

Version control with Git exercise

In this exercise we will show how version control with Git can be implemented for an example R project. The project is the same that is created in the first workflow [exercise](#), however to save time or in case you haven't completed this exercise we will start with the finished output from it.

Step 1: Download and configure Git

- Download Git from <https://git-scm.com/downloads>
- Once downloaded open the Git terminal window and type in the following with your credentials

```
git config --global user.name "NVHarisena1" git config --global user.email  
"NVHarisena1@ethz.ch" git config --global --list
```

The third command should return your updated user-name and email id.

Step 2: Create a repository on Github

We will make a quick repository on Github for an individual project, without changing much of the specific configurations since it will be beyond the scope of this workshop.

- Login to your account at <https://github.com/>
- Create a new repository by clicking the '+' sign in the top right side of the website or in the 'Start new repository' section in the homepage
- Provide a clear name for the repository for e.g. "R_repro_nv" and a quick description like "Test for Reproducible research workshop"
- Set the visibility of the profile to "Public"
- Initialize this repository with: Add a README file.
- Select a license for your repository in the "Choose your license" section. Check out [this website](#) to identify which license works for you. Even though it is optional to add license information to a repository, it is good practice to include this ([See more details](#)).
- Click the green 'create repository' button

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

Start your repository with a template repository's contents.

Owner * / Repository name * **1.**
 ✓ R_repro_nv is available.

Great repository names are short and memorable. Need inspiration? How about [literate-goggles](#) ?

Description (optional)
 2.

☒ 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

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
 5.

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

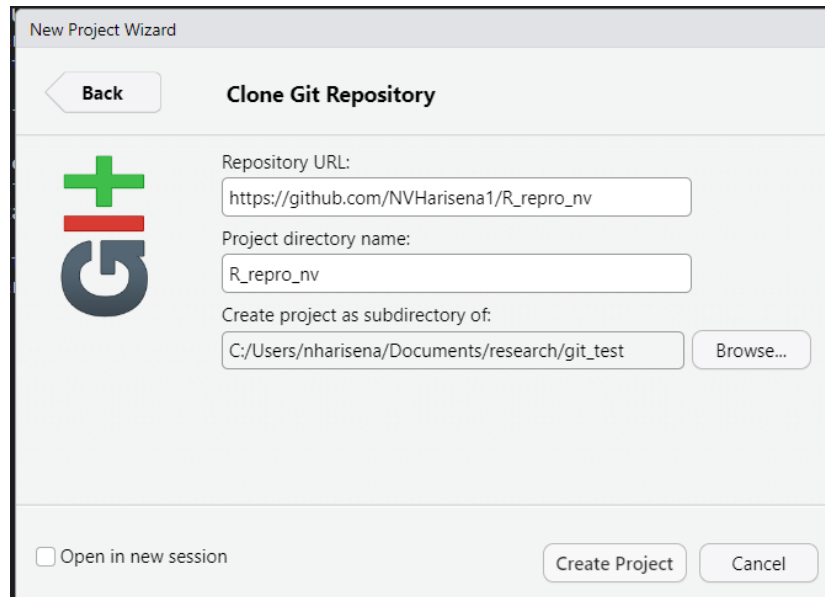
This will set `main` as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

1: Setting up the repository

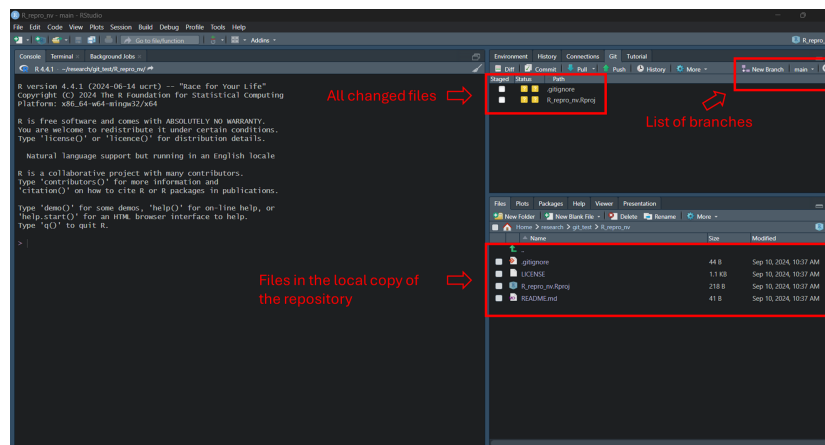
Step 3: Link created repository to R-Studio

- Open a new session in R studio and create a new project
- In the ‘New Project Wizard’ navigate to ‘Version Control’>‘Git’
- In the “repository URL” paste the URL of your new GitHub repository. It will be something like this https://github.com/nvharisena1/R_repro_test.
- Add folder location where you want the project to be saved locally in your computer in “Create project as subdirectory of” section
- Click ‘Create project’



2: Linking repository to R-Studio

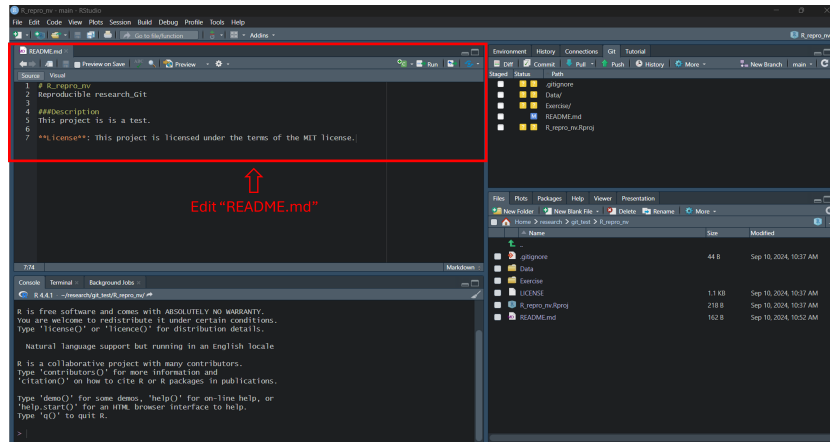
You will see R-studio has now been set up for your local clone of the project to communicate with your online repository. You can see a drop-down to the right of 'New Branch' button in the Git tab. This will show you all the branches available to pull or push data to. Your drop-down should show only a 'main' branch, since no new branches were created. As stated in the workflow introduction, creating branches is useful for projects with multiple collaborators or sub-themes. Pushing to different branches and then setting up a 'pull-request' to merge to the 'main' branch allows for systematic version control of the project.



3: R-Studio new project session with git link

Step 5: Edit the README.md file

- Open the README.md from the file viewer pane
- Add a description section for the project with a heading and a describing sentence, for e.g. “This project is is a test”.
- Add license information for the project, for e.g. “This project is licensed under the terms of the MIT license.”
- Save the file.

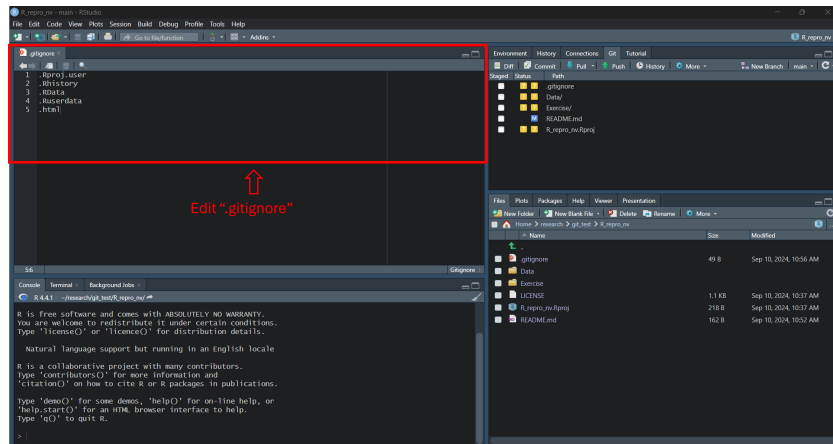


4: Edit README.md

Step 6: Edit the .gitignore file

The git ignore functionality tells git which files to ignore while ‘pushing’ the local changes to the remote (online) repository [see more details](#). In this example we will tell git to ignore all .html files. .html files are created when you preview a file, for example click preview on the edited README.md and a .html should be created.

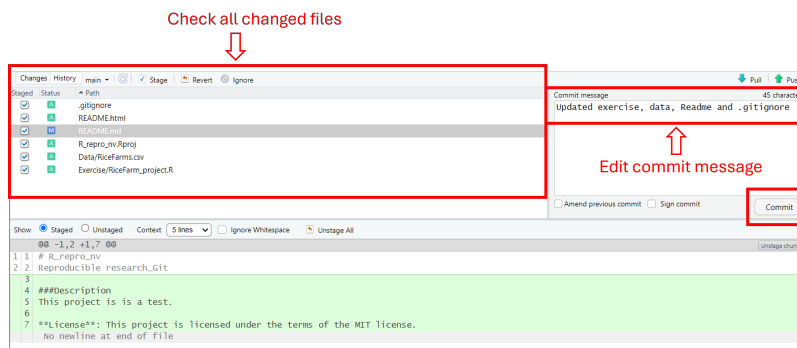
- Open the .gitignore file and add .html in a new line and save the file



5: Editing the .gitignore file

Step 7: Commit and push

- Click the **Commit** button in the Git tab
- Check all the files listed in the top left section
- Write a sentence describing the changes i.e. *“Updated exercise, data, Readme and .gitignore”*
- Click **Commit** and close the window
- click **Push** in the Git tab, a window will pop up showing the interface with the remote system and details of the upload.



6: Commit changes

Great! You have finished your first project update (local changes committed and pushed) via Git on Rstudio.