GitHub Lab Procedures

A repository management start kit for new lab members

Florencia D'Andrea

2024-11-29

Table of contents

Goal

The purpose of this manual is to summarize the steps necessary for using GitHub as a Lab Member.

Why Use GitHub?

Git and GitHub were originally created for professional software development. However, their use has extended into the scientific field for various reasons:

- **Backup**
  Making it a habit to push to GitHub at least once a day allows you to keep an online copy of your data analysis.
- **Improved Documentation Practices**
  Having a complete README and knowing that your colleagues have access to your code encourages better organization of the project, making it clearer, more concise, and well-documented.
- **Version Control**
  It is easy to see how the project has evolved over time and recover changes from previous versions.
- **Reproducibility**
  When publishing scientific articles, maintaining reproducible results is considered a quality practice.

It's important to clarify that lab members are not expected to be expert users of Git and GitHub, but rather to handle basic commands necessary to achieve the use proposed.

Objectives of this Manual

- Provide a quick onboarding guide and basic data management practices for new lab members, including data analysis backup and how to maintain the privacy of datasets by having clear steps to prevent pushing data.

- Set guidelines on the use and management of code within the lab to facilitate sharing unpublished or complementary data analysis among current and former lab members.
- Establish basic rules to publish public repositories associated with scientific articles, ensuring transparency in analyses and reproducibility of results.
- Ensure consistency in managing data analyses over time and preserve and centralize the knowledge generated in the different projects.

References

1 Design for Technology Use in the Lab

Most of the databases used in our lab are private and do not contain enough data to be considered big data. These databases are used locally.

- It is suggested to use a `raw/` folder and a `processed/` folder to separate unprocessed data from processed data.
- To ensure that the data is not made public, it is recommended to use a `.gitignore` file and avoid pushing the folder where the data is stored to the repository.

2 Organization Owner on GitHub

2.1 GitHub Organization Management

- **Adding new members**: Ensure that new lab members join the GitHub Organization once they have created their GitHub account.

3 GitHub Team Maintainer

At least one person in the lab will be assigned as the Team Maintainer of the GitHub organization. This role will allow them to add and remove members from the team, granting or revoking access to private repositories.

For this, it's important to assign the Lab Member in charge the **Team Maintainer** role for the Lab Team.
Follow the instructions in Assigning the team maintainer role to a team member

If properly assigned the label `mantainer` should appear next to they name in this repository: https://github.com/orgs/StringhiniLab/teams/lab-team.

3.1 Responsibilities of the Maintainer

- Add new lab members to the Lab Team.
- Remove team members who are no longer part of the lab.
- Assign other members as Team Maintainers if necessary.

3.2 References

4 Onboarding

Checklist

- ☐ Create your personal GitHub account.
- ☐ Request access to the GitHub Organization.
- ☐ Request access to the Lab Team.
- ☐ Install Git, R and RStudio.

4.1 Create Your Personal GitHub Account

First, you need to create an account on GitHub by following these steps: Creating an account on GitHub.

After creating your GitHub account, you'll notice that your profile is associated with a specific URL, structured as follows:

- `https://github.com/<username>`

This page allows you to access your account settings and all repositories you create.

4.2 Request Access to the GitHub Organization

Our lab has a GitHub organization that centralizes the repositories for everything produced in the lab.
Notice that the organization's URL is different from your profile's:

- https://github.com/StringhiniLab

To gain access, you need to provide your GitHub username to the person managing the organization. Having access to the organization will enable you to create repositories within it.

4.3 Request Access to the Lab Team

Not all lab repositories are public. To access repositories owned by other lab members, you must also be added to the Lab Team.
Dr. Stringhini, as the owner of the organization, always has access to all repositories.

- Adding organization members to a team
- Managing team access to an organization repository

4.4 Installation Instructions

To work in the lab, you will need to have Git, R, and RStudio installed.
Please note that the installation process may vary depending on whether you have a computer with Windows, Linux, Mac (Intel), or Mac (Apple Silicon) operating systems.

- **Git**: Download Git

- **R**: Download R

- **RStudio**: Download RStudio

Once you've completed these steps, you can move on to the next section.

4.5 References

- Best practices for organizations

5 Starting a New Project

These steps only need to be completed once, at the beginning of a project.

Checklist

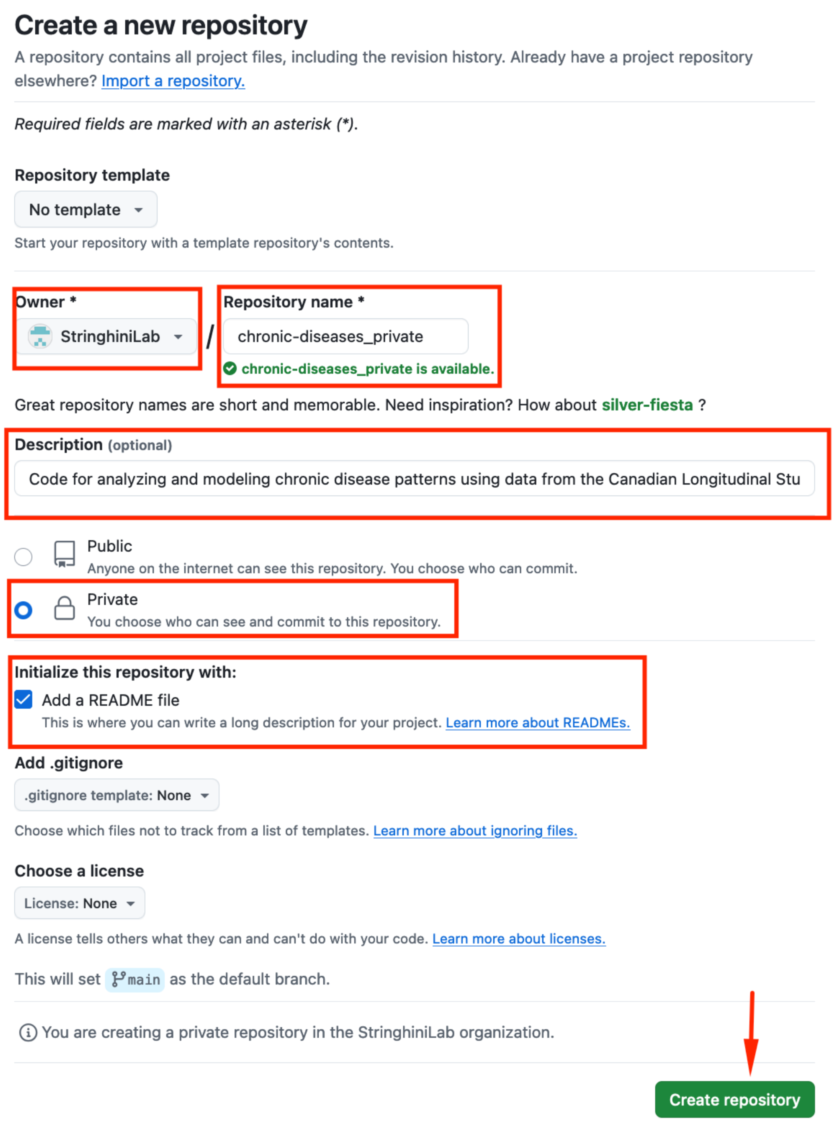
- ☐ Create a private repository by new project.

- ☐ Add the Lab Team to the repository.
- ☐ Clone the repository and associate it with an RStudio Project.
- ☐ Add your Data folder to `.gitignore`.
- ☐ Complete the `README` file using the template.

5.1 Create a private repository by new project

When starting to work on a new project, your first step is to create a **private** repository in the lab's GitHub organization: StringhiniLab GitHub.

1. Click the green **New** button to open a window like this:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner *

StringhiniLab

Repository name *

chronic-diseases_private

chronic-diseases_private is available.

Great repository names are short and memorable. Need inspiration? How about silver-fiesta ?

Description (optional)

Code for analyzing and modeling chronic disease patterns using data from the Canadian Longitudinal Stu

Public
Anyone on the internet can see this repository. You choose who can commit.

[✓ Private]
You choose who can see and commit to this repository.

Initialize this repository with:

[✓] Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch.

You are creating a private repository in the StringhiniLab organization.

2. Complete/select using the following criteria:

☐ **Owner**
Select `StringhiniLab` as the owner, not your personal GitHub account.

☐ **Repository Name**
Choose a name that represents your project. Since this repository will be private, append `_private` to the name.
For example, if the repository name is `chronic-diseases`, name it `chronic-diseases_private`.

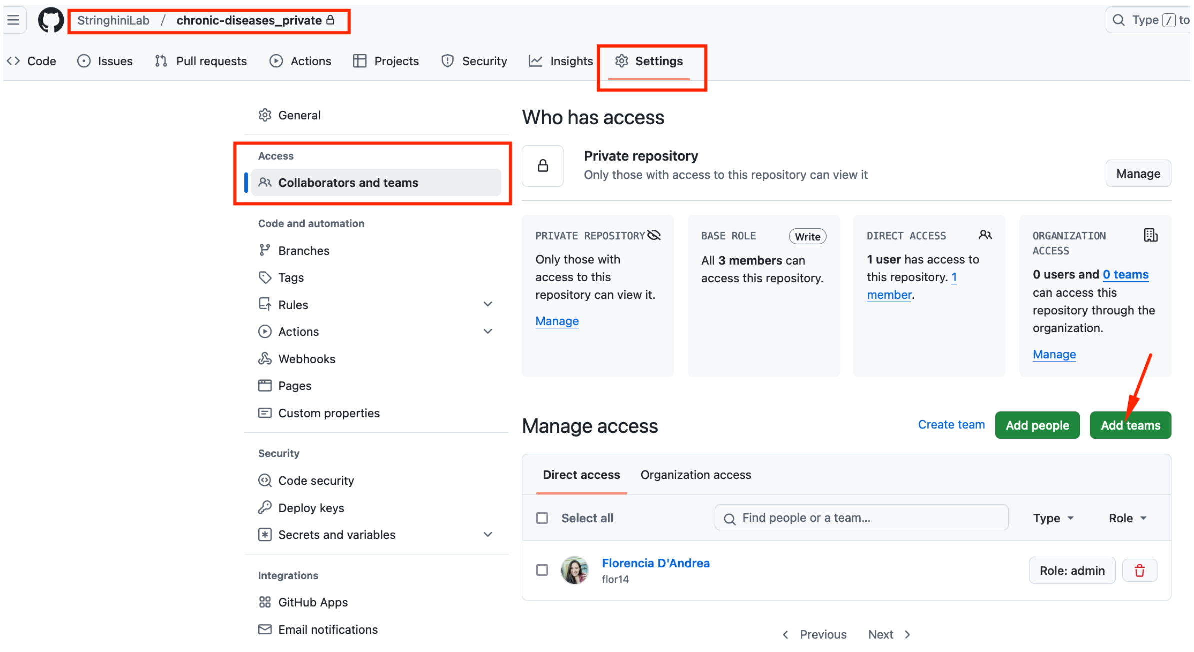
☐ **Description**

Provide a more detailed description of the project here. This helps identify the repository's content in the organization.

- ☐ **Public or Private?**
  Ensure the repository is set to `Private`.
- ☐ **Initialize Repository With**
  Add a `README.md`.

Ignore the other options for now.
If everything works, you'll see your repository within the GitHub organization labeled as `Private`.

5.2 Add the Lab Team to the repository



Private repository
Only those with access to this repository can view it

Only those with access to this repository can view it.

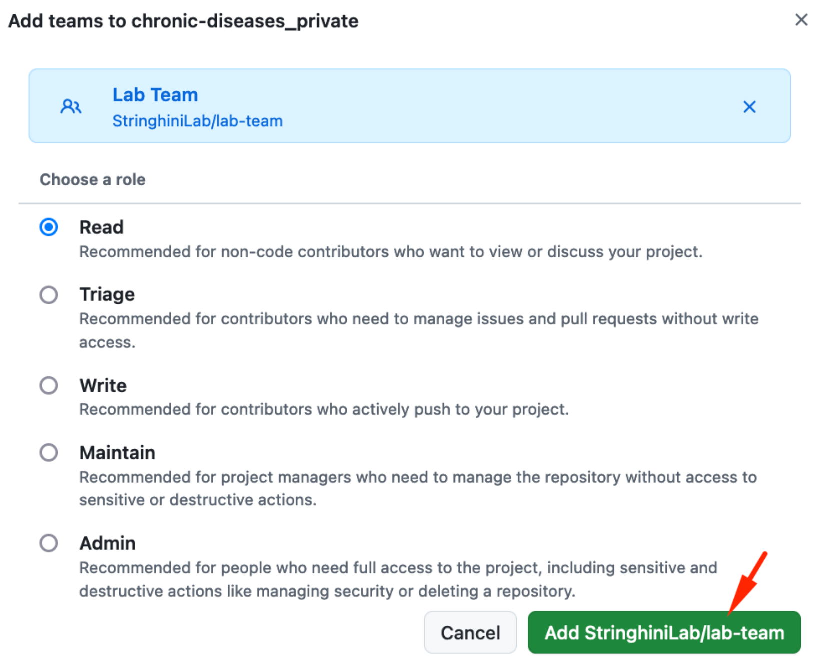
All 3 members can access this repository.

1 user has access to this repository. 1 member.

0 users and 0 teams can access this repository through the organization.

Florencia D'Andrea
flor14
Role: admin

1. **Navigate to the repository's `Settings` tab.**
   In the left-hand sidebar, find and click on `Collaborators and teams`.
2. Click **Add teams** and add `Lab Team`.
   By default, you will select the Read role for the team. The idea is that other Lab Members can view the repository but will not be able to edit it by mistake.


Add teams to chronic-diseases_private

Lab Team
StringhiniLab/lab-team

Choose a role

[✓ Read]
Recommended for non-code contributors who want to view or discuss your project.

Triage
Recommended for contributors who need to manage issues and pull requests without write access.

Write
Recommended for contributors who actively push to your project.

Maintain
Recommended for project managers who need to manage the repository without access to sensitive or destructive actions.

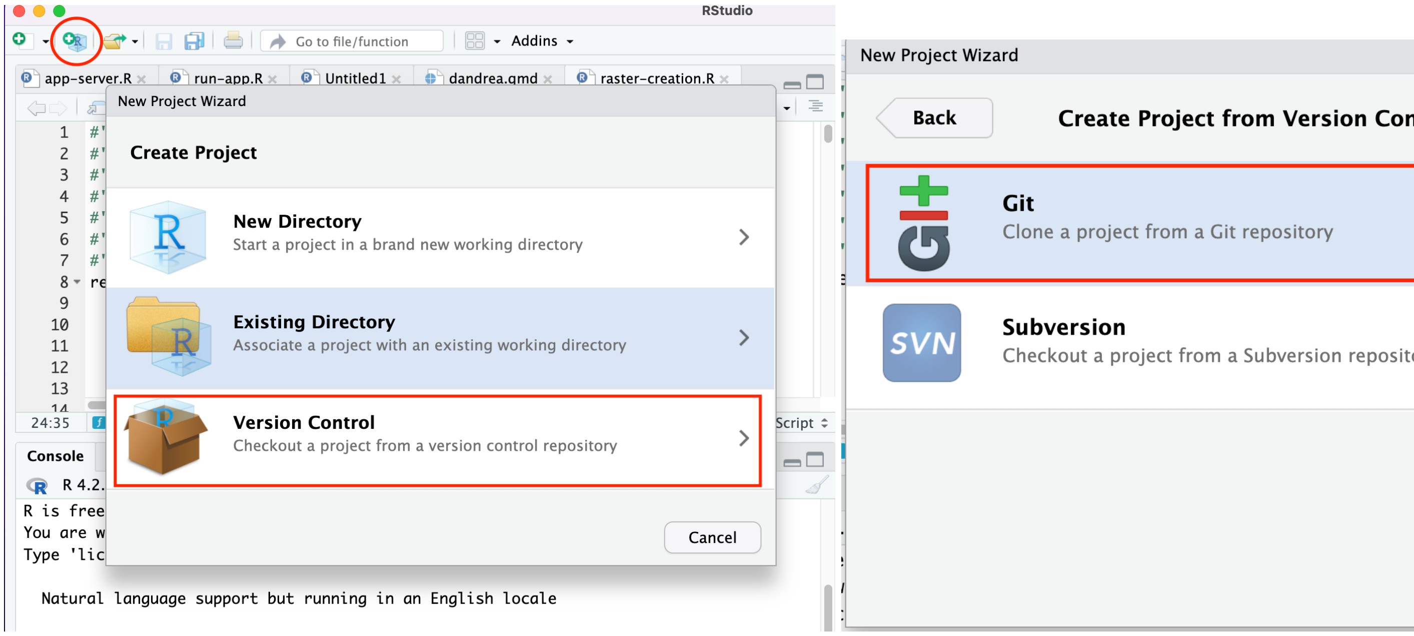
Admin
Recommended for people who need full access to the project, including sensitive and destructive actions like managing security or deleting a repository.

This allows all current Lab Members to view (but not modify) your project.
If you don't want to share an analysis with other Lab Members, you can create a repository in your personal GitHub account instead. However, **always ensure sensitive data is not pushed to GitHub** for confidentiality reasons.

All repositories in `StringhiniLab` should be accessible to the Lab Team, which is why these repositories are hosted in the organization instead of personal accounts. Remember that individuals who are Owners of the organization can view all repositories even if there are not part of the Team.

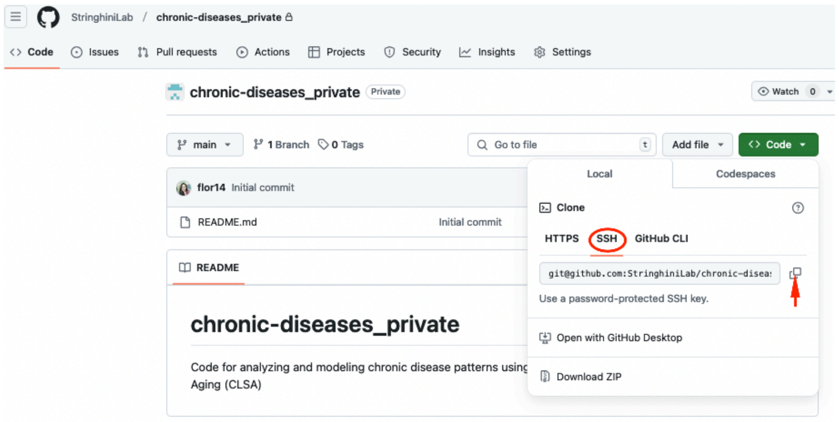
5.3 Clone the repository and associate it with an RStudio Project

1. **Open RStudio**
   If RStudio is not installed, complete first the installation instructions in the Onboarding section.
2. **Clone the Repository**
   In RStudio:
   `File > New Project > Version Control > Clone a Project from a Git Repository`.


RStudio
Go to file/function
Addins
app-server.R
run-app.R
Untitled1
dandrea.qmd
raster-creation.R
New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel
24:35
Script
Console
R 4.2.
New Project Wizard
Back
Create Project from Version Con
Git
Clone a project from a Git repository
Subversion
Checkout a project from a Subversion reposit

```
R is free
You are w
Type 'lic

 Natural language support but running in an English locale
```

Go back to the repository and copy the repository's URL.



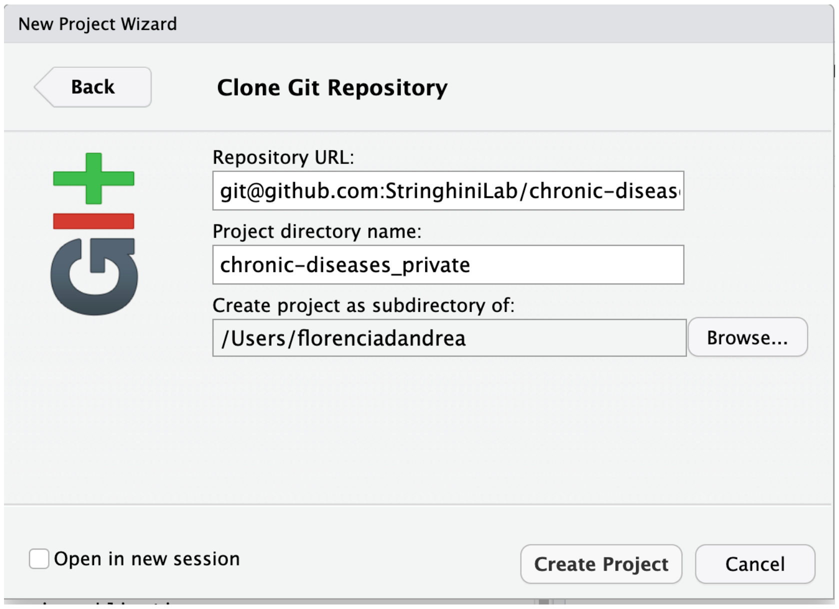
```
git@github.com:StringhiniLab/chronic-diseas
```

Use a password-protected SSH key.

chronic-diseases_private

Code for analyzing and modeling chronic disease patterns using
Aging (CLSA)

And paste it in the correct field:



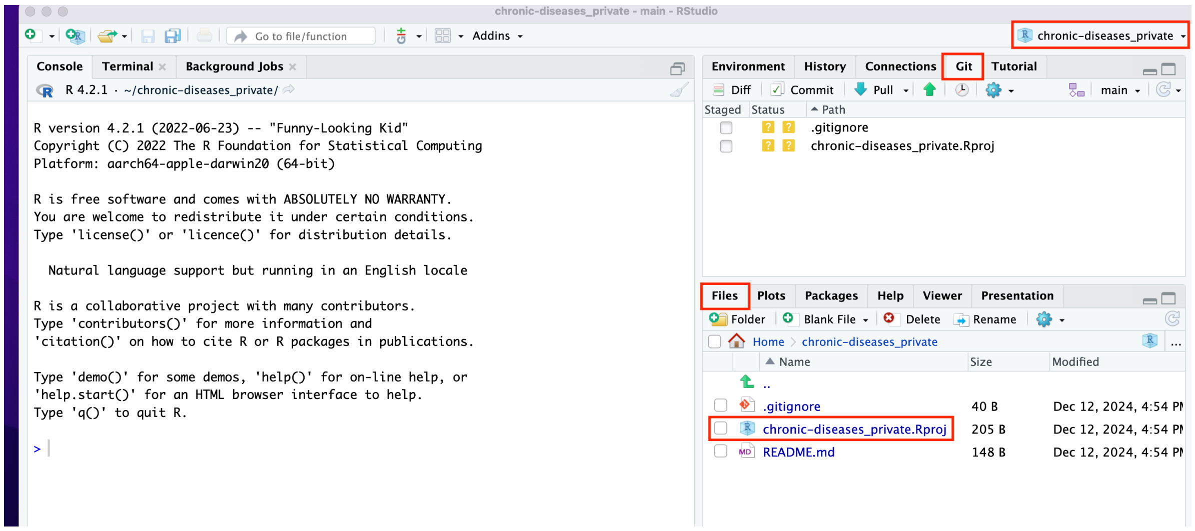
Repository URL: git@github.com:StringhiniLab/chronic-diseas
Project directory name: chronic-diseases_private
Create project as subdirectory of: /Users/florenciadandrea

If successful, you'll see a folder containing your project, including the `README` file that we created on GitHub, in the `Files` tab at the bottom-right of RStudio.

Note that an `.Rproj` icon has appeared. Clicking on this icon outside of RStudio will open RStudio directly within the project.

Additionally, the project name now appears in the top-right corner. By opening that menu, you can easily switch between projects without leaving the RStudio IDE.

You'll also notice a tab named Git in the top-right panel.



```
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Staged	Status	Path
☐	? ?	.gitignore
☐	? ?	chronic-diseases_private.Rproj

Name	Size	Modified
..		
.gitignore	40 B	Dec 12, 2024, 4:54 PM
chronic-diseases_private.Rproj	205 B	Dec 12, 2024, 4:54 PM
README.md	148 B	Dec 12, 2024, 4:54 PM

5.3.1 Add your Data folder to `.gitignore`

We work with sensitive data. If working locally, create a `data` folder (e.g., click the `+ Folder` icon in the `Files` tab). Move your data files into this folder.

Open the `.gitignore` file and add the line `data/`. This tells Git to ignore the contents of the `data` folder, preventing accidental data pushes.

We recommend creating at least two sub-folders within `data/`:

- `raw/`: Use this folder to store the original datasets.
- `processed/`: Use this folder to save any datasets generated as preliminary or final results from your analyses.

If `data/` is listed in your `.gitignore` file, both subfolders will automatically be ignored by Git since they are located within the `data/` folder.

If the folder is not in the project root or has a different name, adjust the `.gitignore` settings accordingly.

Your project structure should look like this:

```
project-folder/
  .gitignore # Specifies files and folders to ignore in version control
  README.md # Documentation about the project
  data/ # Folder to store datasets
      raw/ # Original datasets (never modified directly)
      processed/ # Cleaned and processed datasets
```

and you `.gitignore` file should look like this:

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
/.quarto/
data/
```

5.4 Complete the README using the template

Before starting work, fill out the `README.md` file with the following information:

```
# Title

## Author
**Name:** [Your Name]
**Email:** [Your Email]

## Start Date
**Initial Commit:** [YYYY-MM-DD]

## Objective
The objective of this project is to analyze and model chronic disease patterns using data fr

## Database Used and Version
**Database Name:** Canadian Longitudinal Study of Aging (CLSA)
**Data Version:** [Specify version or date accessed]

-[] Sensitive data from the CLSA database is stored locally and excluded from version contro
-[] All analyses comply with CLSA data use agreements.

## Project Structure

chronic-diseases/
  data/ # Folder for datasets
    raw/ # Original datasets (never modified directly)
    processed/ # Cleaned and processed datasets
  scripts/ # R scripts for analysis
  outputs/ # Figures, tables, and other results
  README.md # Project overview and documentation
  .Rproj # RStudio project file


## Reproducibility

sessionInfo()
```

Now you're ready to start writing code!

5.5 References

6 Regular Project Workflow

These steps should be completed every day you work on the project. Although they may seem complex at first, once you get accustomed to them, you won't need to think about it anymore.

Checklist

- ☐ Add all the files you want to commit to the staging area.
- ☐ Create a commit message.
- ☐ Push the changes to the GitHub repository.

6.1 How to use the Git tab in RStudio

Pay attention to the Git tab located in the top-right corner. Git will only display files that have been added, modified, or deleted since the project was initialized or since the last commit (we'll cover what that means shortly).

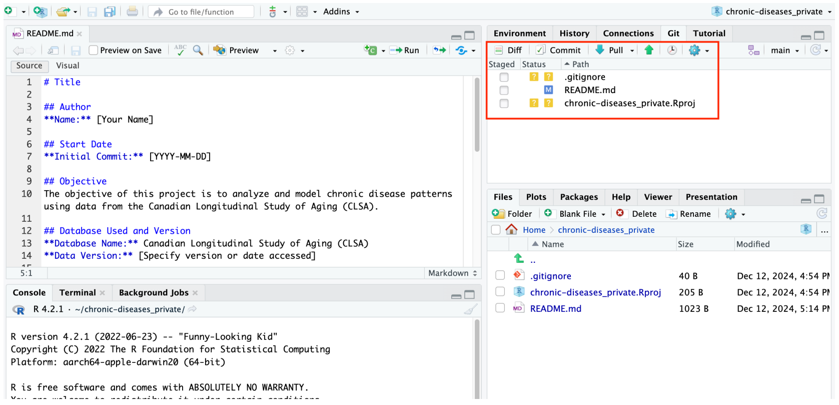
Keep in mind that when we cloned the project from GitHub, two new files were added:

- The `.Rproj` file, created because we based our RStudio project on the repository.
- The `.gitignore` file, automatically generated as part of the project setup in RStudio.

These files will appear with a yellow question mark, indicating they are *untracked*—in other words, Git is aware of them but has not yet saved them under version control.

The `README.md` file initially did not appear in the Git tab. However, after adding the template and saving the changes, it now shows a blue "M," which indicates that the file has been *modified*.

If you were to remove a file, you would see it next to a red 'D', indicating that it has been *deleted* from the project.



```
# Title

## Author
**Name:** [Your Name]

## Start Date
**Initial Commit:** [YYYY-MM-DD]

## Objective
The objective of this project is to analyze and model chronic disease patterns
using data from the Canadian Longitudinal Study of Aging (CLSA).

## Database Used and Version
**Database Name:** Canadian Longitudinal Study of Aging (CLSA)
**Data Version:** [Specify version or date accessed]
```

```
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
```

Staged	Status	Path
☐	? ?	.gitignore
☐	M	README.md
☐	? ?	chronic-diseases_private.Rproj

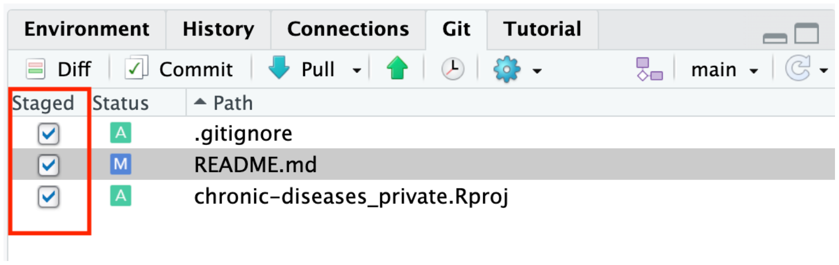
Name	Size	Modified
..		
.gitignore	40 B	Dec 12, 2024, 4:54 PM
chronic-diseases_private.Rproj	205 B	Dec 12, 2024, 4:54 PM
README.md	1023 B	Dec 12, 2024, 5:14 PM

The next step is to save these changes in the project and add a descriptive title. Each time you save a new version of the project, we say you are making a commit, which you label with a title.

6.1.1 Add all the files you want to commit to the staging area.

The first thing you need to do is check in the Staged section of the Git tab all the changes you want to save under the same title. You will notice that sometimes a green 'A' appears. You can ignore this. The important part is that you check all the changes you want to save.

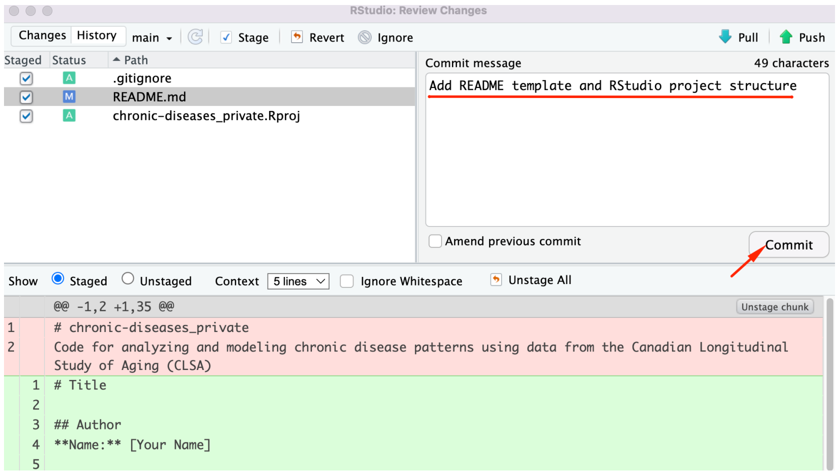
In our case, since we are working with non-collaborative repositories and the main purpose of using GitHub is to share data with other coworkers and maintain a backup, there's no need to focus too much on the details in this section.



Staged	Status	Path
✓	A	.gitignore
✓	M	README.md
✓	A	chronic-diseases_private.Rproj

6.1.2 Create a commit message.

After doing this, you need to click the commit button to make these changes permanent in the project. You will then choose a message for the commit and click the `Commit` button.



.gitignore
README.md
chronic-diseases_private.Rproj

Add README template and RStudio project structure

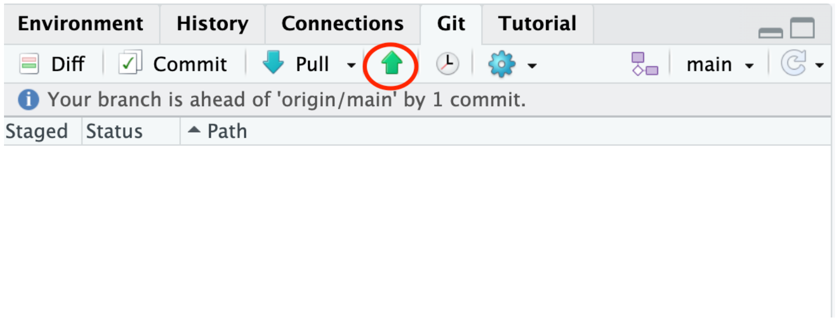
```
@@ -1,2 +1,35 @@
# chronic-diseases_private
Code for analyzing and modeling chronic disease patterns using data from the Canadian Longitudinal
Study of Aging (CLSA)
# Title

## Author
**Name:** [Your Name]

```

How Often Should You Commit?

Think of commits as checkpoints for related changes. If you might want to revert a set of changes later, commit them together.

6.2 Push the changes to the GitHub repository.

Finally, to push the changes to GitHub, click the `Push` button.



Your branch is ahead of 'origin/main' by 1 commit.

You'll notice that the files in the Git tab disappear after you commit. This is expected, as Git only tracks changes between commits. Remember, if you don't click the green arrow representing push, the changes will not take effect on GitHub.

If everything went smoothly, you should navigate to the repository URL and see the changes you made.

How Often Should You Push?

Push your changes **at least once a day** after completing your work.

6.3 References

- Article: Research Data Management
- UBC Research Data Management Guide

7 Offboarding

The Lab Member must be removed from the GitHub Lab Team.