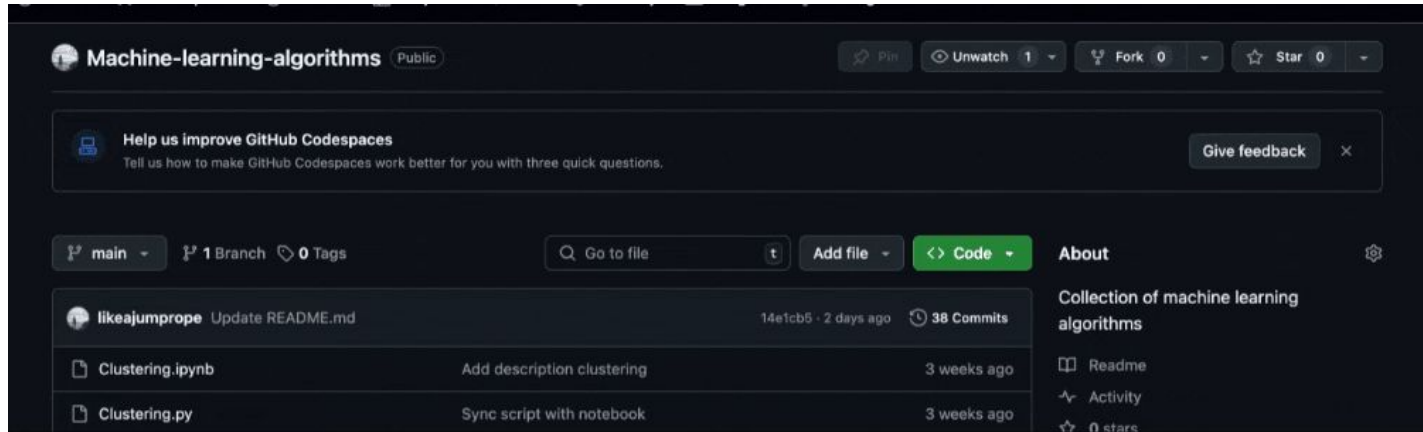
Licenses

Licenses

- You're under no obligation to create a license. However, without a license, the default copyright laws apply, meaning that you retain all rights to your source code and no one may reproduce, distribute, or create derivative works from your work.
- “Trap” lab or group repositories
 - Example: Lab PI creates repository without license, students contribute
 - Lack of license: Copyright of each contribution lies with the contributor
 - No one can make changes to the entire repository, INCLUDING LAB PI
 - This can be avoided using a license.
- Liability - most open source licenses protect the creator against liability claims

Licenses

- Licenses can be easily added to a github repository creating a file license.md
- Git will suggest license template text that you can copy and paste



Finding, citing,
managing &
licensing

Licenses

An overview about a large variety of licenses and comparisons are given here:

<https://choosealicense.com/appendix/>

<https://choosealicense.com/>

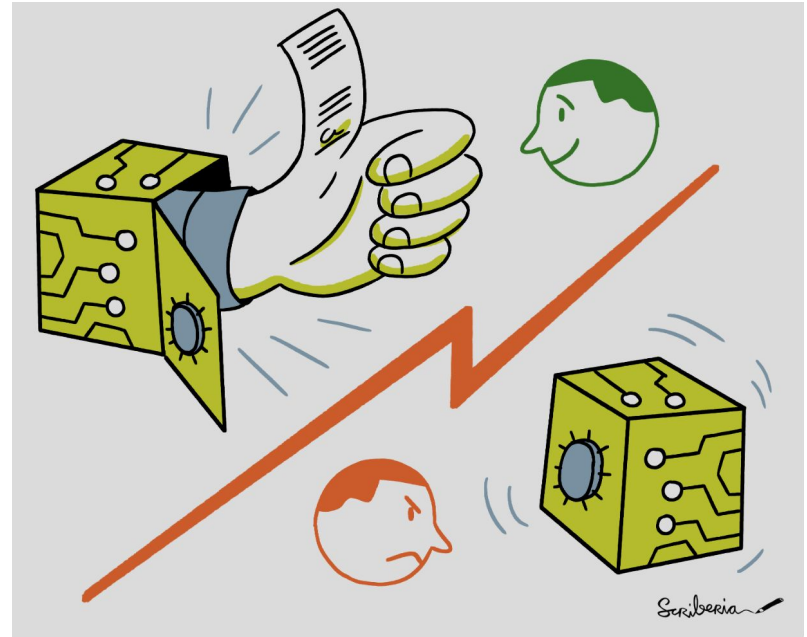


Fig. 35 Licensing. The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: [10.5281/zenodo.3332807](https://doi.org/10.5281/zenodo.3332807).

Finding, citing,
managing &
licensing

	Free			Non-free		
	Public domain & equivalents	Permissive license	Copyleft (protective license)	Noncommercial license	Proprietary license	Trade secret
Description	Grants all rights	Grants use rights, including right to relicense (allows proprietization, license compatibility)	Grants use rights, forbids proprietization	Grants rights for noncommercial use only. May be combined with share-alike.	Traditional use of copyright ; certain rights may or may not be granted	No information made public
For software	PD, Unlicense , CC0	BSD , MIT , Apache	GPL , AGPL	JRL , AFPL	Proprietary software , no public license	Private, internal software
For other creative works	PD, CC0	CC BY	CC BY-SA , FAL	CC BY-NC	Copyright , no public license, CC BY-ND	Unpublished

Finding, citing,
managing &
licensing

<https://en.wikipedia.org/wiki/Copyleft>

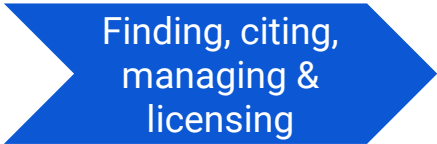
Versioning software

Versioning software on GitHub typically involves using **Git tags** to label specific commits with version numbers. Tags provide a snapshot of your codebase at a particular point in time, often corresponding to software releases. GitHub integrates these tags with release management tools, making it easier to distribute and document new versions of your software.

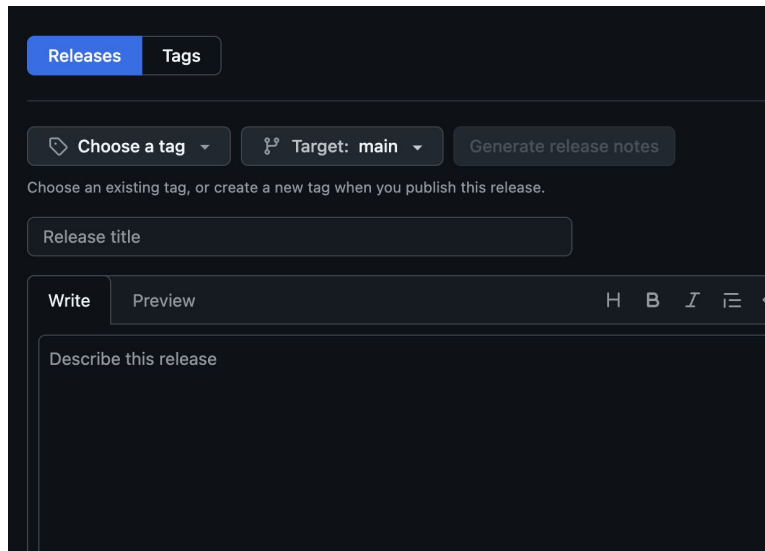
Versioning system:

MAJOR.MINOR.PATCH

- **MAJOR**: Increments for breaking changes.
- **MINOR**: Increments for backward-compatible feature additions.
- **PATCH**: Increments for bug fixes.



Finding, citing,
managing &
licensing



The screenshot shows the GitHub Releases interface. At the top, there are two tabs: 'Releases' (active) and 'Tags'. Below the tabs, there are three buttons: 'Choose a tag' (with a dropdown arrow), 'Target: main' (with a dropdown arrow and a branch icon), and 'Generate release notes'. Below these buttons, there is a text input field for 'Release title'. At the bottom, there is a text area for 'Describe this release' with a 'Write' tab and a 'Preview' tab. The 'Write' tab is active, and the text area contains the placeholder text 'Describe this release'.

Activity 1: Create a good demo github repo and add a license to your github repo

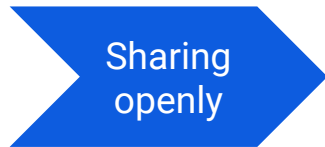
- (Fork my repo)
- Add a good Readme.md - description, purpose
- Add a license
- Make a version release

**Sharing <insert
research output here>
openly**

Zenodo



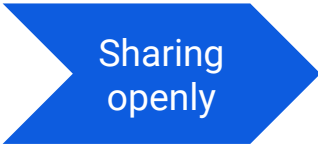
- Free and open platform for sharing, preserving and citing research outputs
- Developed by European Organisation for Nuclear Research CERN, supported by the [OpenAIRE](#) initiative
- Used to store and share:
 - Software
 - Publications
 - Datasets
 - Other types of research software



Zenodo



- **Open Access Repository:** enables researchers to share their work openly, promoting transparency and collaboration.
- Every upload is assigned a **Digital Object Identifier (DOI)**, making it easy to cite and permanently reference the work.
- **Support for All Research Outputs:** Accepts a variety of file types, including datasets, code, publications, posters, presentations, and multimedia.
- **Versioning:** Allows for versioning of submissions, ensuring that updates to datasets or software can be tracked while maintaining the ability to reference specific versions.
- **Integration with GitHub:** Automates the archiving of GitHub repositories. Researchers can push their code or project directly from GitHub to Zenodo for publication and DOI generation.
- **Open Licenses:** Supports the application of open licenses (e.g., CC-BY, GPL) to promote reuse and clarity about usage rights.
- **Long-term Preservation:** Ensures long-term preservation of research outputs by leveraging CERN's robust storage infrastructure.
- **Metadata Standards:** Complies with metadata standards to improve discoverability and interoperability.
- Compatible with EU Standards



Sharing
openly

How to use Zenodo for sharing



1. Create an Account

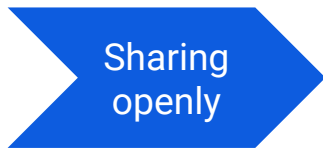
- Go to [Zenodo](#) and sign up using your ORCID or email address

2. Upload Research Outputs

- Click **Upload** and fill out the metadata fields (title, authors, description, etc.).
- Attach the files you want to share (datasets, code, PDFs, etc.).
- Choose a license (e.g., Creative Commons, GPL).
- Publish the record to generate a DOI.

3. Share the DOI

- Use the DOI to cite your research outputs in publications, presentations, or online profiles.



Linking your Zenodo to your Github



1. Create an Account

- Go to [Zenodo](#) and sign up using your ORCID or email address

2. Integrate GitHub with Zenodo

- Link your GitHub account to Zenodo .
- Switch the flip: Enable archiving for specific repositories.
- When you create a release on GitHub, Zenodo automatically archives the repository and assigns a DOI.

3. Share the DOI

- Use the DOI to cite your research outputs in publications, presentations, or online profiles.

Sharing
openly

A screenshot of the Zenodo user interface. On the left is a 'Settings' sidebar with options: Profile, Change password, Security, Notifications, Linked accounts (highlighted in blue), Applications, and GitHub. The main content area is divided into two panels. The top panel, 'Linked accounts', shows 'GitHub' with a green checkmark and 'ORCID' with a green checkmark. The bottom panel, 'Enabled Repositories', shows a repository 'likeajumprope/Git_course' with a DOI '10.5281/ze'. Above this panel is a section titled '1 Flip the switch' with a toggle switch set to 'ON' and the text '(example)'.

Settings

- Profile
- Change password
- Security
- Notifications
- Linked accounts**
- Applications
- GitHub

Linked accounts

Tired of entering password for Zenodo every time

GitHub ✓
Software collaboration platform.

ORCID ✓
Connecting Research and Researchers.

OpenAIRE
Open Science Services.

1 Flip the switch

Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software.

ON (example)

Enabled Repositories

likeajumprope/Git_course DOI 10.5281/ze

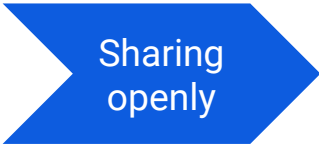
Other (often institutional) file sharing repositories

Figshare

- [Figshare](#) is a cloud-based platform designed for storing, sharing, and managing **research outputs of all kinds**, including datasets, figures, images, code, and publications.
- However, **free accounts have limited storage**.

Dryad

- [Dryad](#) is an open-access repository specifically designed for **research data**. It focuses on datasets that accompany peer-reviewed publications, ensuring that they are preserved and accessible.
- However **limited to data sets** and it relies on a subscription model.



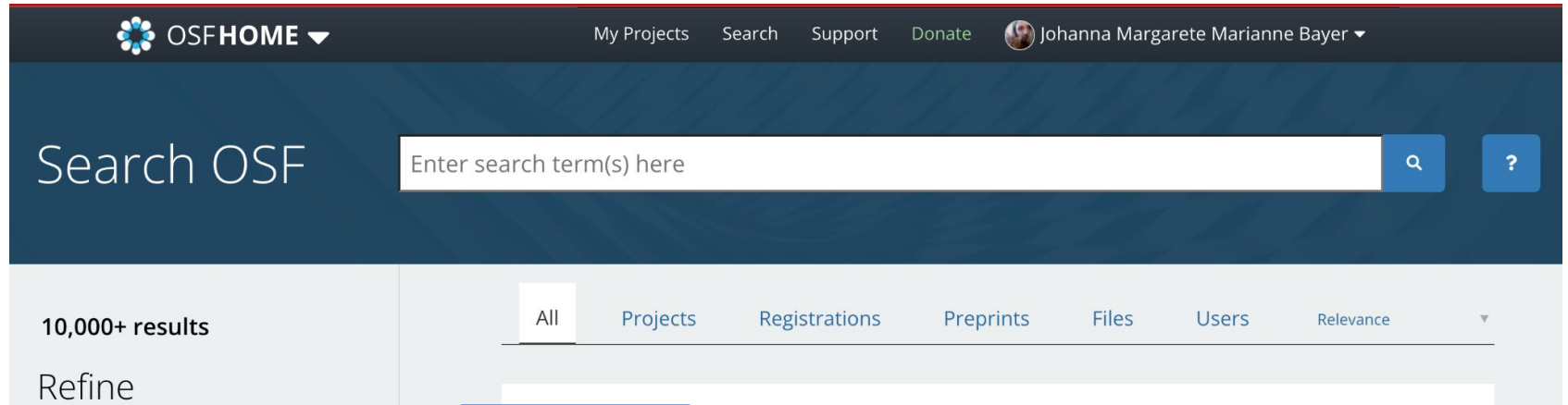
Sharing
openly

Activity 2: create Zenodo profile

- Link your Github to Zenodo
- Link your Github release to your zenodo profile and create a DOI
- Add the DOI as a badge on your Github repo

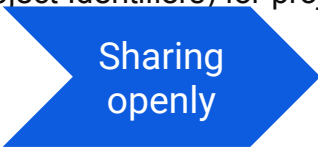
Open Science Framework (OSF)

- The **Open Science Framework (OSF)** is a free, open-source platform designed to support collaboration, transparency, and reproducibility throughout the research lifecycle. Researchers, educators, and professionals use OSF to organize and share their projects, data, and research outputs.
- It is maintained by the **Center for Open Science (COS)**, a nonprofit organization dedicated to increasing the openness and accessibility of scientific research.



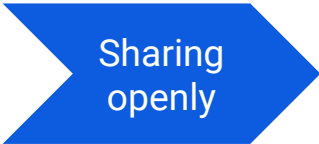
OSF

- **Project Management:**
 - Create and manage research projects with a centralized workspace
 - Share and organize files, data, code, and other materials.
- **Collaboration:**
 - Add collaborators with flexible permissions (read-only, read/write, admin).
 - Work with team members from different institutions.
- **File Storage and Integration:**
 - Store files (up to 5 GB per file for free users), from storage services like Google Drive, Dropbox, and GitHub.
- **Pre-registration:**
 - Pre-register research plans and hypotheses to promote transparency and reduce publication bias.
- **Version Control:**
 - Track changes to files and materials, maintaining a clear history of updates.
- **Preprints:**
 - Publish preprints (early research drafts) and link them to your OSF project.
 - Access preprint services like **PsyArXiv**, **SocArXiv**, **BioRxiv**, and others.
- **DOI Assignment:**
 - Generate DOIs (Digital Object Identifiers) for projects and files to make them citable.



Sharing
openly

	OSF	Zenodo	Github
Purpose	Project management, collaboration, preprints	Data and software sharing	Code and small script version control
Open Access	yes	yes	Depends on repository
Pre-registration	yes	no	no
File storage	5 GB limit per file	50 GB limit per upload	File size and repos size limits
DOI supports	yes	yes	no
Integration with other tools	Zotero, Mendeley, Github, Dropbox	Github, OpenAire	Many others



Sharing
openly

Good coding practices

The background of the slide features two large, overlapping rectangular blocks. A light green rectangle occupies the left and central portions of the frame, while a pink rectangle is positioned on the right, partially overlapping the green one. The text 'Good coding practices' is centered horizontally and partially overlaid by the green rectangle.

Activity 3:

What are good coding practices for you?
What benefits and challenges do you see in
teaching good coding practices to
researchers?

Teaching good coding practices across platforms

Research code is lacking:

- Modularity by defining **functions**
- Documentation
 - Docstrings in Python (Functions!)
 - Roxygen (Functions!)
- Tests
 - Pytest (Functions!)
 - test_that(Functions!)

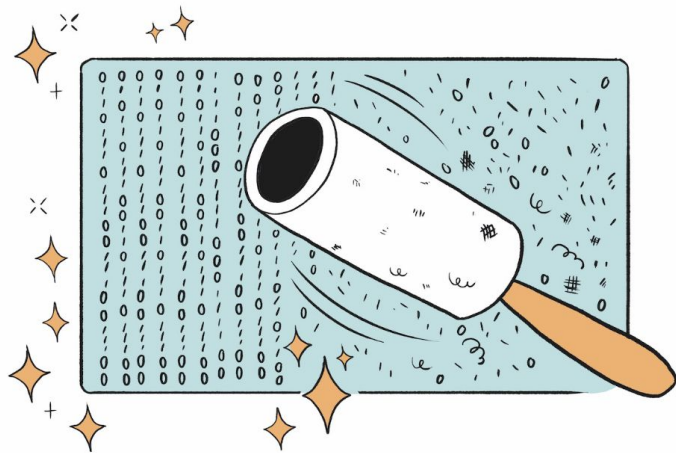


Fig. 75 The Turing Way project illustration by Scriberia. Used under a CC-BY 4.0 licence. DOI: [10.5281/zenodo.13882307](https://doi.org/10.5281/zenodo.13882307).

Scriberia

Good coding
practices

Good coding practices: Meaningful variable names

- Descriptive names for scripts and functions instead of code comments

```
# Bad
def calc(a, b):
    return a * b


# Good
def calculate_area(length, width):
    return length * width
```

Good coding
practices

Good coding practices - Documentation

Documentation

- Write Docstrings
 - Python: [Sphinx](#), [pdoc](#)
 - JavaScript: [JSDoc](#)
 - C++: [Doxygen](#)
- Github repos: Add a Readme.md for detailed instructions for installation
Contributing.md for contribution
- License.md for licensing



Good coding
practices

Create docstrings in Python and R

```
def calculate_bmi(weight, height):  
    """  
    Calculate the Body Mass Index (BMI) of a person.  
  
    Args:  
    weight (float): Weight in kilograms  
    height (float): Height in meters  
  
    Returns:  
    float: The calculated BMI  
  
    Raises:  
    ValueError: If weight or height is not positive  
    """  
    if weight <= 0 or height <= 0:  
        raise ValueError("Weight and height must be positive values")  
  
    return weight / (height ** 2)
```

```
#' Add together two numbers  
#'  
#' @param x A number.  
#' @param y A number.  
#' @returns A numeric vector.  
#' @examples  
#' add(1, 1)  
#' add(10, 1)  
add <- function(x, y) {  
  x + y  
}
```

Add together two numbers

Description

Add together two numbers

Usage

```
add(x, y)
```

Arguments

x A number

y A number

Value

The sum of x and y

Examples

```
add(1, 1)  
add(10, 1)
```

Good coding
practices

Good coding practices: Code Comments

Rule 1: Comments should not duplicate the code.

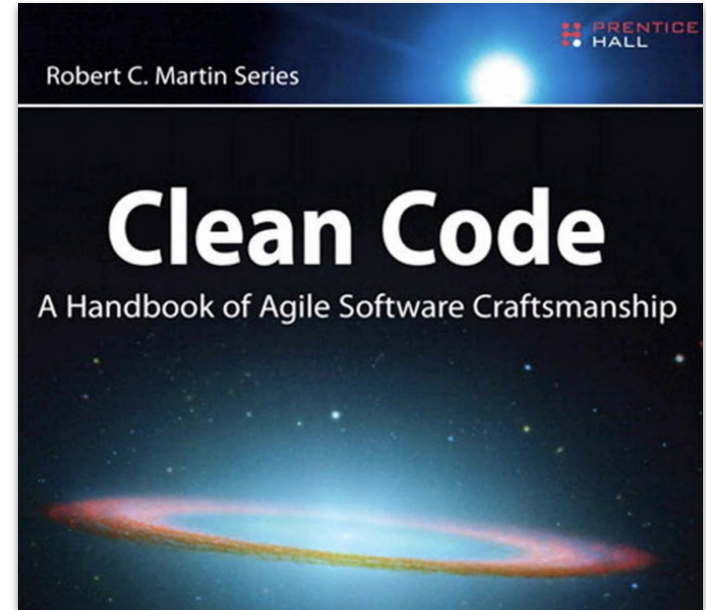
```
i = i + 1;           // Add one to i
```

Rule 2: Good comments do not excuse unclear code.

```
private static Node getBestChildNode(Node node) {  
    Node n; // best child node candidate  
    for (Node node: node.getChildren()) {  
        // update n if the current state is better  
        if (n == null || utility(node) > utility(n)) {  
            n = node;  
        }  
    }  
    return n;  
}
```

Rule 3: If you can't write a clear comment, there may be a problem with the code.

```
/* You are not expected to understand this.  
*/  
if(rp->p_flag&SSWAP) {  
    rp->p_flag = ~SSWAP;  
    aretu(u.u_ssav);  
}  
...
```



<https://stackoverflow.blog/2021/12/23/best-practices-for-writing-code-comments/>

Good coding
practices

Good coding practices: Code Comments

Rule 4: Explain unidiomatic code in comments.

```
final Object value = (new JSONTokener(jsonString)).nextValue();  
// Note that JSONTokener.nextValue() may return  
// a value equals() to null.  
if (value == null || value.equals(null)) {  
    return null;  
}
```

Rule 5: Provide links to the original source of copied code, and include external references

```
// Magical formula taken from a stackoverflow post, reputedly related to  
// human vision perception.  
return (int) (0.3 * red + 0.59 * green + 0.11 * blue);
```

Rule 6: Add comments when fixing bugs.

Rule 7: Use comments to mark incomplete implementations.

```
/* TODO(hal): We are making the decimal separator be a period,  
regardless of the locale of the phone. We need to think about  
how to allow comma as decimal separator, which will require  
updating number parsing and other places that transform numbers  
to strings, such as FormatAsDecimal
```

90% of all code comments:



Good coding
practices

Good coding practices: Readability

```
# Less readable
def is_prime(n):
    return n > 1 and all(n % i for i in range(2, int(n**0.5) + 1))

# More readable
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True
```

Good coding
practices

Good coding practices: Error handling

- Fail early and loudly
- Good error messages
 - Clearly pinpoints the problem.
 - Points a user to next steps: items to check, or other steps they can take to further understand or fix the problem.
 - Uses jargon appropriately, and keeps its target audience in
 - honest about what it knows and does not know.



Good coding
practices

Good coding practices: Single responsibility principle

```
# Bad: Multiple responsibilities
class User:
    def __init__(self, name):
        self.name = name

    def save_to_database(self):
        # Database logic here
        pass

    def send_welcome_email(self):
        # Email logic here
        pass
```

```
# Good: Separated responsibilities
class User:
    def __init__(self, name):
        self.name = name

class UserDatabase:
    def save_user(self, user):
        # Database logic here
        pass

class EmailService:
    def send_welcome_email(self, user):
        # Email logic here
        pass
```

Good coding
practices

Good coding practices: Use of linting tools

E.g pylint

```
# Run this command in your terminal to use a linter (e.g., pylint)
# pylint your_file.py

# Example of code that a linter might flag
x = 5
y = 10
z = x+y # Linter might suggest adding spaces around the '+' operator

# Corrected version
x = 5
y = 10
z = x + y
```

Good coding
practices

Good coding practices - automation



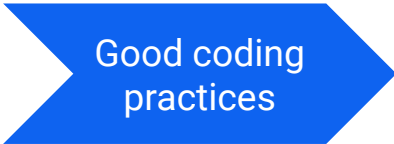
Snakemake (python)

- Allows to schedule scientific workflows, data analysis pipelines, especially in bioinformatics and computational science.
- Install using conda

Submit using:

```
$ snakemake --slurm --default-resources slurm_account=<your SLURM account>  
slurm_partition=<your SLURM partition>
```

Or in bash script

A blue arrow pointing to the right, containing the text "Good coding practices" in white.

Good coding
practices

Good coding practices - Testing

Research code is chronically
undertested

- obstacle might lie in code
structure

```
import unittest

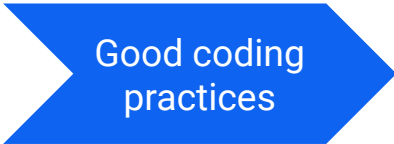
def add_numbers(a, b):
    return a + b

class TestAddNumbers(unittest.TestCase):
    def test_add_positive_numbers(self):
        self.assertEqual(add_numbers(2, 3), 5)

    def test_add_negative_numbers(self):
        self.assertEqual(add_numbers(-1, -1), -2)

    def test_add_zero(self):
        self.assertEqual(add_numbers(5, 0), 5)

if __name__ == '__main__':
    unittest.main()
```



Good coding
practices

R example: documentation

```
#' Add together two numbers
#'  
#' @param x A number.  
#' @param y A number.  
#' @returns A numeric vector.  
#' @examples  
#' add(1, 1)  
#' add(10, 1)  
add <- function(x, y) {  
  x + y  
}
```

Add together two numbers

Description

Add together two numbers

Usage

```
add(x, y)
```

Arguments

x A number
y A number

Value

The sum of x and y

Examples

```
add(1, 1)  
add(10, 1)
```

Python example: tests

```
home12ng1 /root/.johbay/22222/22222/home12ng1  
johbay@ip-213-71 make_dataframes % tree  
.  
├── pytest.ini  
├── requirmentes.txt  
├── scr  
│   └── create_dataframe.py  
├── setup.py  
└── test  
    ├── __init__.py  
    └── test.py
```

create_dataframe.py

```
def create_dataframe():
```

Merge operation
here

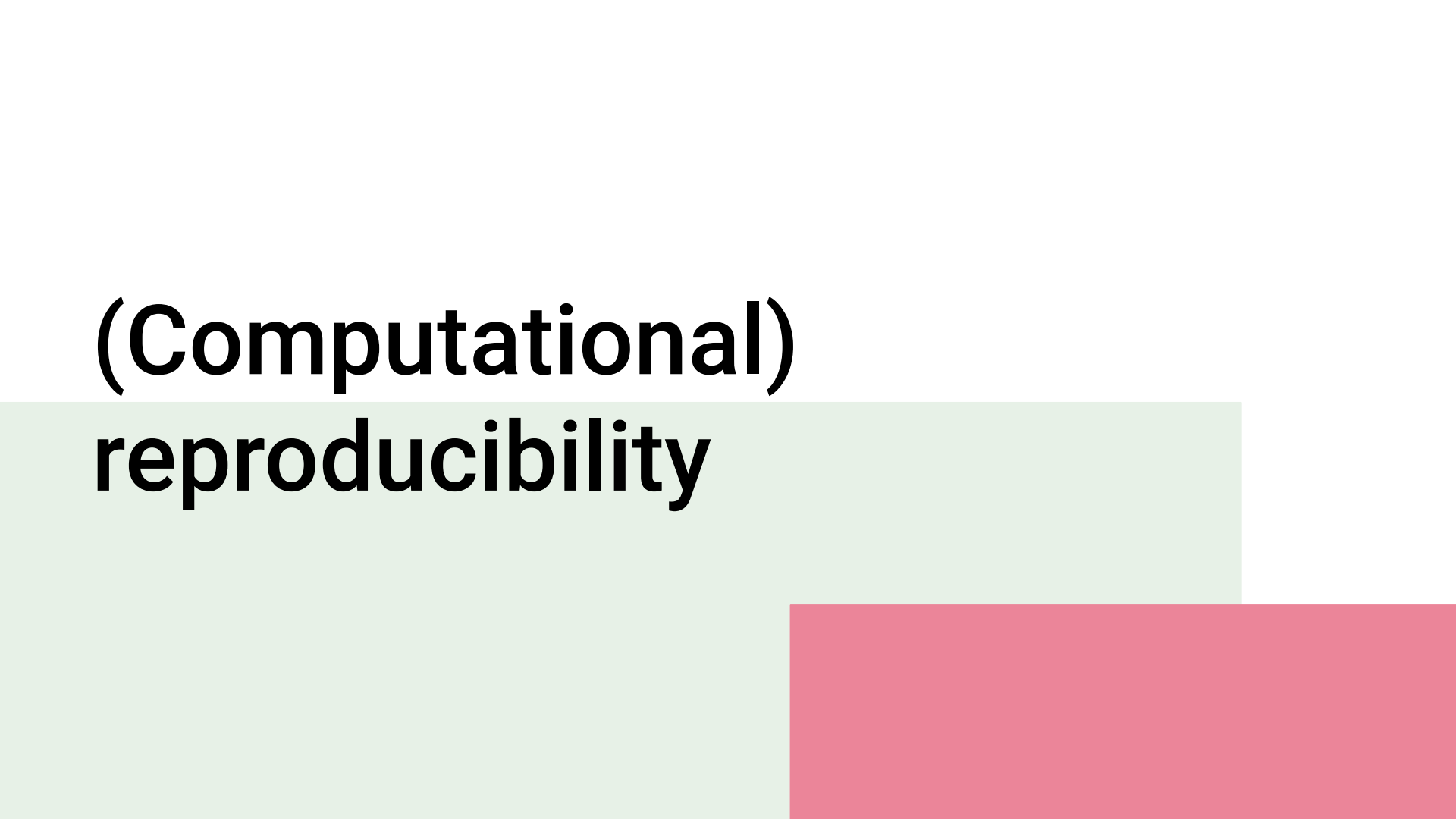
```
)
```

test.py

```
def check_no_nans(df: pd.DataFrame):  
    assert not df.isnull().values.any(), "DataFrame contains NaN values"  
  
def test_no_nans_in_dataframe():  
    df = create_dataframe()  
    check_no_nans(df)
```

Good coding
practices

**(Computational)
reproducibility**

The background features a light green rectangular block on the left and a pink rectangular block on the right, both positioned below the text.

Example of non computational reproducibility

In Neuroimaging, the reproducibility of results have been shown to be dependent on (Everything Matters):

- Computational environments matter (Glatard et al., 2015); eg. Operating system
- Tool selection matters (Tustison et al., 2014; Dickie et al., 2017);
- Tool version matters (Dickie et al., 2017);
- Statistical model matters (Tan et al., 2016);

Computational reproducibility and false positives

Open access, freely available online

Essay

Why Most Published Research Findings Are False

John P.A. Ioannidis

- When the power ($1 - \beta$) of a statistical test falls, the probability of a significant result is more likely to be due to false positives than to an actual result.
- This dynamic is more likely in fields with
 - The number of test
 - Small sample sizes
 - Small effect sizes
 - Flexibility in the description of methods and design

Computational
reproducibility



Activity 4:

From your own experience/work:

What enhances computational
reproducibility?

What hinders computational reproducibility?

Capturing computational environments

What is reproduced? Interaction style	Graphical	Command line
	Software & versions	CONDA
Entire system		

Computational
reproducibility

Binder (<https://mybinder.org/>)

Binder is a service which generates fully-functioning versions of projects from a git repository and serves them on the cloud. These “binderized” projects can be accessed and interacted with by others via a web browser. In order to do this, Binder requires that the software (and, optionally, versions) required to run the project are specified. Users can make use of Package Management Systems or Dockerfiles to do this if they so desire.

How it works

- Enter you repo information
- Binder builds a docker image of it
- A JupyterHub will host your repo
- Comes with badge and link to share



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

Computational
reproducibility

New to Binder? Get started with a [Zero-to-Binder tutorial](#) in Julia, Python, or R.

Build and launch a repository

GitHub repository name or URL

GitHub ▼

GitHub repository name or URL

Git ref (branch, tag, or commit)

HEAD

Path to a notebook file (optional)

Path to a notebook file (optional)

File ▼

launch

Copy the URL below and share your Binder with others:

Fill in the fields to see a URL for sharing your Binder.

📋

Expand to see the text below, paste it into your README to show a binder badge:



Computational
reproducibility

Package management systems (conda, venv etc)

Package Management Systems are tools used to install and keep track of the software (and critically versions of software) used on a system and can export files specifying these required software packages/versions. The files can be shared with others who can use them to replicate the environment, either manually or via their Package Management Systems.

In a research context:

Allows to export and share environment settings via:

```
$conda env export > environment.yml
```

And re-create projects using:

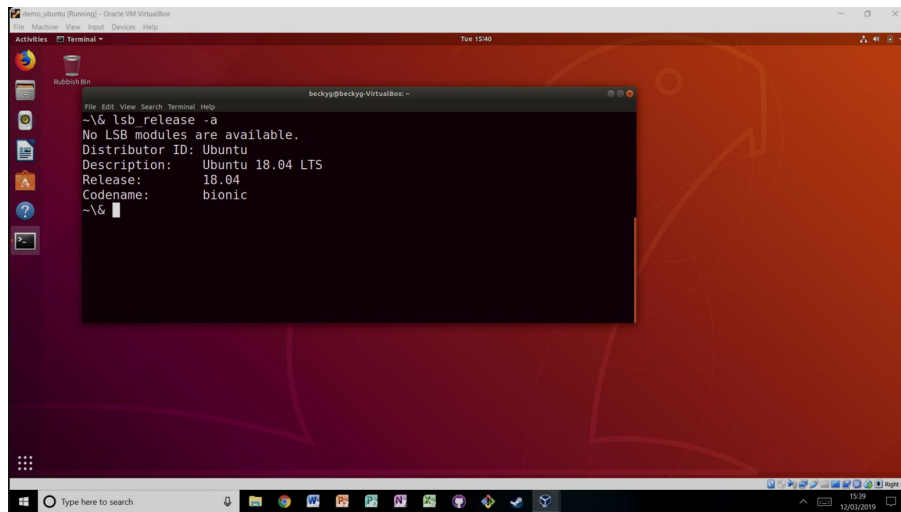
```
$conda create --name Project_Two --clone Project_One
```

A blue arrow pointing to the right, containing the text "Computational reproducibility".

Computational
reproducibility

Virtual machines (for reproducible research)

VirtualBox allows to create virtual machines that can be shared exporting a single file. A second researcher can import the shared file and run an experiment in the same environment.



<https://book.the-turing-way.org/reproducible-research/renv/renv-virtualmachine>

Computational
reproducibility

Virtual machines (for reproducible research)

Vagrant “enables users to create and configure lightweight, reproducible, and portable development environments”. In this context, an environment is a virtual machine (its CPUs, RAM, networking and so on) and the machines state (operating system, packages).

Vagrant can set up virtual machines **using text scripts**, instead of pointing and clicking through a graphical user interface. This makes it particularly useful for automating the process of setting up virtual machines and making that process reproducible.

A blue arrow pointing to the right, containing the text 'Computational reproducibility' in white.

Computational
reproducibility

Containers

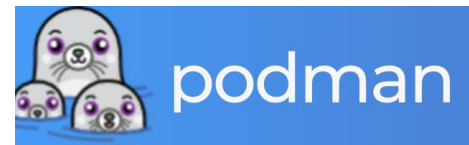
- contain the individual components they need in order to operate the project they contain -> performance boost

Common container formats are:

- Docker (<https://www.docker.com/>)
- Podman (<https://podman.io/>)
- Singularity (Apptainer)

More on containers:

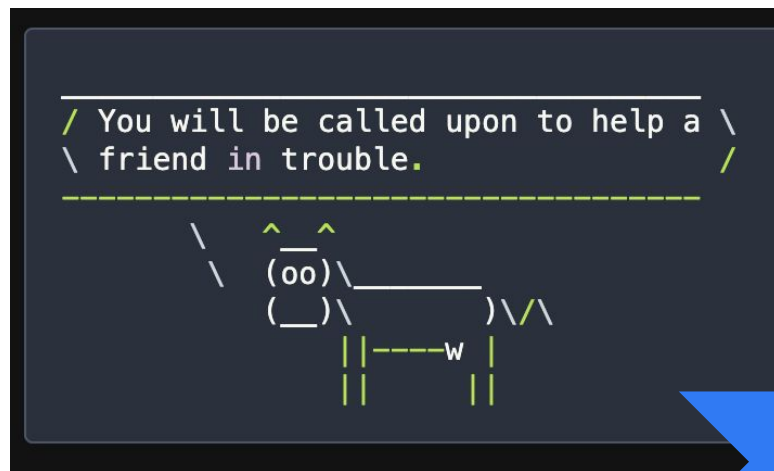
[https://book.the-turing-way.org/reproducible-research/r
env/renv-containers](https://book.the-turing-way.org/reproducible-research/r
env/renv-containers)



Computational
reproducibility

Singularity (Apptainer)

- Docker historically requires root access
- Singularity (which does not) as alternative on HPC systems
- The Open Source version **Apptainer** is a fork of Singularity and is run by the Linux foundation, after a change in licensing by Singularity.



Computational
reproducibility

Activity 5:

What system would you prefer?

What challenges and disadvantages do you see (HPC specific and general)

Open Science Reproducibility Checklist

Open Science

Reproducibility Checklist

- *Is your data available to others?*
- *Is 'reproducibility' included in your Data Management Plan?*
- *Does your research folder include a 'readme' file, explaining the context of the research, the file structure, and procedures?*
- *Does your research folder include a codebook, explaining all variables in your research?*
- *Are your analysis steps or scripts well described and available to others?*
- *Does your project minimise the effort it takes to reproduce your research?*
- *Is your research software available to others by attribution of a licence?*
- *Will your results be reproducible in the future?*
- *Has the reproducibility of your work been verified by a colleague?*
- *Do you know whom to contact if you need help with reproducibility?*

**Open and reproducible
science in complex
research projects**

Data (and code) is not available upon request

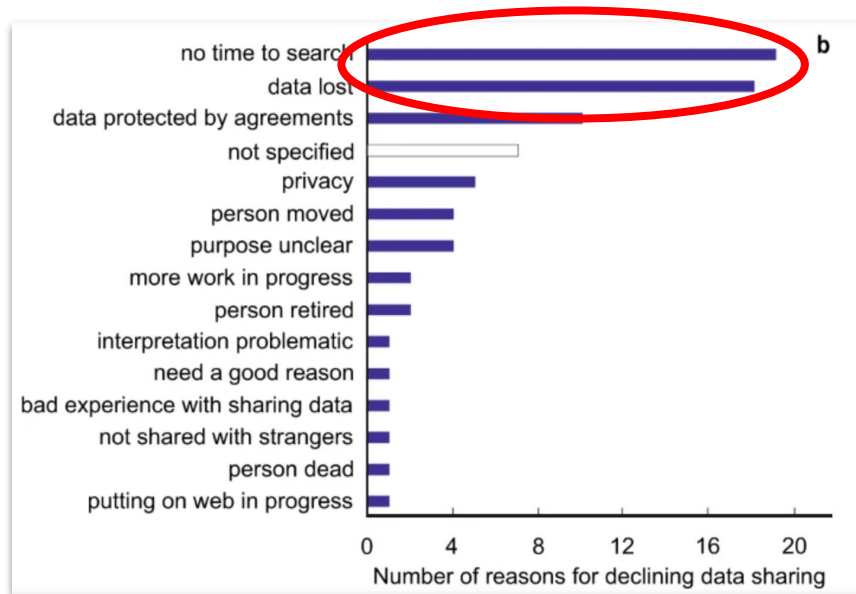
52 articles that included the statement “Data is available upon request” were asked to share their data

- 26.9% shared their data

Reasons to not share the data:

- Legal or ethical issues
- Some authors simply stopped replying
- “Data can only be shared via University platform that does not yet exist”

Hussey, 2023



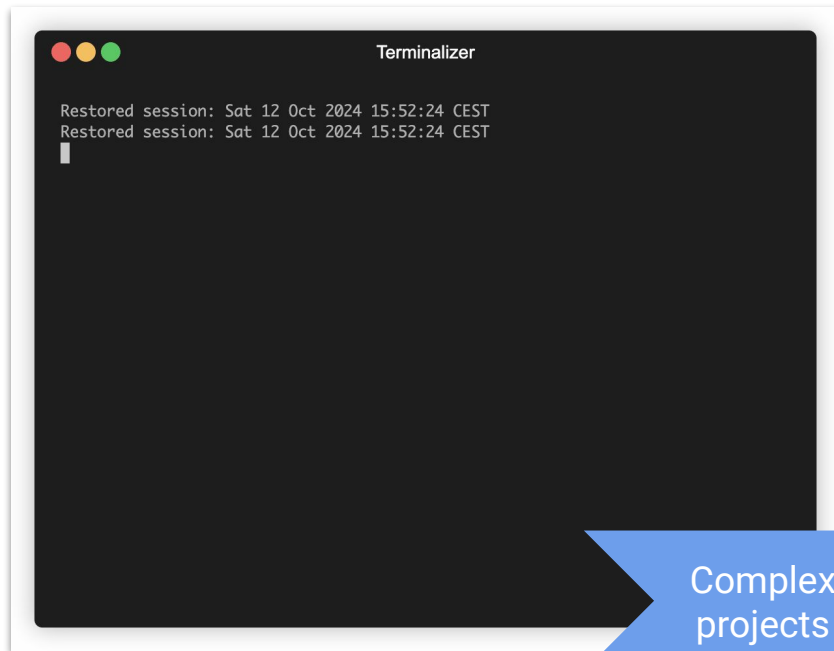
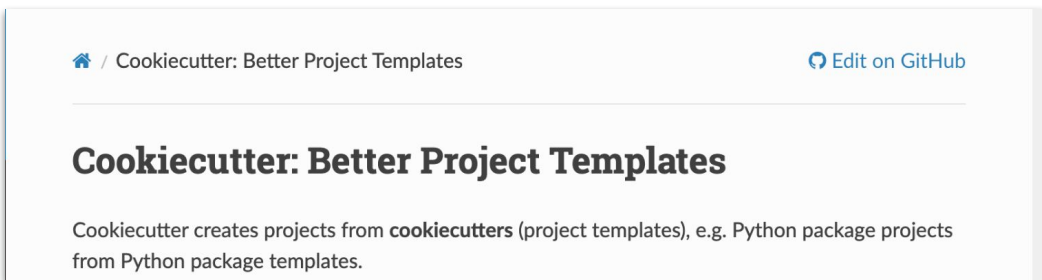
Tedersoo, 2021

Complex
projects

Project folder structure

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

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



Complex projects

Result - cookiecutter project template

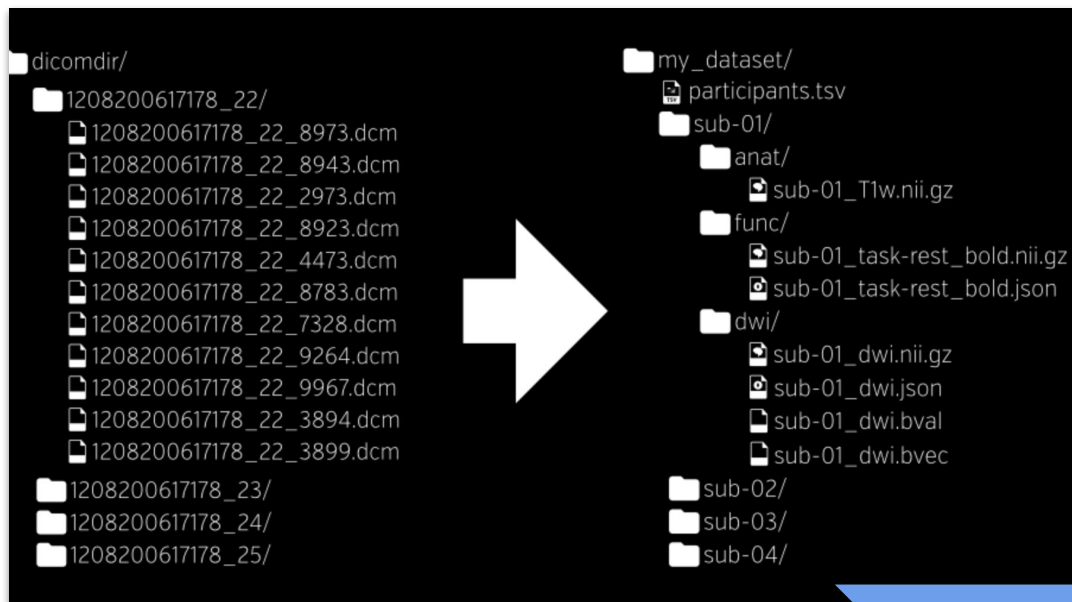
- Code and data version for good file organisation

```
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
```

Complex
projects

File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures (<https://bids.neuroimaging.io/>):



Slides by courtesy of Dr. Remi Gau, adapted

Complex
projects

File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures (see also chapter in [Handbook](#))

- 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

Complex
projects

File organisation: (stealing from the) BIDS standard

Brain Imaging Derived Structures (see also chapter in [Handbook](#))

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.

(Stealing from) the BIDS standard

- Metadata
 - .json files
 - Have the same name as the original file and contain information in key-value pair format

sub-001_population-cross_mri.json

- Example:

```
{  
  "population": "cross-sectional",  
  "atlas": "desikan",  
  "fixed_effects": "age, sex",  
  "random_effects": "site",  
  "likelihood": "normal",  
  "function": "fit"  
}
```

File organisation: learn basic bash/unix powershell commands

- Scripting bash/unix command makes your file organisation reproducible
- Examples:

- Get a list of all paths to all control files in all subfolders and save them in a file

```
$ find /path/to/search -type f -name "*your_string*" > file.txt
```

- Sync folders (between server and local):

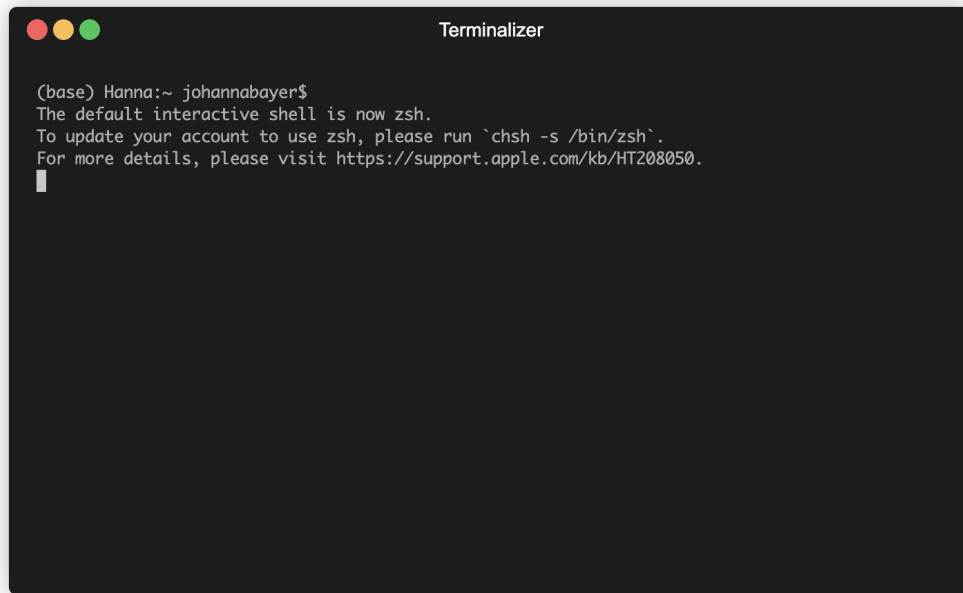
```
$ rsync -avz user@192.168.1.10:/remote/directory/ /local/directory/
```

- Create symbolic links etc:

```
$ ln -s /path/to/original /path/to/symlink
```

Choose your coding platform for your analysis

- Python:
 - Environments are important
 - Jupyter Notebooks
 - Scripts
- R/R studio
 - Projects
 - R Notebooks and Markdown
 - workflower
- Matlab
 - Matlab projects
 - Matlab Live Scripts



```
(base) Hanna:~ johannabayer$  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
█
```

Consideration for stable computing platform

Complex
projects

The repro pub

Published July 15, 2019 | Version v1

Conference paper

 Open

The ReproPub: A hybrid research object for supporting publication-level re-execution and generalization of neuroimaging research findings

Kennedy, David N¹ 

Show affiliations

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

Complex
projects

Neurolibre

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

A reproducible benchmark of resting-state fMRI denoising strategies using fMRIPrep and Nilearn

Jupyter Notebook Python Submitted 13 May 2023 • Published 19 June 2023



A reproducible benchmark of resting-state fMRI denoising strategies using fMRIPrep and Nilearn

DOI: [10.55458/neurolibre.00012](https://doi.org/10.55458/neurolibre.00012)

Reproducible Preprint

Hao-Ting Wang^{1,7}, Steven L. Meisler^{2,3}, Hanad Sharmarke¹, Natasha Clarke¹, François Paugam⁴, Nicolas Gensollen⁵, Christopher J. Markiewicz⁶, Bertrand Thirion⁵, and Pierre Bellec^{1,7}

 LIVING PREPRINT

 Download PDF

 Preprint repository

 Technical screening

 Repository archive

 Data archive

 HTML archive

 Container archive

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

Text editor	pro	con
MS Word/Google docs	<ul style="list-style-type: none">- Plugins for most reference managers- good (?) platform for getting feedback/review- (Version controlled)	<ul style="list-style-type: none">- Figures need to be imported each time
Markdown	<ul style="list-style-type: none">-Integration with R (markdown, knitr)-In theory, fully reproducible (code and paper)	<ul style="list-style-type: none">- No review option
Overleaf (online)	<ul style="list-style-type: none">- Bibtex/Latex- Pro version has direct git and Dropbox link- Math support, templates- review option	<ul style="list-style-type: none">- steeper learning curve
LaTeX (Desktop)	<ul style="list-style-type: none">- Free- Bibtex file- Entirely version controllable- Math support- Create a figure folder that you save your figs in	<ul style="list-style-type: none">-Very steep learning curve-No review option

Complex projects

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

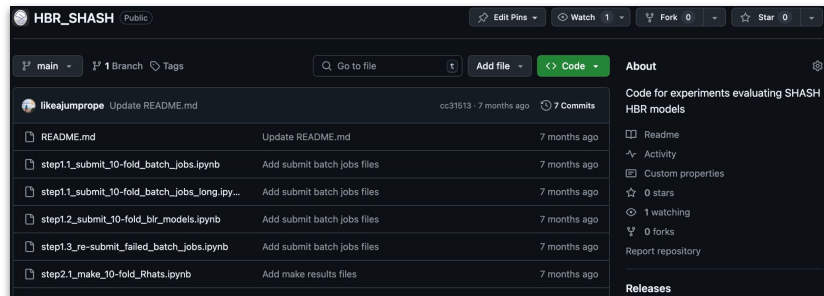
Downside

- Internet dependency (except offline mode)

Complex
projects

Publishing non-traditional research outputs

- Code:
 - JOSS (<https://joss.theoj.org/>)
 - Neurolibre (<https://neurolibre.org/>)
- Preprints:
 - OSF, arXiv
- Data, ppts:
 - Zenodo
 - OSF
- Publish your code on Github has part of your publication, with a DOI



Non-Gaussian Normative Modelling With Hierarchical Bayesian Regression

Code for experiments evaluating SHASH HBR models as described in:

de Boer, A. A., Bayer, J. M. M., Kia, S. M., Rutherford, S., Zabihi, M., Frazza, C., Barkema, P., Westlye, L. T., Andreassen, O. A., Hinne, M., Beckmann, C. F., & Marquand, A. (2024). Non-Gaussian normative modelling with hierarchical Bayesian regression. *Imaging Neuroscience*, 2, 1–36.

Complex
projects

Journal for Open Source Software

- Fully run via Github issues & editorial bot, fully transparent review
- Focus on Software, papers < 1 page
- Looking for editors!!

[PRE REVIEW]: reflectorch: a deep learning package for X-ray and neutron reflectometry #7089

Open editorialbot opened this issue on Aug 8 · 20 comments

editorialbot commented on Aug 8 · edited ▾ Member ...

Submitting author: @valentinsingularity (Valentin Munteanu)
Repository: <https://github.com/schreiber-lab/reflectorch>
Branch with paper.md (empty if default branch):
Version: v1.2.1
Editor: @likeajumprope
Reviewers: Pending
Managing EIC: Chris Vernon

Assignees
 likeajumprope

Labels
Dockerfile pre-Track: 5 (DSAIS)

Complex
projects

The elife model

- Elife shifted to a model of “reviewed pre-prints”
- Their reviewing model to be fully open and there is “no hard rejection”
- Once a paper is accepted for review, it will be published, with all comments from the review and in various versions

The web of science stopped indexing elife papers on Oct 23, 2024

This means that elife will not receive an impact factor this year

Thoughts?

(<https://elifesciences.org/inside-elifesciences/14e77604/elifesciences-new-model-what-is-a-reviewed-preprint>)

<https://elifesciences.org/inside-elifesciences/16afe6ec/update-on-elifesciences-indexing-status-at-web-of-science>



Complex
projects

Publishing research software

Measuring the impact of your research software

- Citations alone might not be the best measure for your research software
- Issues:
 - Often, in a paper only a link to the software is provided. This will NOT count towards the citations of your software
 - Github download stats (Insights) are only available for the owner of the repo. This is an issue when your software is being hosted in a lab account
- Consider also:
 - Zenodo views and downloads
 - Altmetrics
 - Host workshops and count attendees



Complex
projects

Publish various sections of your research

Zenodo/OSF (<https://osf.io>):

- Powerpoints, small data sets
- Zenodo identifier can be added to everything
- Preregistration in OSF

Publishing data sets:

- Open Neuro (<https://openneuro.org/>)
- Zenodo (<https://zenodo.org/>)
- GIN (<https://gin.g-node.org/explore/repos>)

Activity 6:

What are alternatives to the current publishing and peer review model?

How can peer review made more open and fair?

Questions and discussion?

The background features a light green rectangular area on the left and a pink rectangular area on the right, both positioned below the main text.