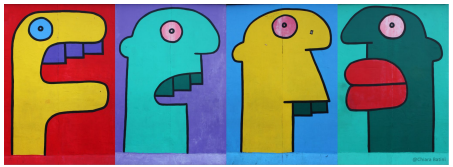
Reproducible science in action

Setting up git for practicals

Google scholar

Franco Marsico

Github: https://github.com/MarsicoFL



EMBO POPGEN

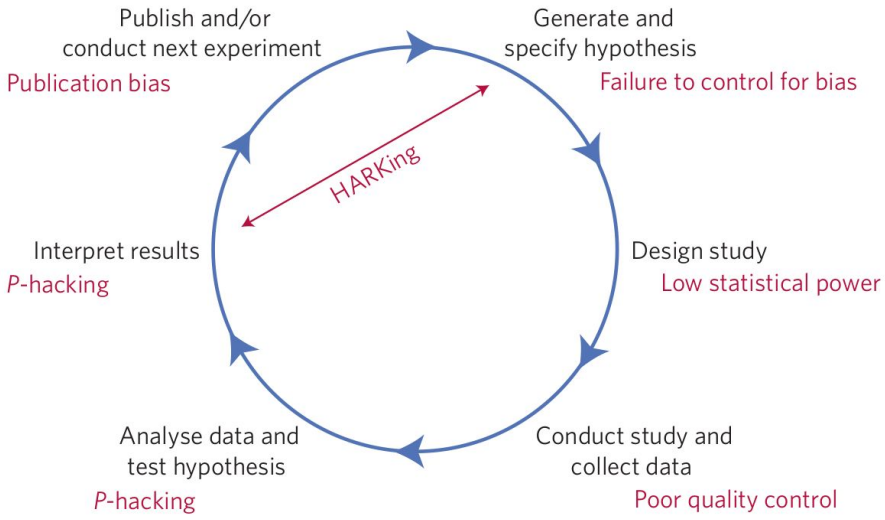
Overview

- What is reproducible science?
- Sharing the code
- A conceptual introduction to Git
- Using GitHub repositories
- Practical example: Setting up RStudio with GitHub
- Using git in the command line

Reproducible Science

Definition: Reproducible science refers to the ability to replicate the results of a scientific study using the same methods, data, and conditions. Its application is expected to:

- Ensures the credibility and reliability of scientific findings.
- Enhances transparency and trust in research.
- Facilitates the efficient accumulation and validation of knowledge.



Munafò, M. R., Nosek, B. A., Bishop, D. V., Button, K. S., Chambers, C. D., Percie du Sert, N., ... & Ioannidis, J. (2017). A manifesto for reproducible science. *Nature human behaviour*, *1*(1), 1-9.

What we can do?

Transparency and Open Science:

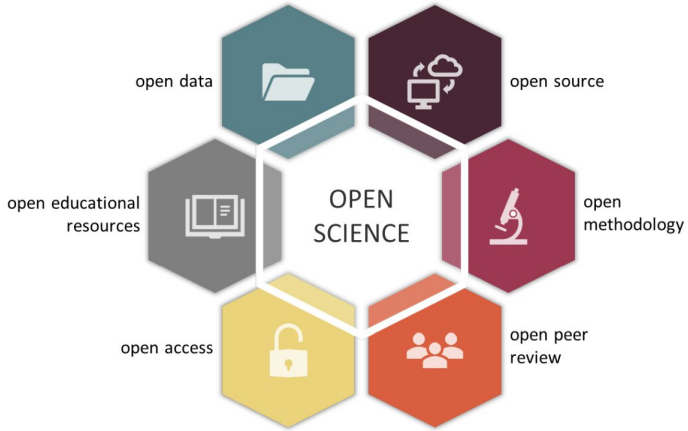
- Share data, materials, and code openly.
- Use open-access repositories and platforms like GitHub.

Blinding and Methodological Rigor:

- Implement blinding to reduce bias.
- Employ rigorous and standardized methods.

Collaboration and Team Science:

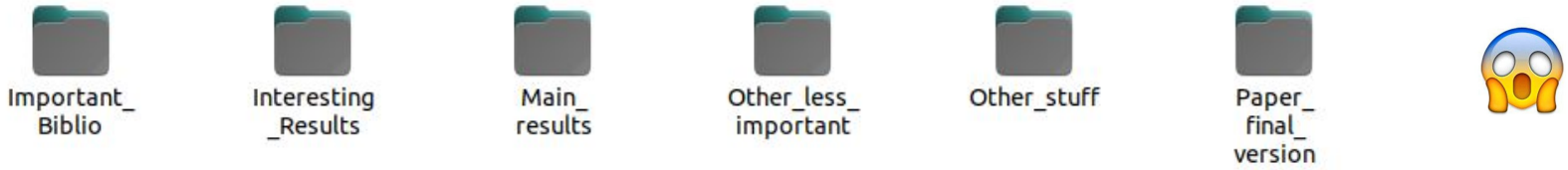
- Foster collaborative efforts and multi-site studies to enhance statistical power and generalizability.



Gallagher, R. V., Falster, D. S., Maitner, B. S., Salguero-Gómez, R., Vandvik, V., Pearse, W. D., ... & Enquist, B. J. (2020). Open Science principles for accelerating trait-based science across the Tree of Life. Nature ecology & evolution, 4(3), 294-303.

Sharing the code

It is not enough to share the data and the code



Interpretability and traceability become central

Good practices when sharing the code

Documentation:

- Provide clear and comprehensive documentation.
- Include a README file with instructions on how to use the code.

Licensing:

- Choose an appropriate open-source license (e.g., MIT, GPL).

Accessibility:

- Share your code on platforms like GitHub or GitLab.

Examples and Tests:

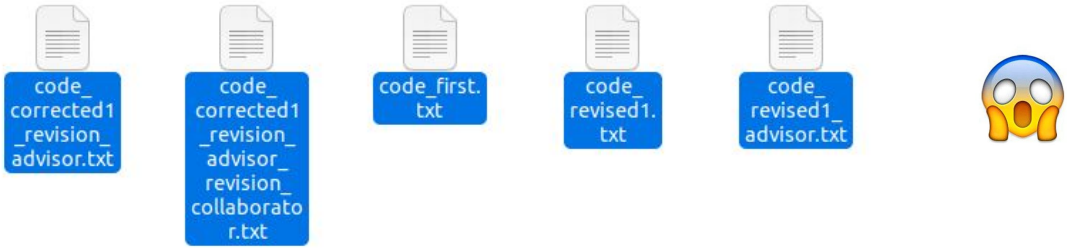
- Include example datasets and usage examples.
- Provide tests to ensure the code works as intended.

Version Control (very important for collaborations!):

- Use Git for version control to track changes and collaborate efficiently.
- Create a well-organized repository structure.

https://docs.github.com/en

An introduction to Git

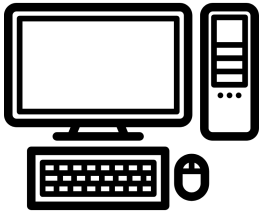


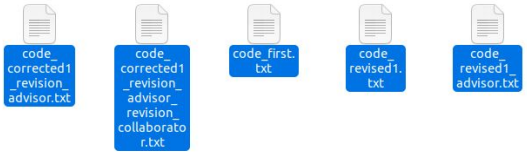
VCS: Version Control Systems are designed to organize collaborative software development

- They store all revisions of each file.
- They allow us to switch between versions and view differences.
- Each revision must be explicitly created with a message indicating the changes made.
- Distributed: Each user works independently on their own computer but can share their modifications with others.

Git: Key concepts

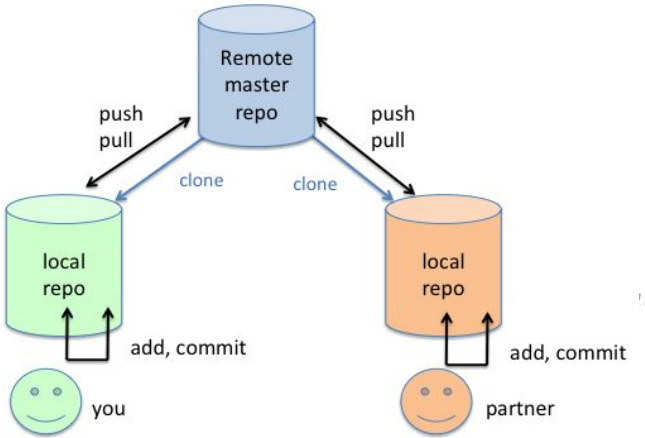
- **Local repository**: The place where the files of a project are stored, along with all the additional information needed for version control.
- **Revision**: A snapshot of the repository at a given moment.
- **History**: A set of revisions ordered chronologically.
- **Remote repository**: A server to which Git sends changes when a push is executed.





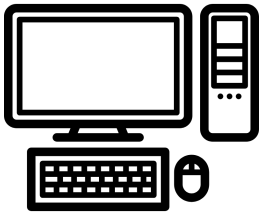
Git actions

- **add**: Adds a file to the revision.
- **commit**: Creates a new revision with a message describing changes.
- **clone**: Downloads a repository and its history.
- **push**: Shares local commits with a remote server or user.
- **pull**: Updates local repository with changes from others.
- **merge**: Combines local changes with others' changes.
- **status**: Shows the state of the local repository.

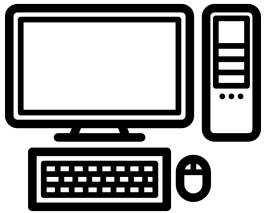


Git actions

Remote repository



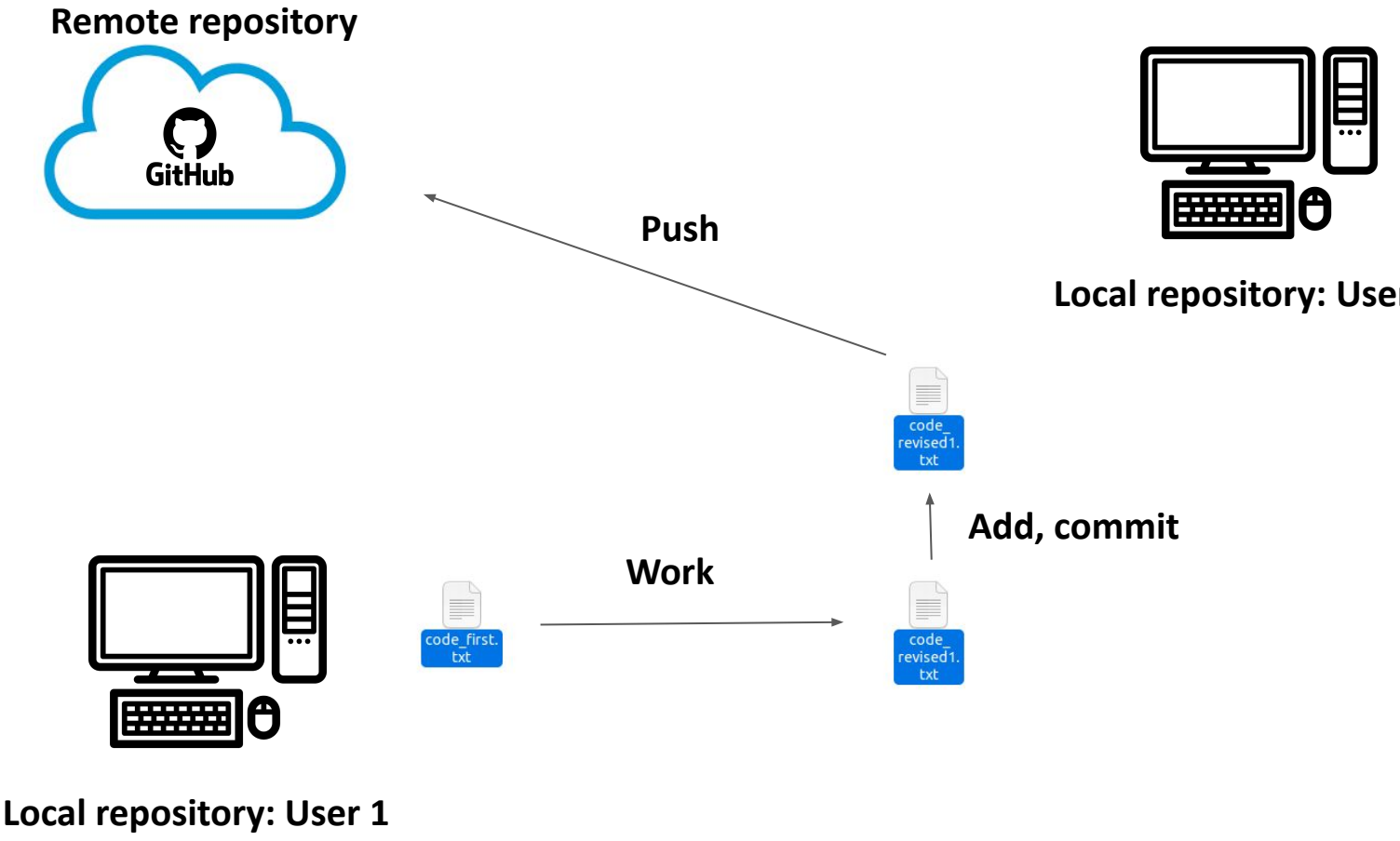
Local repository: User 2



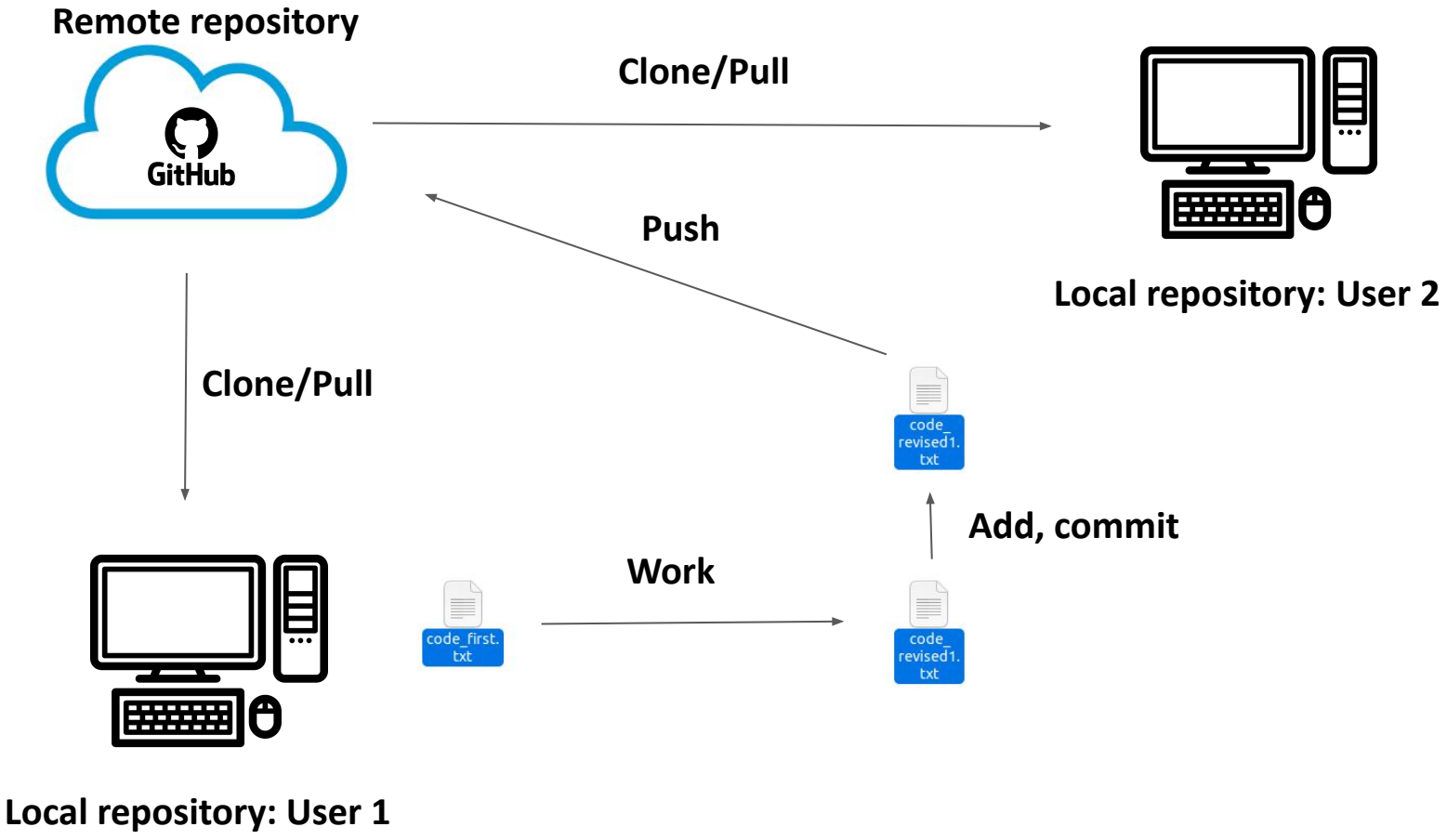
Work

Local repository: User 1

Git actions


Remote repository
GitHub
Push
Local repository: User 2
code_revised1.txt
Add, commit
Work
code_first.txt
code_revised1.txt
Local repository: User 1

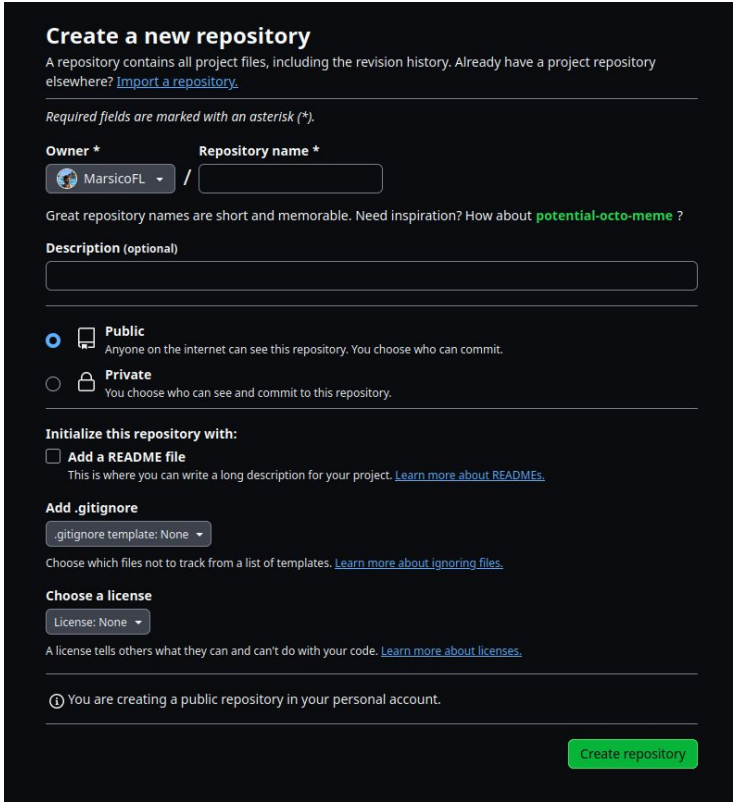
Git actions


Remote repository
GitHub
Clone/Pull
Local repository: User 2
Push
Clone/Pull
code_revised1.txt
Add, commit
Work
code_first.txt
code_revised1.txt
Local repository: User 1

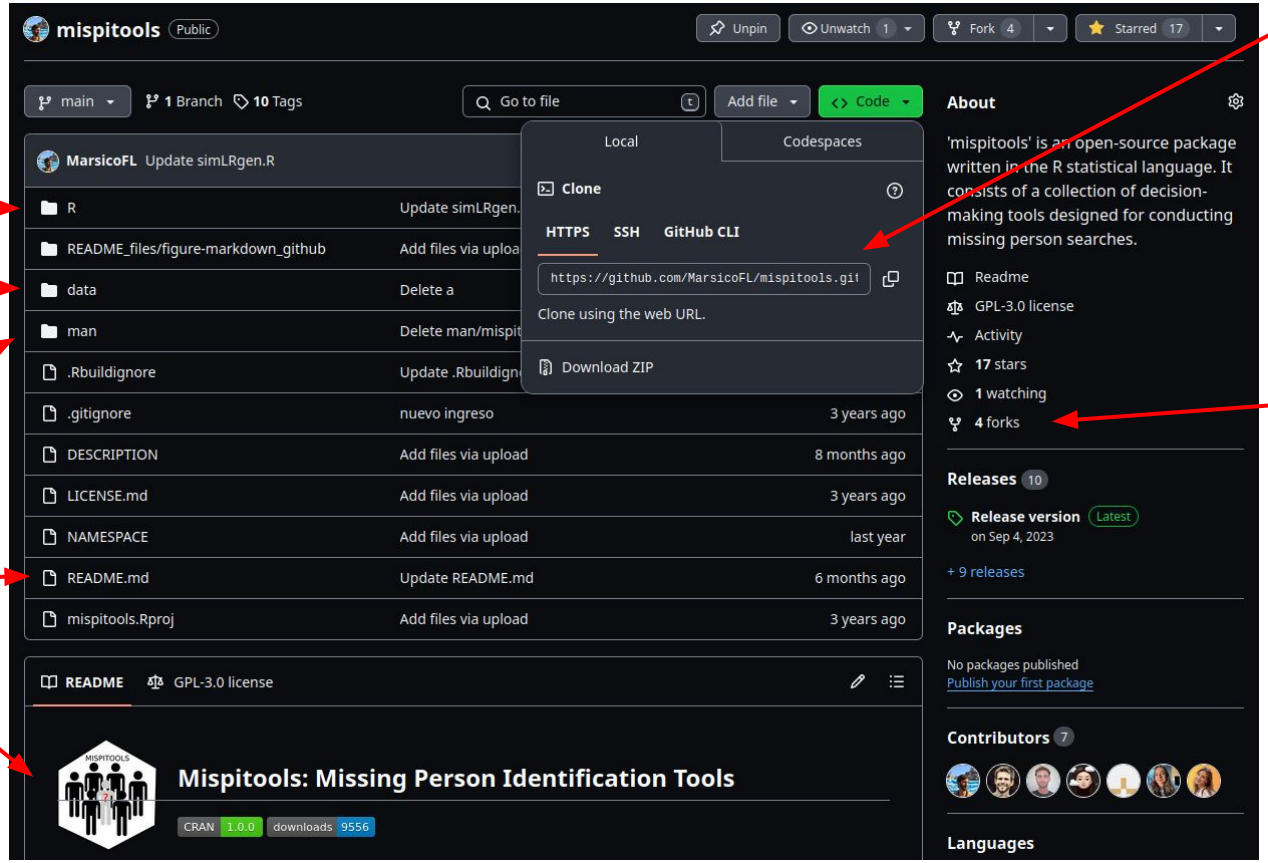
Using GitHub repositories

- Several services offer Git repository hosting:
  - Provide a convenient web interface for:
    - Creating repositories
    - Managing user access
    - Viewing files
    - Comparing revisions
    - Viewing history
- Well-known services include:
  - GitHub
  - GitLab
  - Bitbucket

Repositories > New



Using GitHub repositories



Code

Data

Code and data documentation

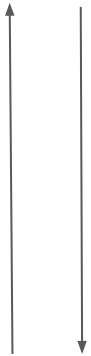
General documentation

Path

Copies

GitHub

Practical example: Setting up RStudio with GitHub



1 - Setting Up the key

Guide for Setting Up Git and GitHub with RStudio

EMBO POPGEN 2024

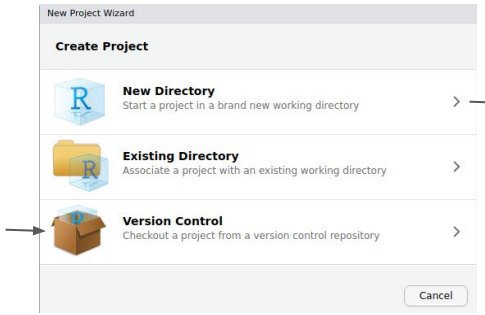
This guide provides a step-by-step process to set up R, RStudio, Git, and GitHub with RStudio. It includes instructions for installing R, RStudio, and Git, configuring Git with RStudio, creating a personal access token for GitHub, and verifying the setup.

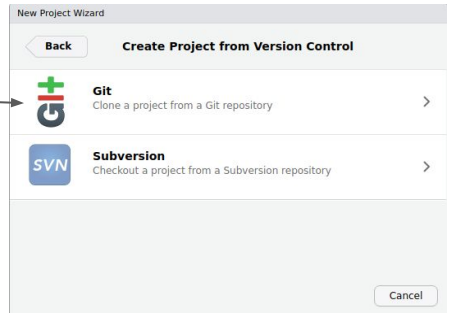
https://github.com/ColonnaLab/EMBO_popgen/blob/main/popgen2024/Franco_Marsico/Tutorial_RStudio_Github.pdf

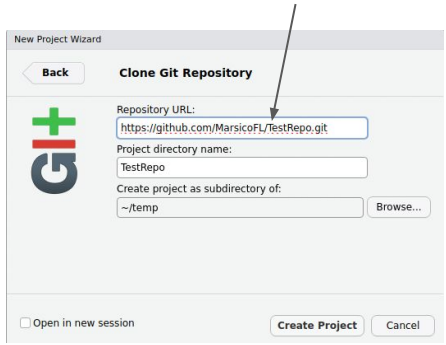
2 - Open a new git project

RStudio: File > New Project

Your github repo



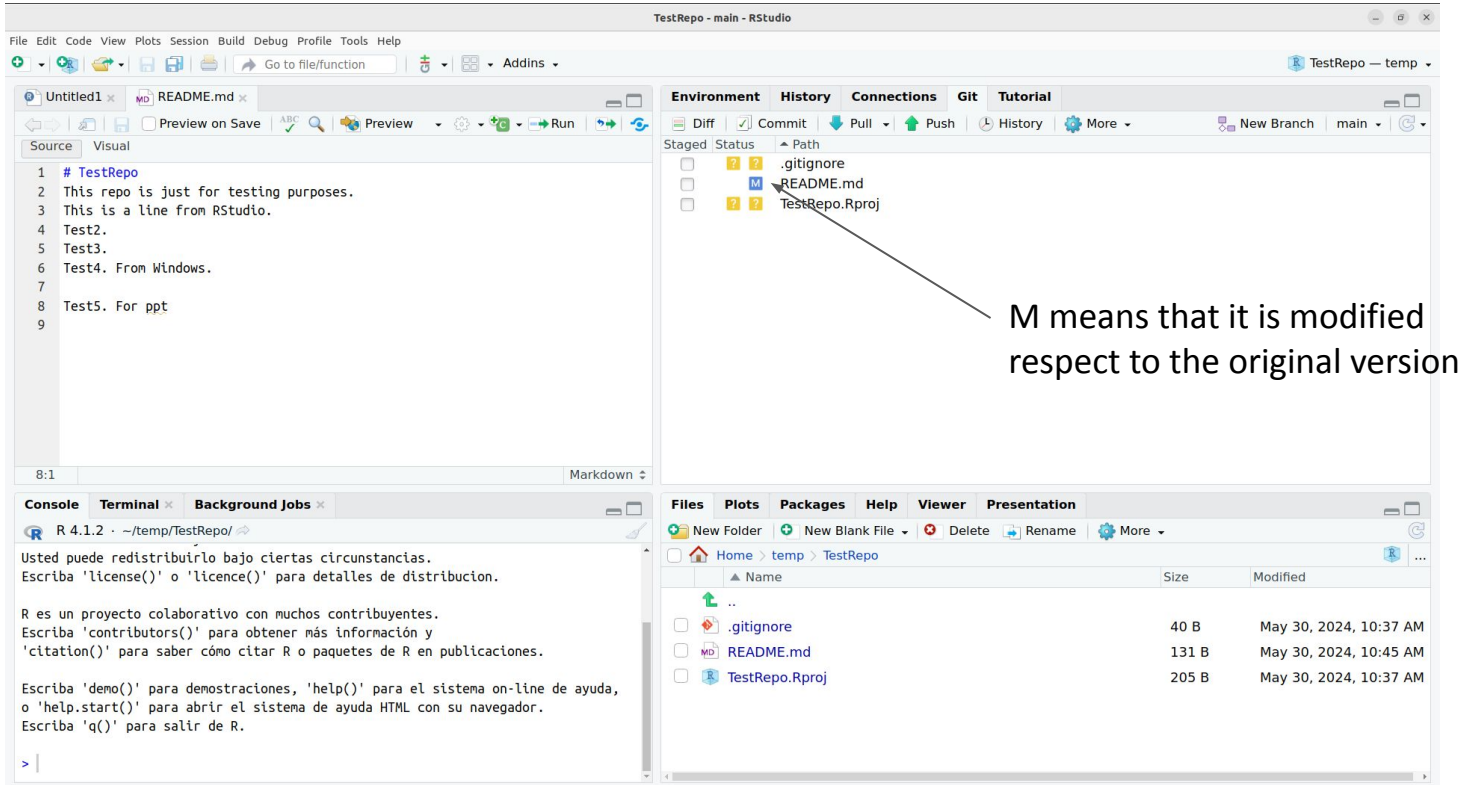




Practical example: Setting up RStudio with GitHub

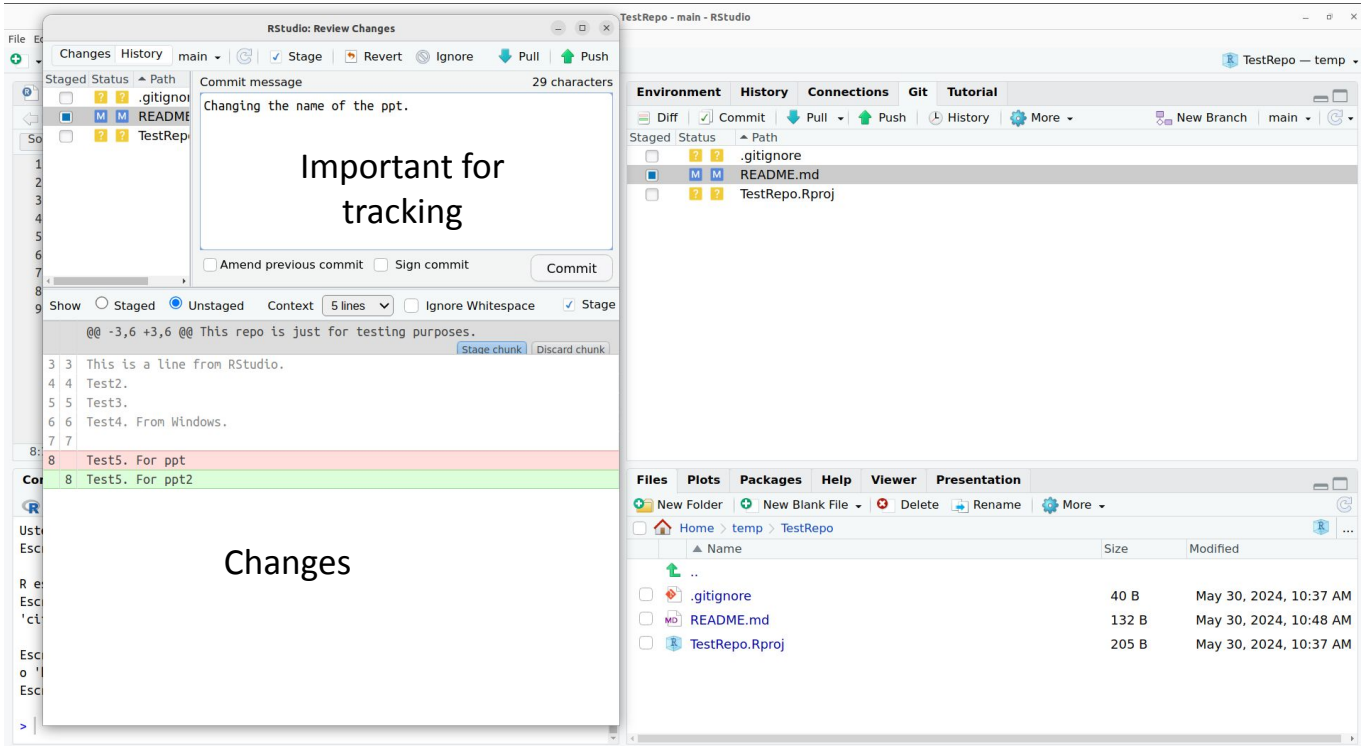
3 - Open and modify files

Panel showing git actions



Practical example: Setting up RStudio with GitHub

4 - Commit selecting modified files



After commit this message appears

Practical example: Setting up RStudio with GitHub

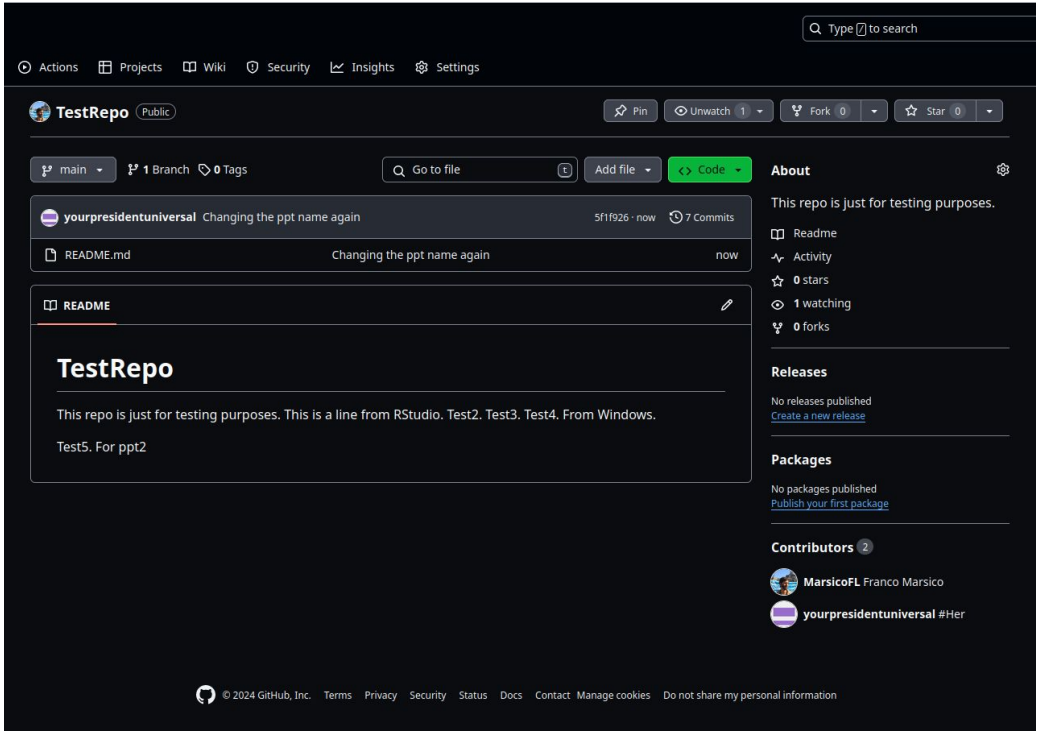
5 - Push

Git panel > Push

6 - Pull

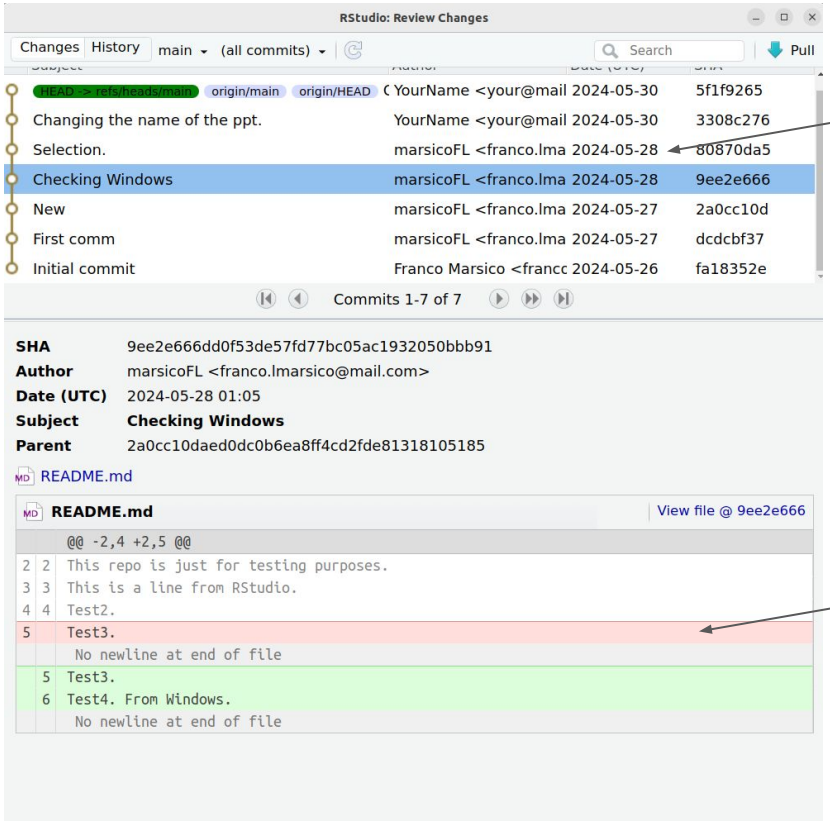
If the repository is a collaborative exercise is a good practice pull (actualize with changes from other users) after push.

Changes are incorporated in the remote repository



Practical example: Setting up RStudio with GitHub

7 - History



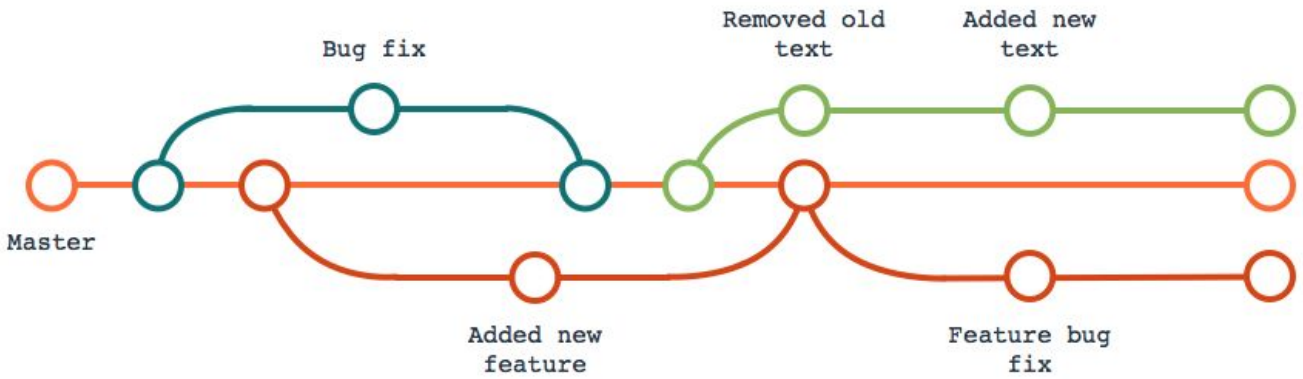
Commit message

User

Changes

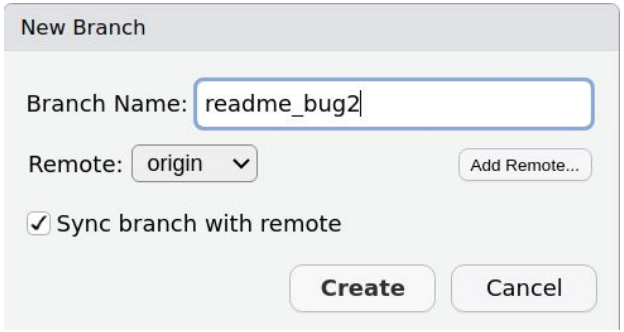
8 - Branching

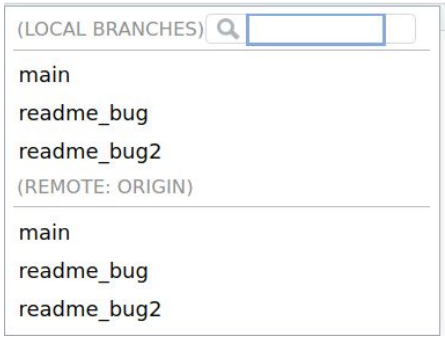
- **branch**: A fork of the repository's history.
- **merge**: The action of combining the history of two different branches.
- **master**: The main branch that always exists.
- **HEAD**: The most current revision of the branch you are on.
- **checkout**: The action to switch from one branch to another (also works with commits!).

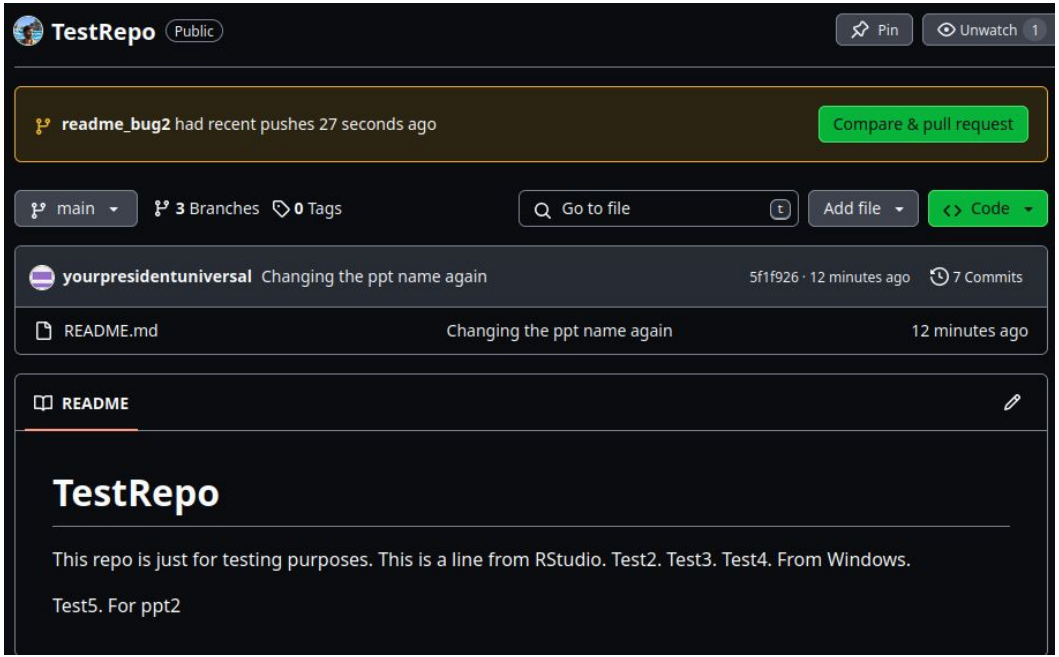


Practical example: Setting up RStudio with GitHub

Git panel > New Branch

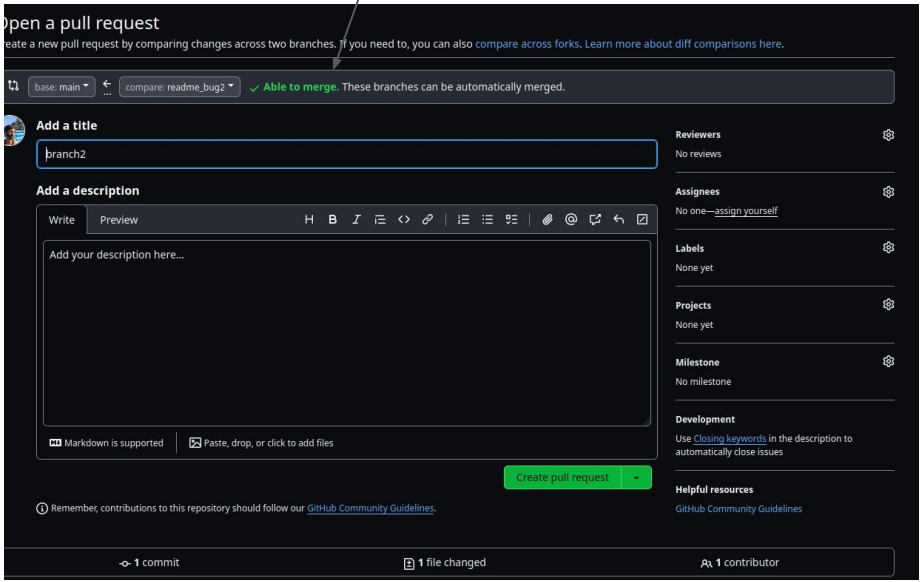


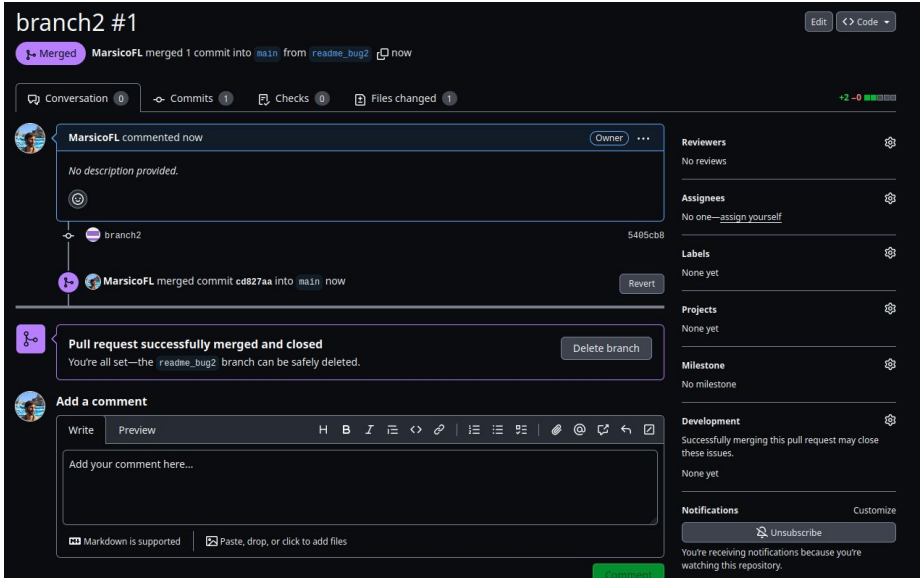




Practical example: Setting up RStudio with GitHub

Check for conflicts





Using git in the command line

Why use the git command line?

- The RStudio GUI is very useful for GitHub, but it may fail with other clients.
- It allows working with the terminal and not in interactive mode (useful when working in HPC).
- It provides access to a full range of commands and not only those supported by the GUI.
- Avoiding the GUI is more efficient.

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Conclusions

- We have introduced key aspects of reproducible science.
- Assessed the role of sharing code in reproducibility.
- Introduced central concepts of Git and GitHub.
- Set up RStudio GUI for connecting with GitHub.
- Discussed the benefits of Git command line functions.