Using Git within RStudio

Paulina Jedynak, PhD
github.com/paujedynak

Environmental Epidemiology lab
gricad-gitlab.univ-grenoble-alpes.fr

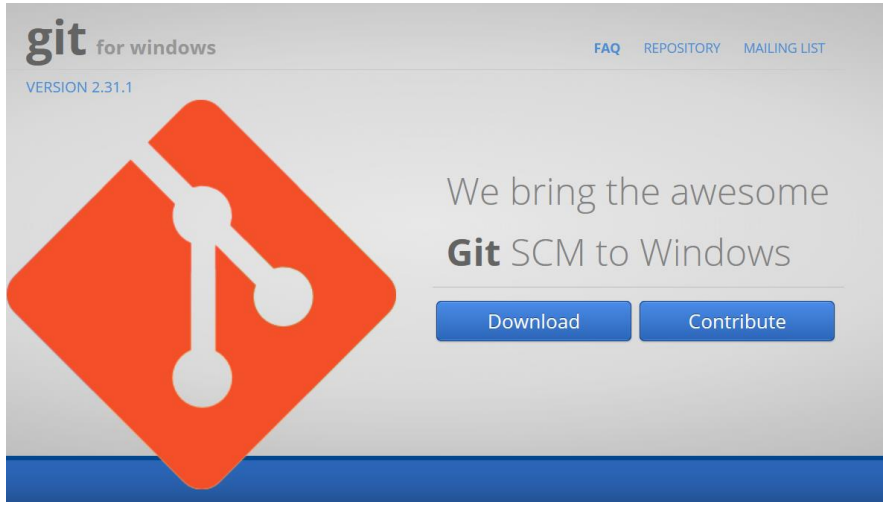
May 2021

Using Git within RStudio

This demo will be based on Windows OS and GitHub platform but do not worry – most of the procedures are common for different systems and platforms

Using Git within RStudio

This demo will be based on Windows OS and GitHub platform but do not worry – most of the procedures are common for different systems and platforms



After this workshop you should be able to:

- Install all the necessary software to use Git version control with RStudio
- Configure Git to communicate with an online repository (e.g. GitHub)
- 'Stage', 'commit', 'push' and 'pull' your code to the online repository

More details on each step discussed here you will find at: happygitwithr.com

If you prefer less details but straight to the poing, try cfss.uchicago.edu/setup/git-with-rstudio

1. Initial installations

Half the battle

Getting all the necessary software installed, configured, and playing nicely together is honestly half the battle when first adopting Git. Brace yourself for some pain. The upside is that you can give yourself a pat on the back once you get through this. And you WILL get through this.

Source: https://happygitwithr.com

1. Initial installations

1. Install or upgrade RStudio

1. Initial installations

1. Install or upgrade RStudio
2. Register an online account where the ‘remote’ version of your repositories will be kept. As an example I will use the GitHub platform but several other platforms exist
   - *github.com*
   - *gitlab.com*
   - *gricad-gitlab.univ-grenoble-alpes.fr* *(login with AGALAN credentials)*
   - …

NOTE: Git != GitHub

https://github.com/paujedynak

Search or jump to... / Pull requests Issues Marketplace Explore

Overview | Repositories 26 | Projects | Packages

Paulina Jedynak

paujedynak

Can be easily changed at any time

Doing biostatistics and bioinformatics at University of Grenoble-Alpes in Environmental Epidemiology lab, coding in R and python.

Edit profile

3 followers · 13 following · 1

University Grenoble-Alpes

Grenoble

paulina.jedynak@gmail.com

https://www.linkedin.com/in/pau-jedyn...

Pinned

Customize your pins

jedynak_prenatal_2020

Git repo for my first article in environmental epidemiology: Jedynak et al., "Prenatal exposure to a wide range of environmental chemicals and child behaviour between 3 and 7 years of age - An expo...

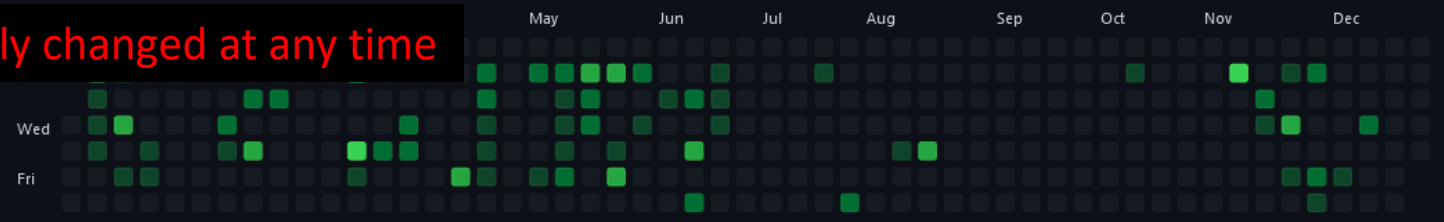
TeX 1

reffree_cell_mix_tutorial

Example of an estimation of cell-type proportions using reference-free method (RefFreeEWAS)

348 contributions in 2020

Contribution settings



Learn how we count contributions

Less More

Contribution activity

December 2020

Created 22 commits in 1 repository

paujedynak/Jedynak2021EHP 22 commits

2021

2020

2019

2018

1. Initial installations

1. Install or upgrade RStudio
2. Register an online account where the ‘remote’ version of your repositories will be kept. As an example I will use the GitHub platform but several other platforms exist
3. Install Git software on your machine (accept the default setup options!)

   git-scm.com/downloads

   You can check if you already have Git by typing `where.exe git` (Windows)

   or `which git` (Mac, Linux) in the shell

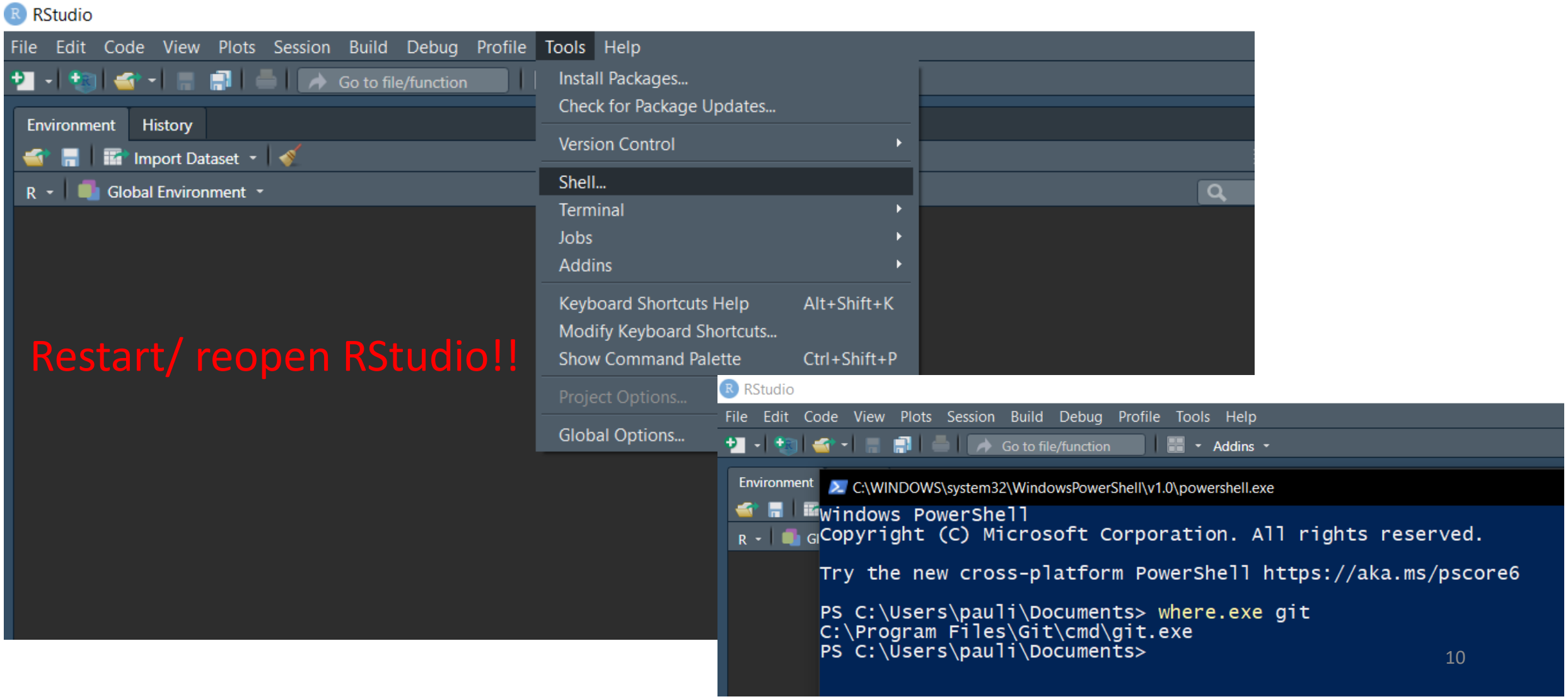
1. Initial installations

1. Install or upgrade RStudio
2. Register an online account where the ‘remote’ version of your repositories will be kept. As an example I will use the GitHub platform but several other platforms exist
3. Install Git software on your machine (accept the default setup options!)

   git-scm.com/downloads

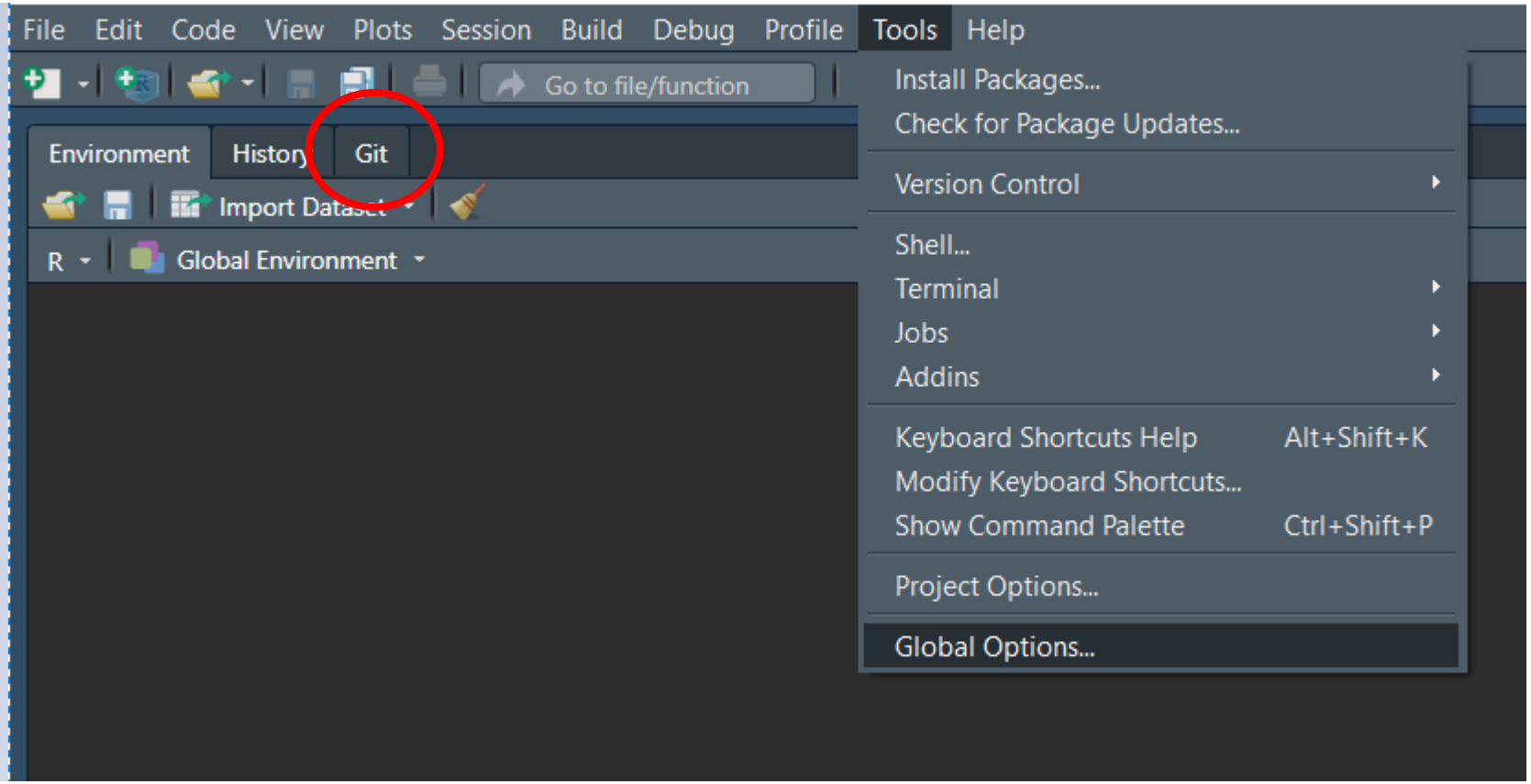
   You can check if you already have Git by typing `where.exe git` (Windows) or `which git` (Mac, Linux) in the shell
4. Open RStudio

1. Initial installations



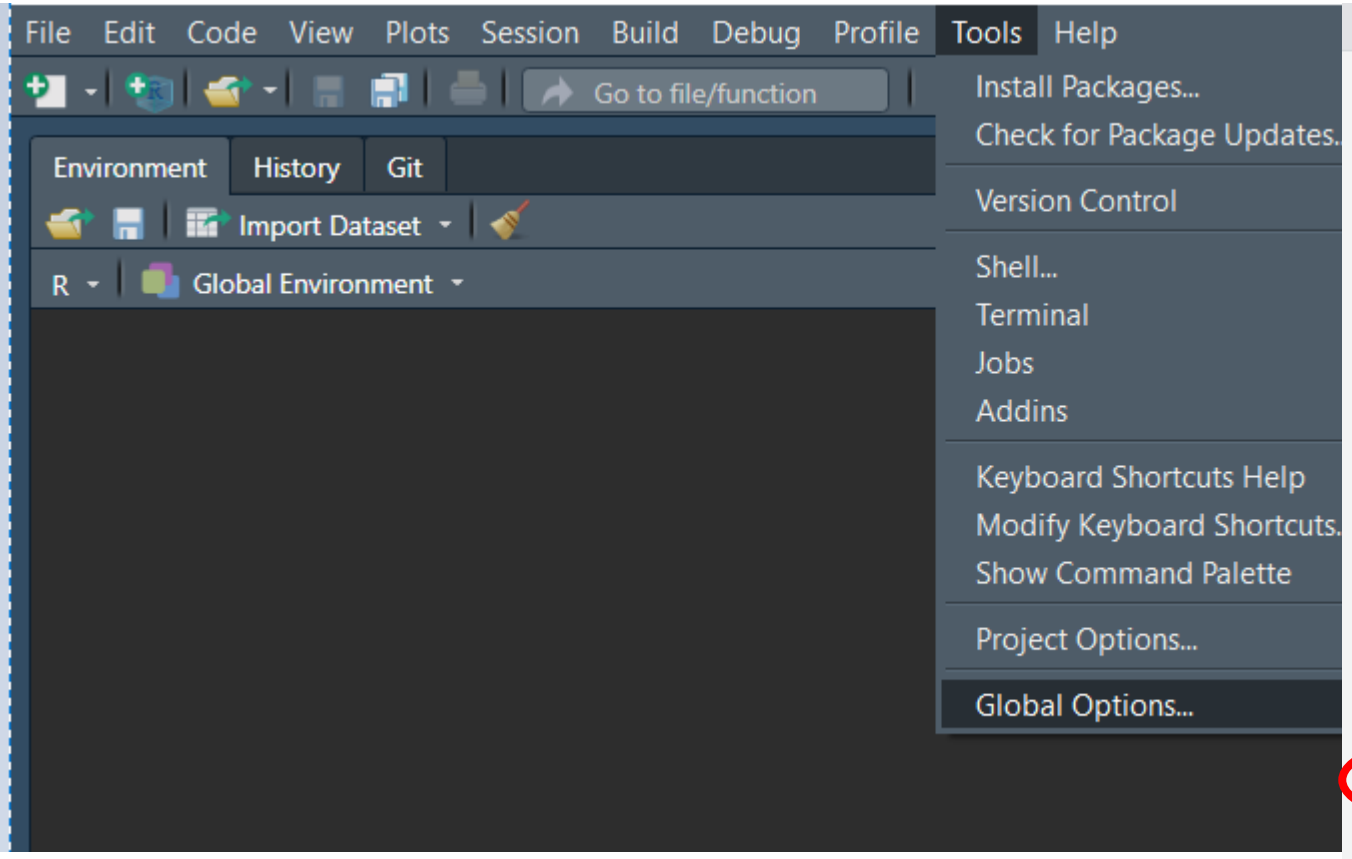
1. Initial installations

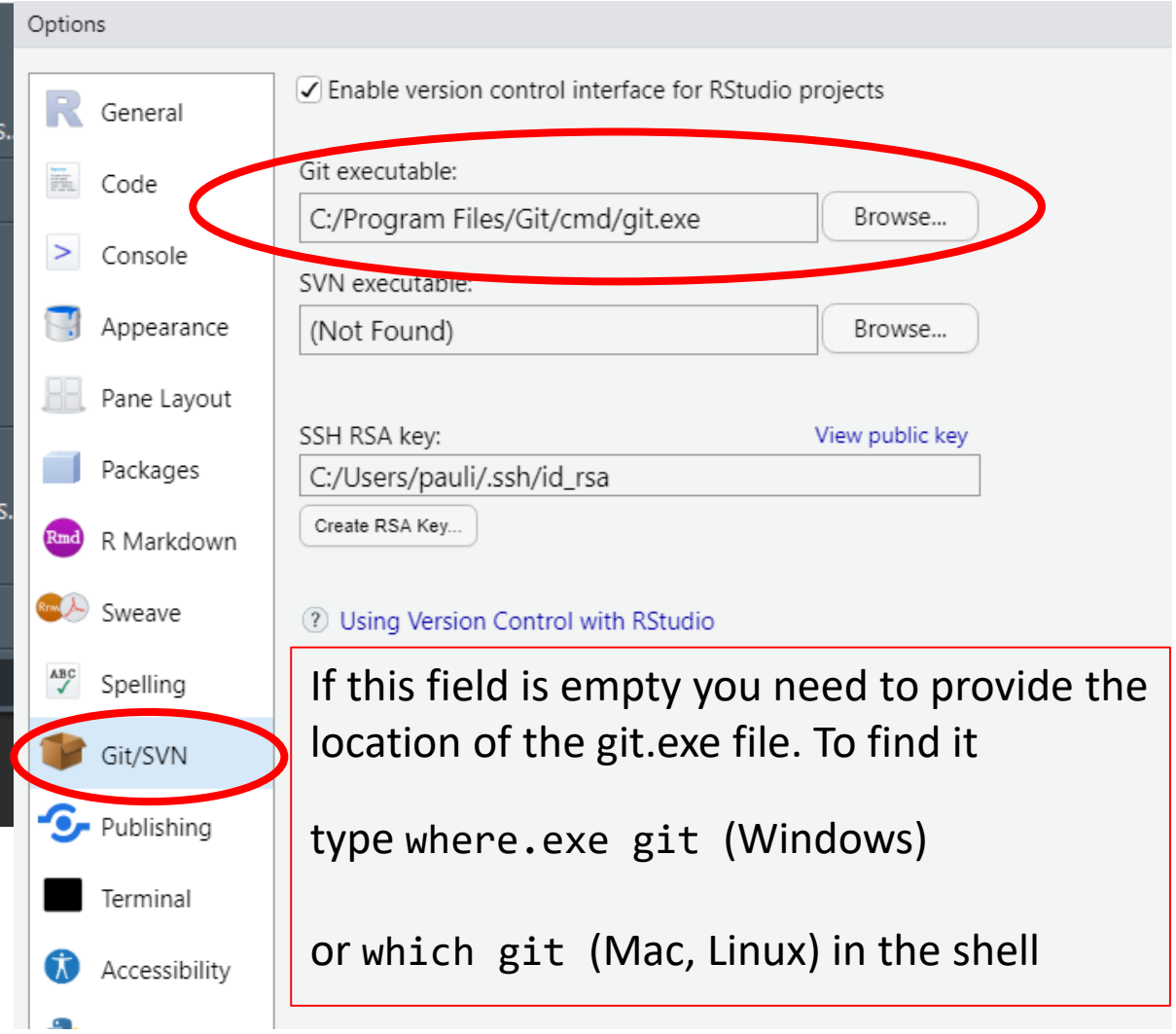
Make sure the RStudio “sees” Git (in Windows, after installation Git should be recognized automatically)



1. Initial installations

Make sure the RStudio “sees” Git (in Windows, after installation Git should be recognized automatically)





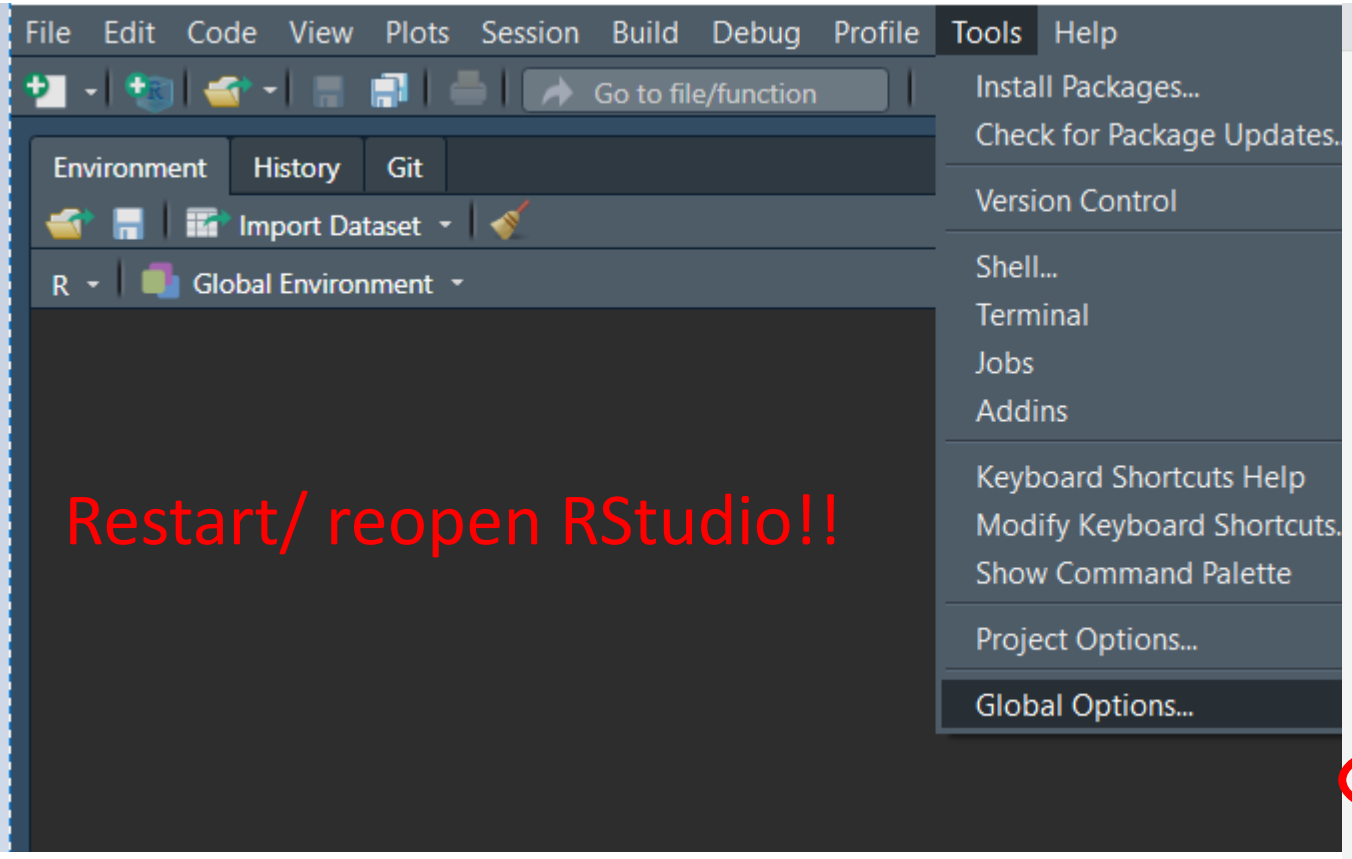
If this field is empty you need to provide the location of the git.exe file. To find it

type `where.exe git` (Windows)

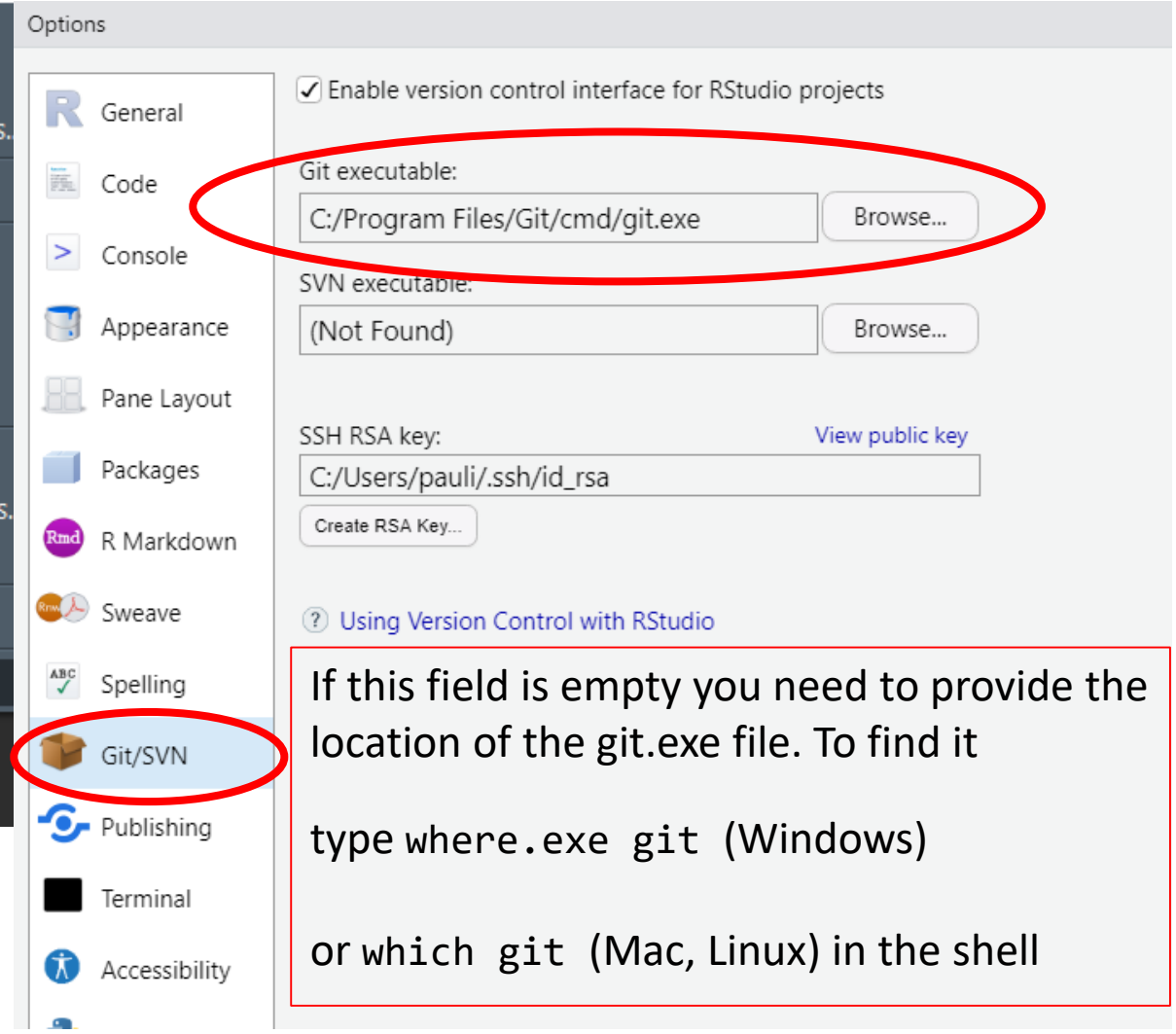
or `which git` (Mac, Linux) in the shell

1. Initial installations

Make sure the RStudio “sees” Git (in Windows, after installation Git should be recognized automatically)



Restart/ reopen RStudio!!



If this field is empty you need to provide the location of the git.exe file. To find it

type `where.exe git` (Windows)

or `which git` (Mac, Linux) in the shell

2. Introduce yourself to Git

```
## install if needed (do this exactly once):
## install.packages("usethis")

library(usethis)
use_git_config(user.name = "Jane Doe", user.email = "jane@example.org")
```

2. Introduce yourself to Git

```
## install if needed (do this exactly once):
## install.packages("usethis")

library(usethis)
use_git_config(user.name = "Jane Doe", user.email = "jane@example.org")
```

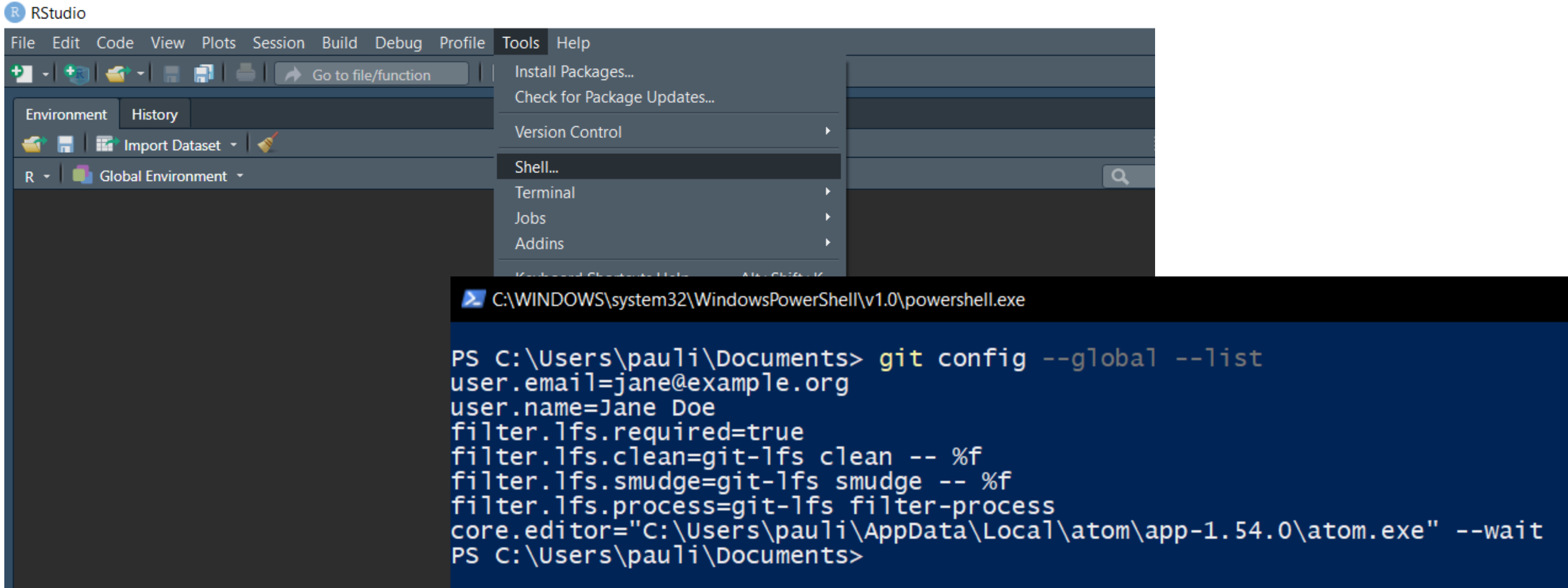
This **does not have to be** your GitHub user name

This **must be** the email associated with GitHub account

2. Introduce yourself to Git

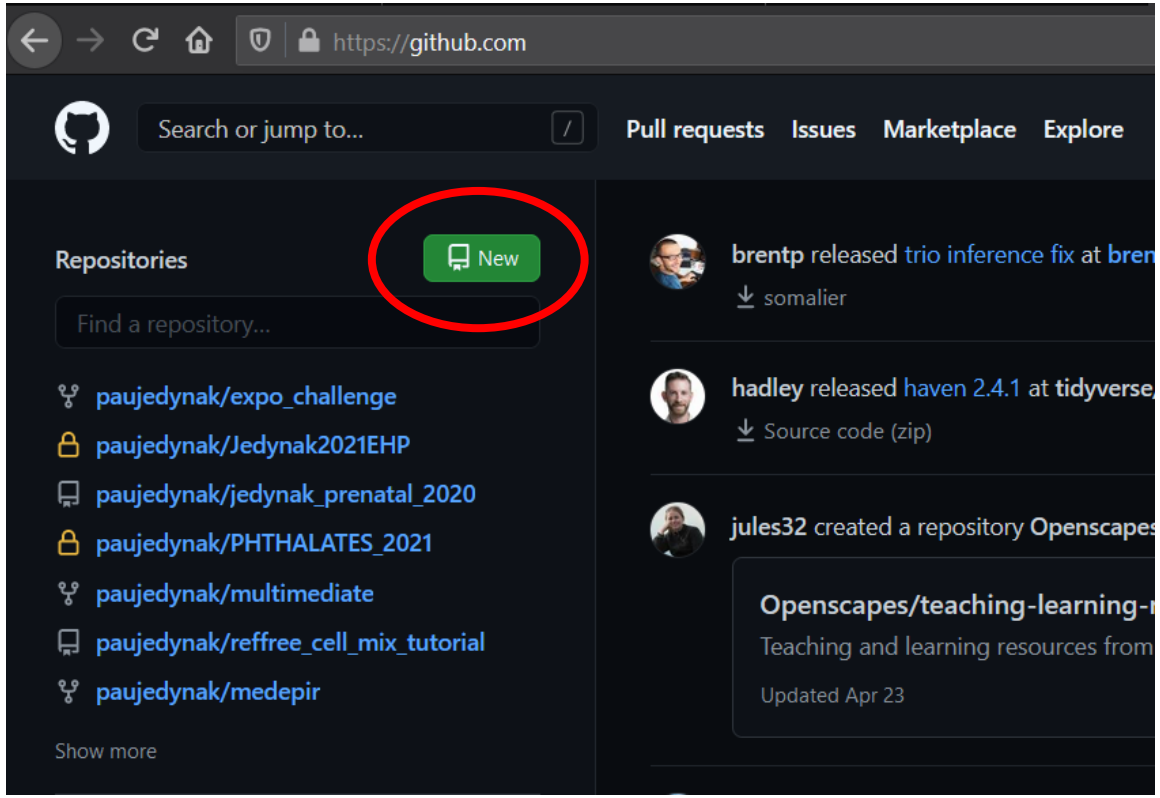
You may check whether Git understood what you typed by using the command

`git config --global –list` (again in the shell)



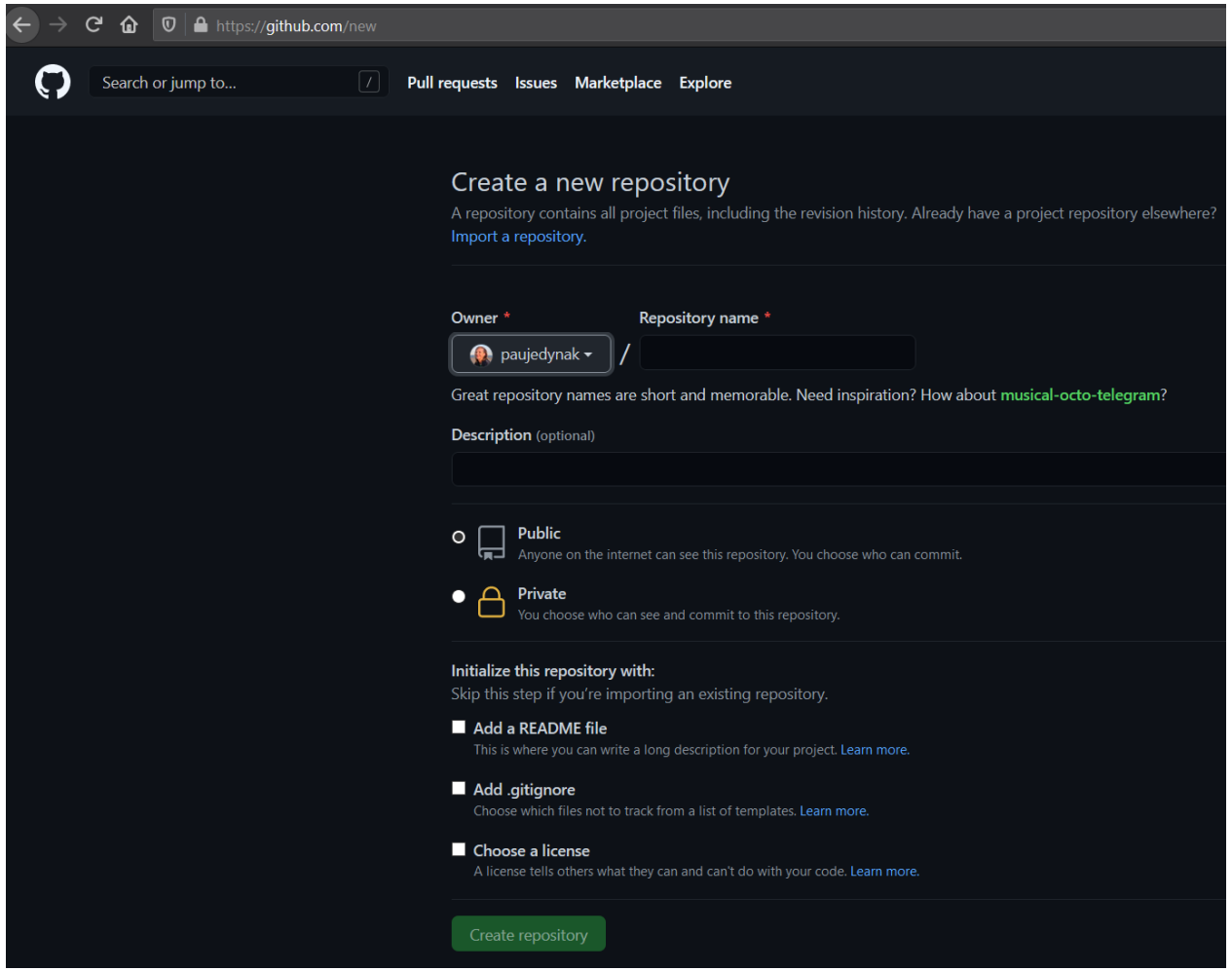
3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button



3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

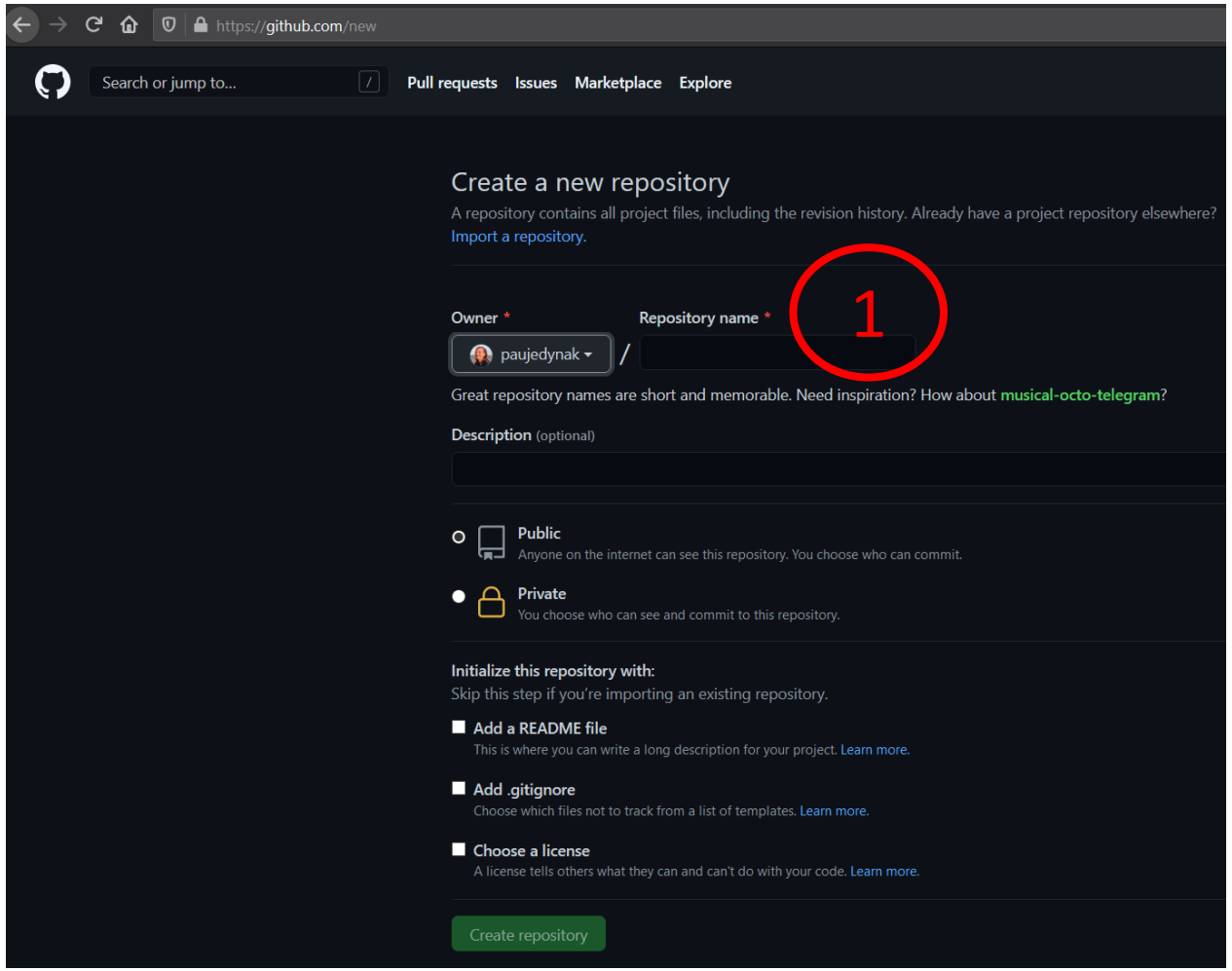


3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

How to fill this in:

1. Repository name: e.g. test

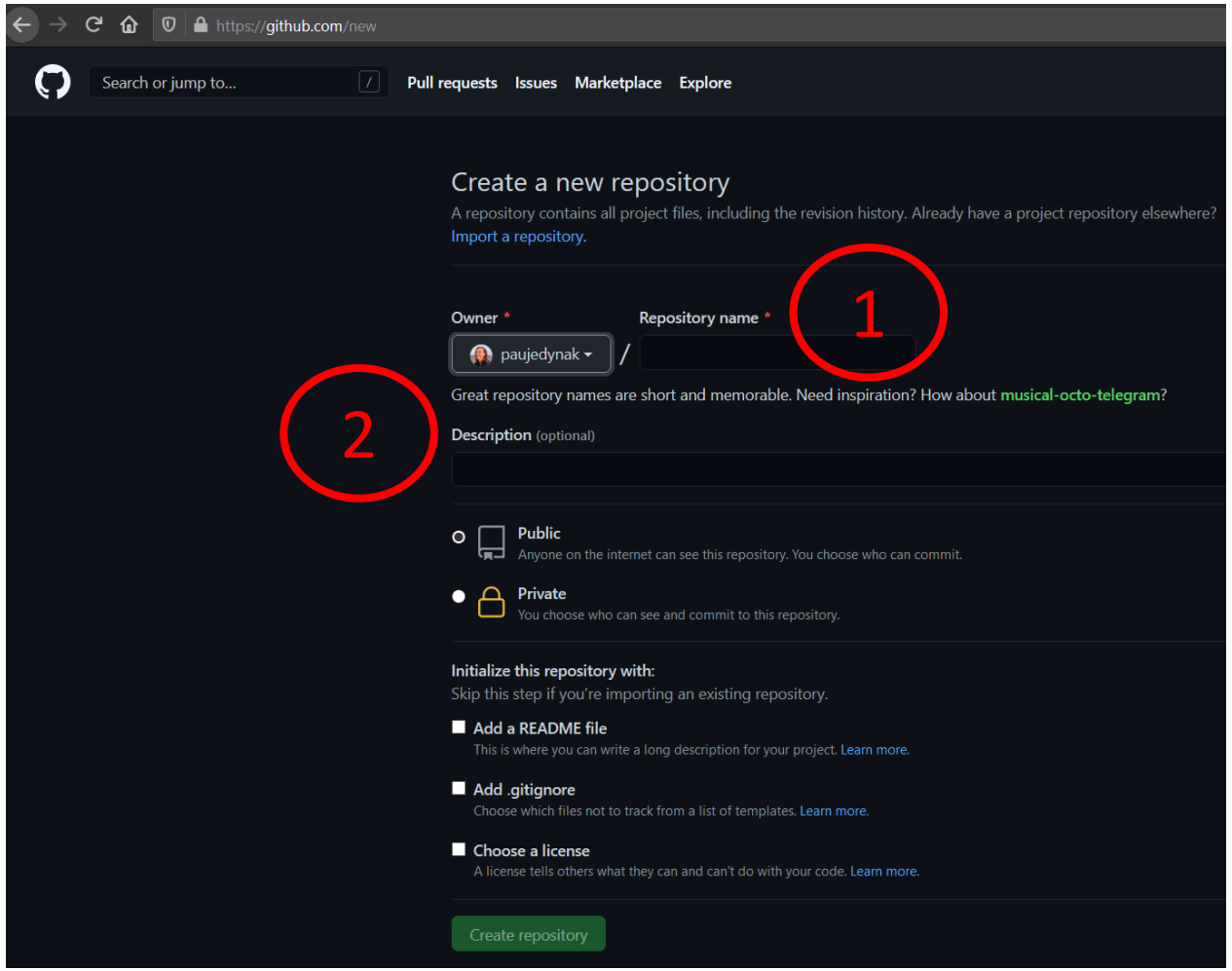


3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

How to fill this in:

1. Repository name: e.g. test
2. Description: optional but useful



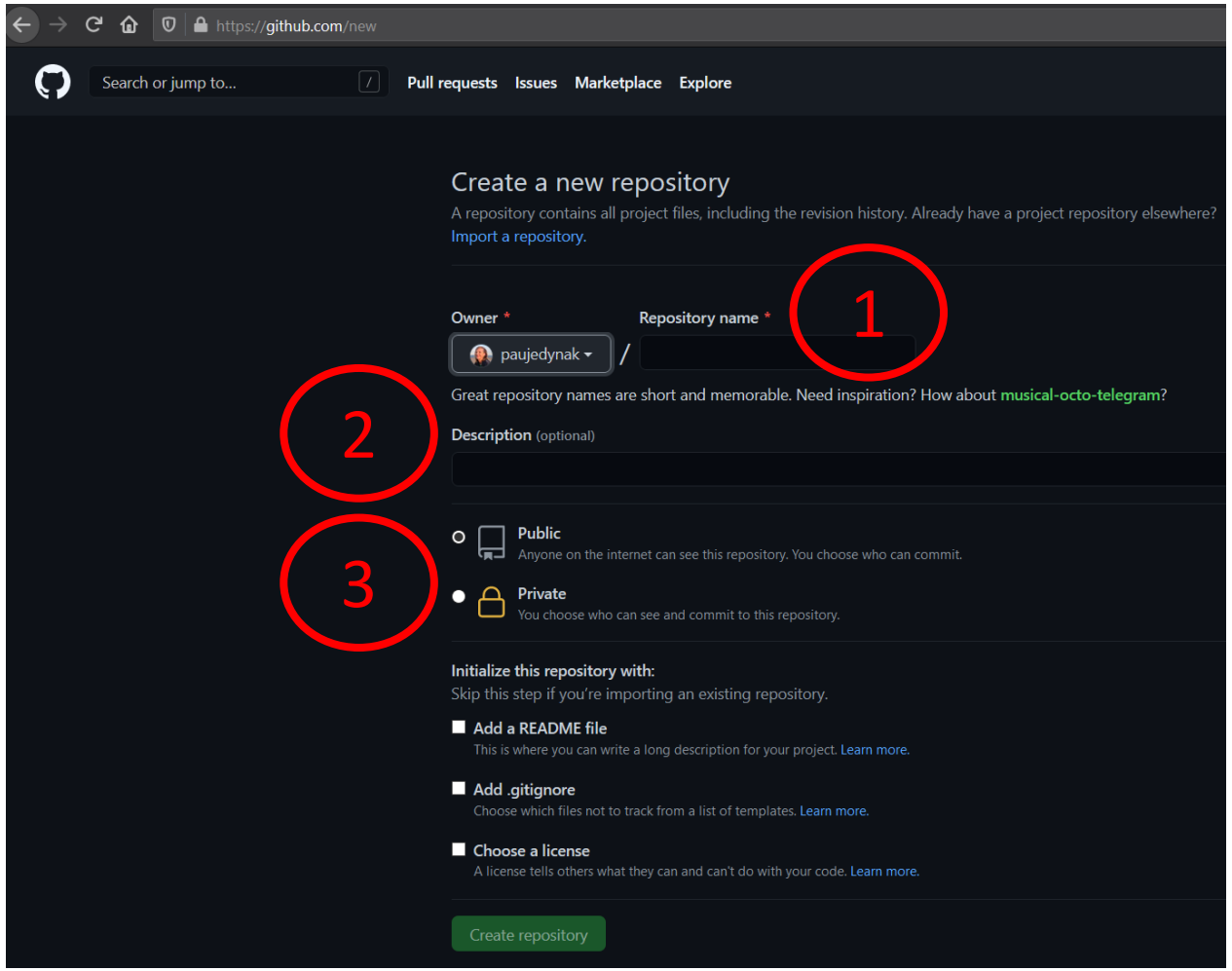
3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

How to fill this in:

1. Repository name: e.g. test
2. Description: optional but useful
3. Choose visibility: public or private*

*Up to 3 collaborators



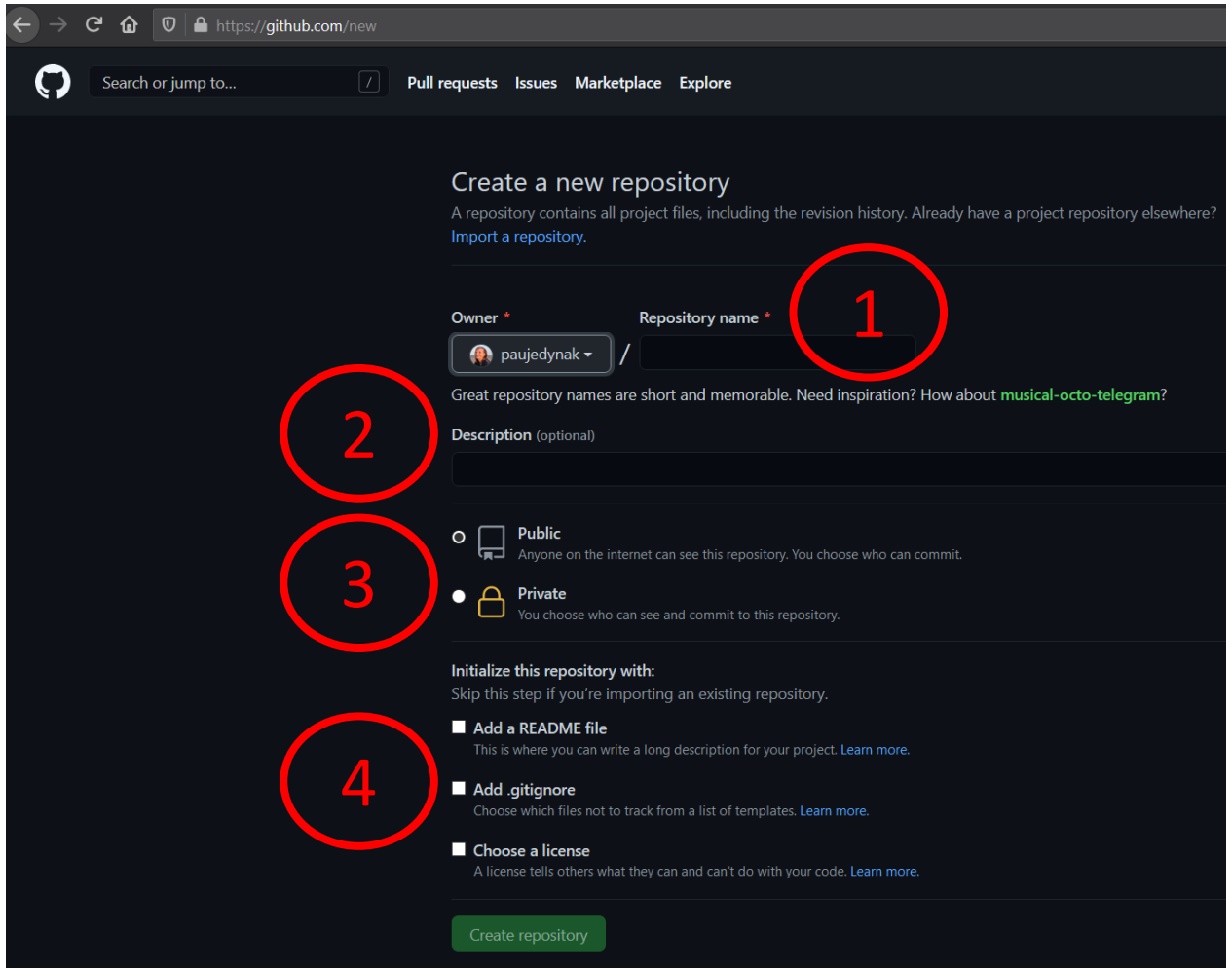
3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

How to fill this in:

1. Repository name: e.g. test
2. Description: optional but useful
3. Choose visibility: public or private*
4. Optional: add README if needed;

   add .gitignore file** with R template



*Up to 3 collaborators

**Enables ‘hiding’ some files from being traced by Git

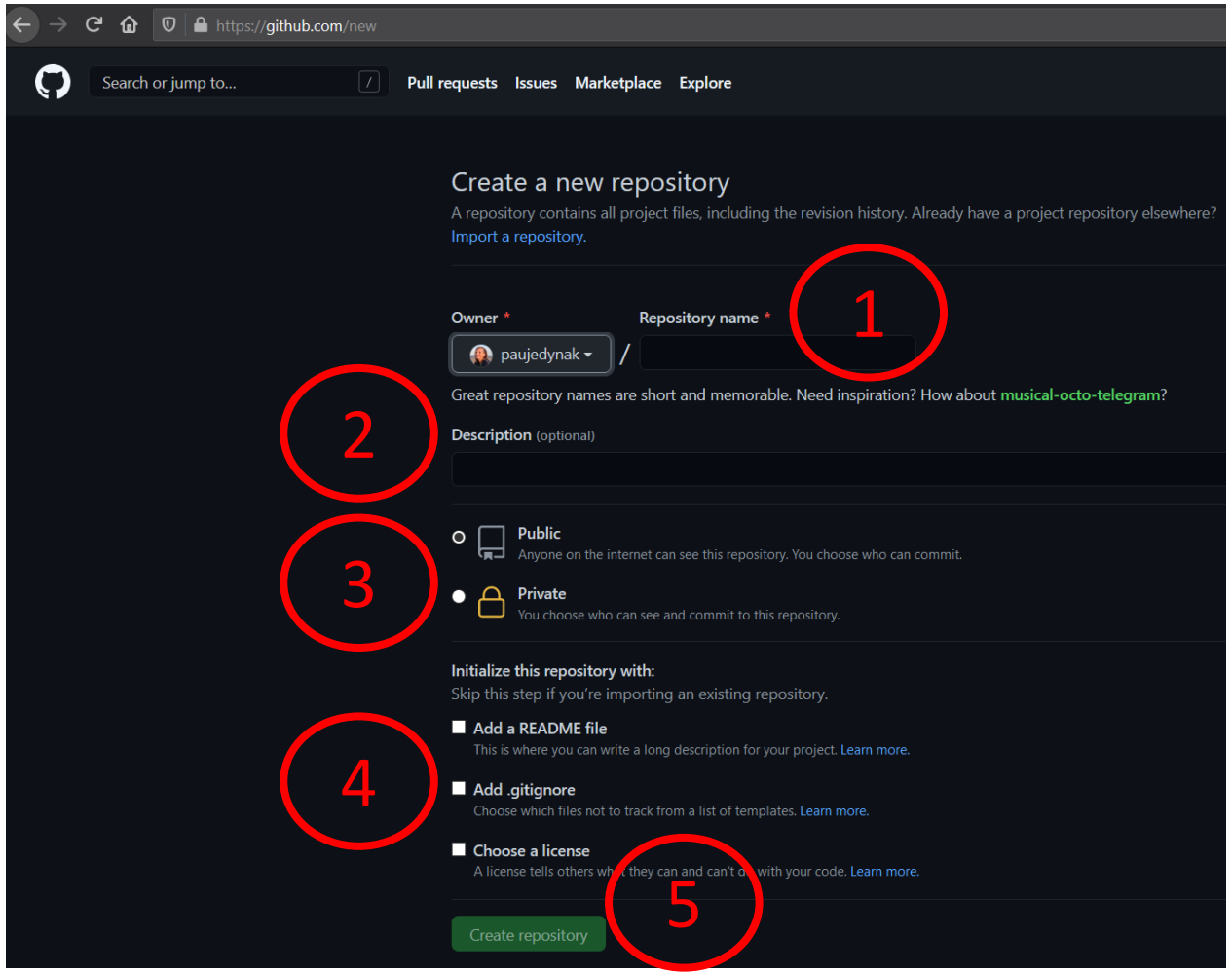
3. Connect to GitHub

1. Go to https://github.com and make sure you are logged in
2. Click green “New” button. Or, if you are on your own profile page, click first on “Repositories”, then click the green “New” button

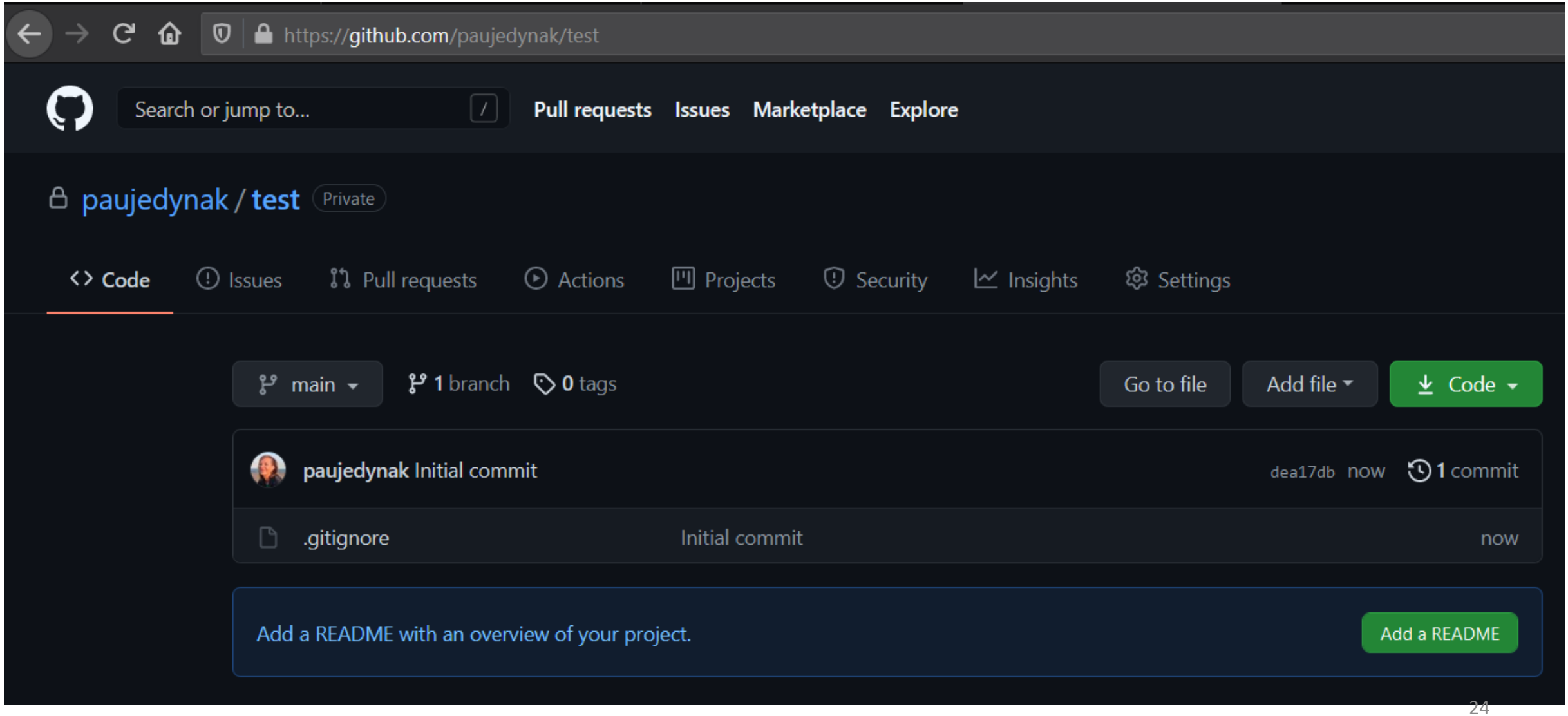
How to fill this in:

1. Repository name: e.g. test
2. Description: optional but useful
3. Choose visibility: public or private*
4. Optional: add README if needed; add .gitignore file** with R template
5. Click green button “Create repository”

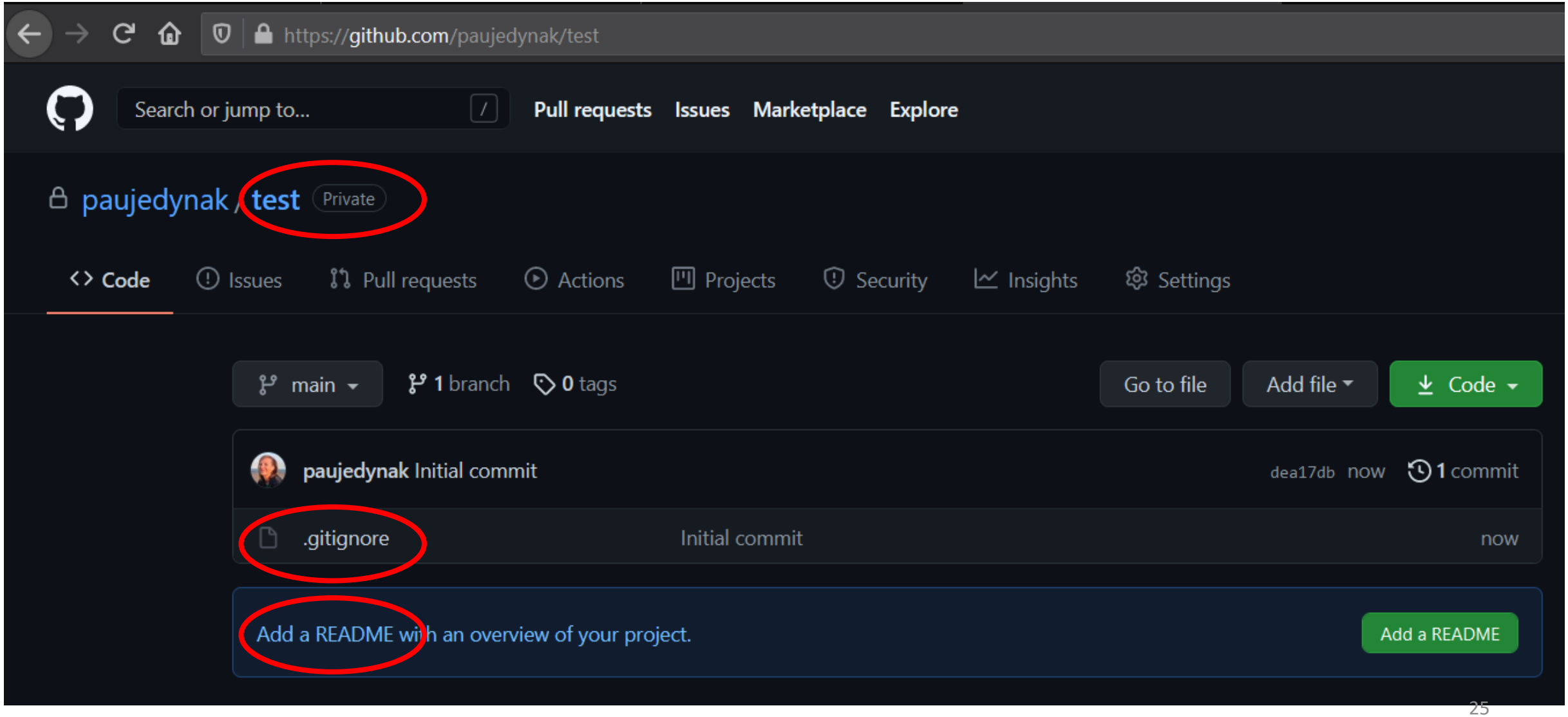
*Up to 3 collaborators
**Enables ‘hiding’ some files from being traced by Git



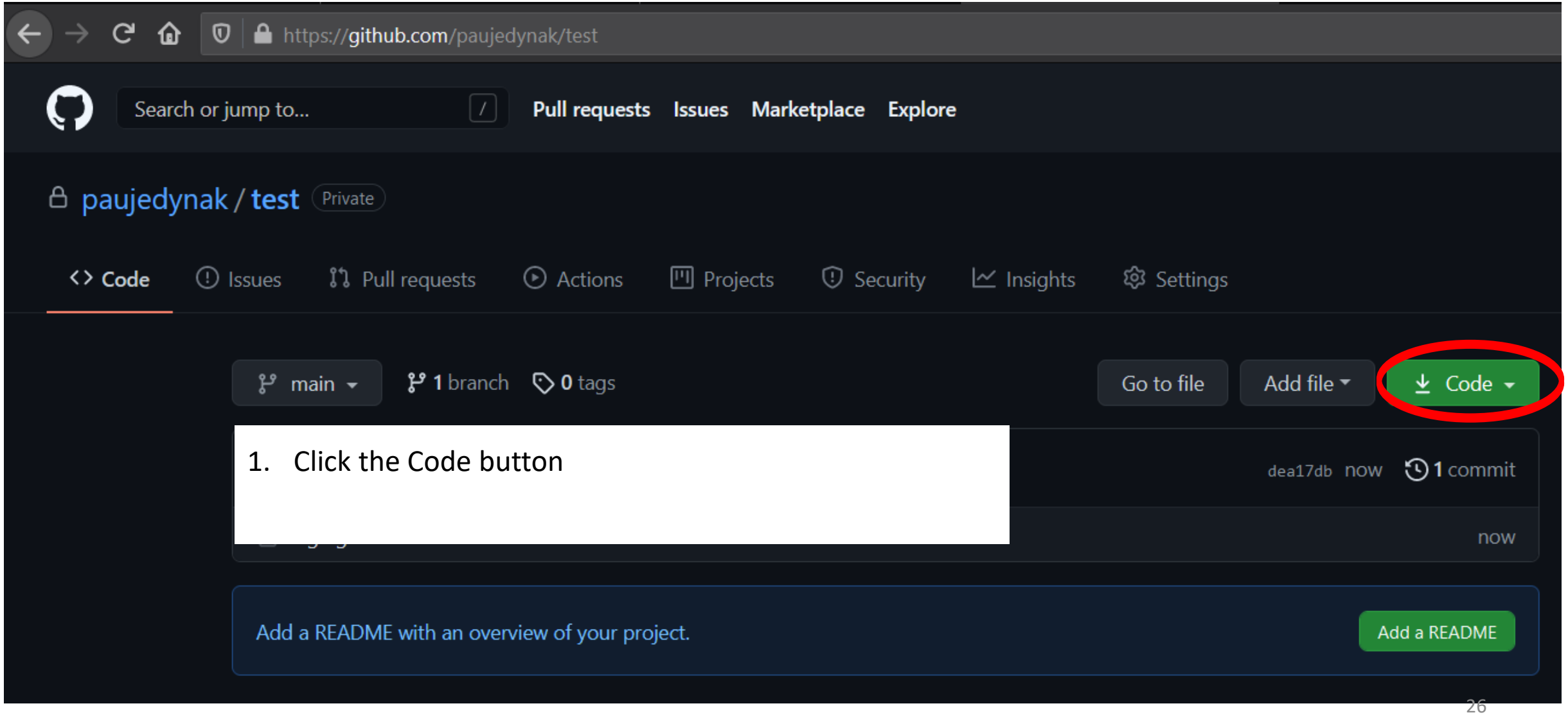
3. Connect to GitHub



3. Connect to GitHub



3. Connect to GitHub



1. Click the Code button

https://github.com/paujedynak/test

Search or jump to... / Pull requests Issues Marketplace Explore

paujedynak / test Private

Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

1. Click the Code button
2. Copy the link from the HTTPS tab to your Clipboard

Add a README with an overview of your project.

Clone

HTTPS SSH GitHub CLI

https://github.com/paujedynak/test.git

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

https://github.com/paujedynak/test

Search or jump to... / Pull requests Issues Marketplace Explore

paujedynak / test Private

Code Issues Pull requests Actions Projects Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

1. Click the Code button
2. Copy the link from the HTTPS tab to your Clipboard

Clone

HTTPS SSH GitHub CLI

https://github.com/paujedynak/test.git

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

IF YOU CANNOT FIND THE ‘CODE’ BUTTON, DON’T PANIC!

It means you created an empty repository (without README, .gitignore or any other file). As soon you add a file to the repo (e.g. README), the button will appear

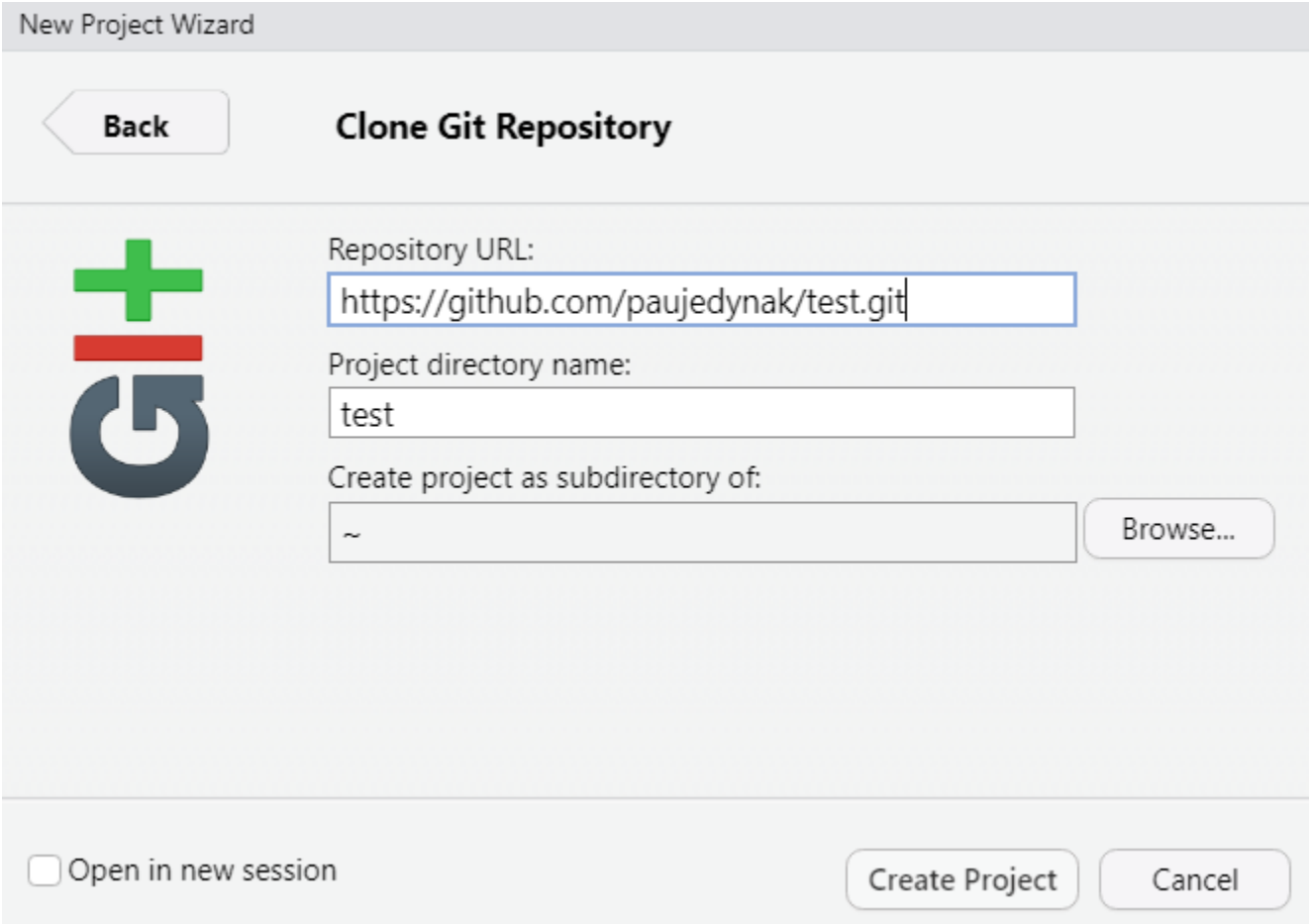
4. Clone the repo to your local machine

Now you have 2 options depending if you are starting a new R Project or you already have an existing R Project that you want to connect with Git

4. Clone the repo to your local machine

Brand new R Project:

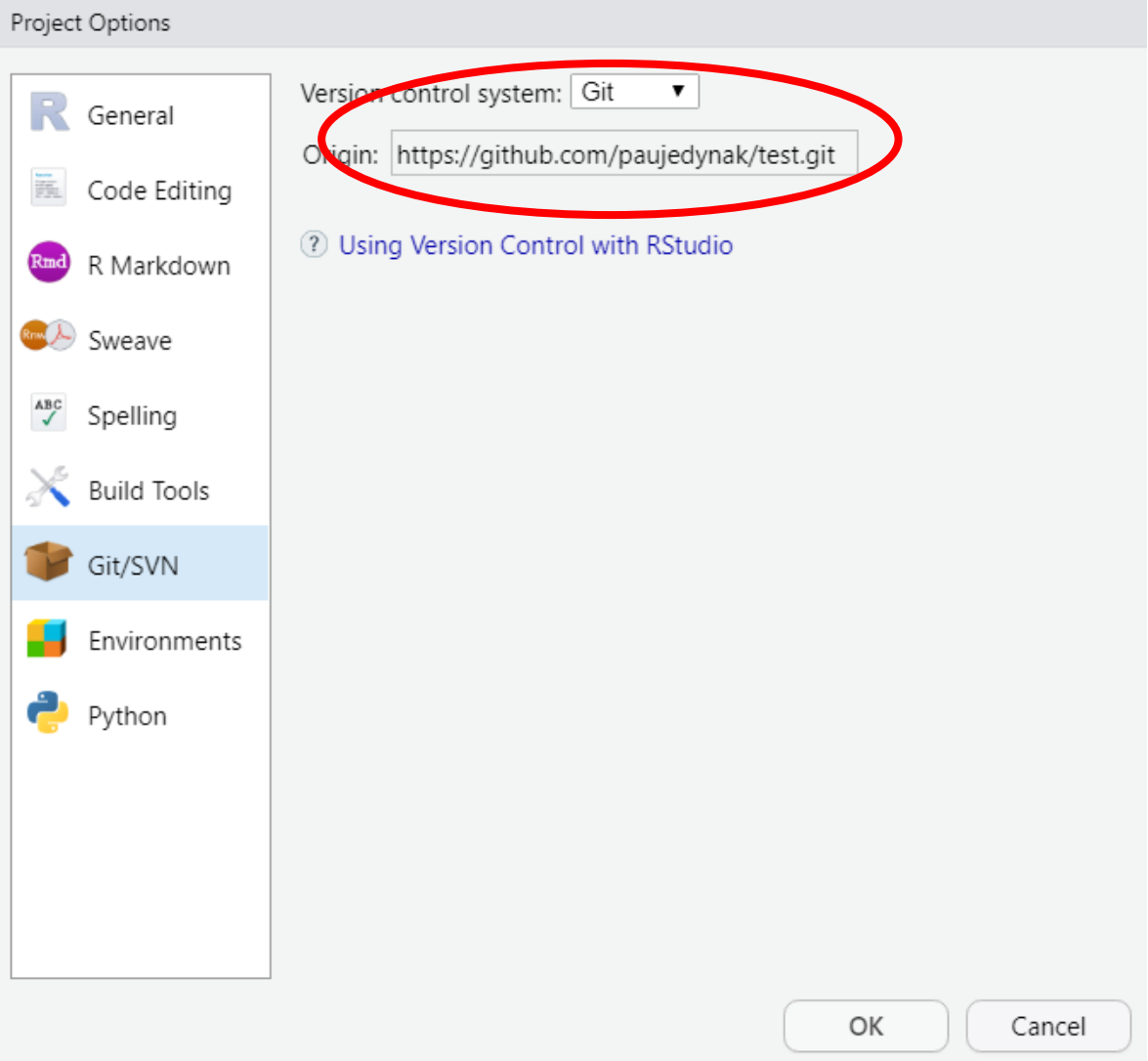
1. In RStudio >> File >> New Project…
2. Choose Version Control
3. Choose Git
4. Paste the link to your repository into the Repository URL field (the Project directory name will be filled automatically)
5. Click Browse… to choose the location where your new Project will be created
6. Click Create Project



4. Clone the repo to your local machine

Brand new R Project:

1. In RStudio >> File >> New Project…
2. Choose Version Control
3. Choose Git
4. Paste the link to your repository into the Repository URL field (the Project directory name will be filled automatically)
5. Click Browse… to choose the location where your new Project will be created
6. Click Create Project



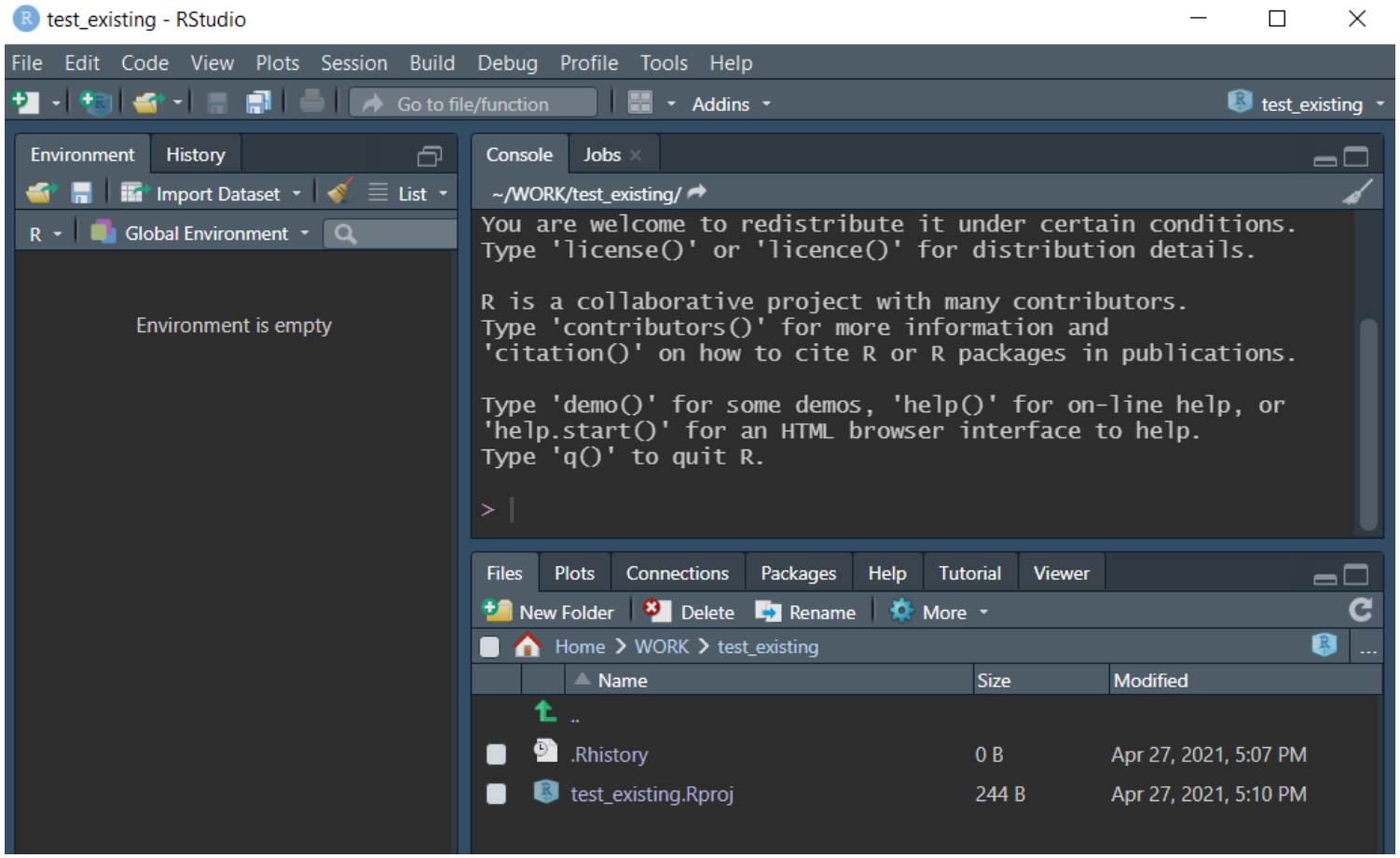
4. Clone the repo to your local machine

Brand new R Project:

1. In RStudio >> File >> New Project…
2. Choose Version Control
3. Choose Git
4. Paste the link to your repository into the Repository URL field (the Project directory name will be filled automatically)
5. Click Browse… to choose the location where your new Project will be created
6. Click Create Project

You can commit, push and pull…

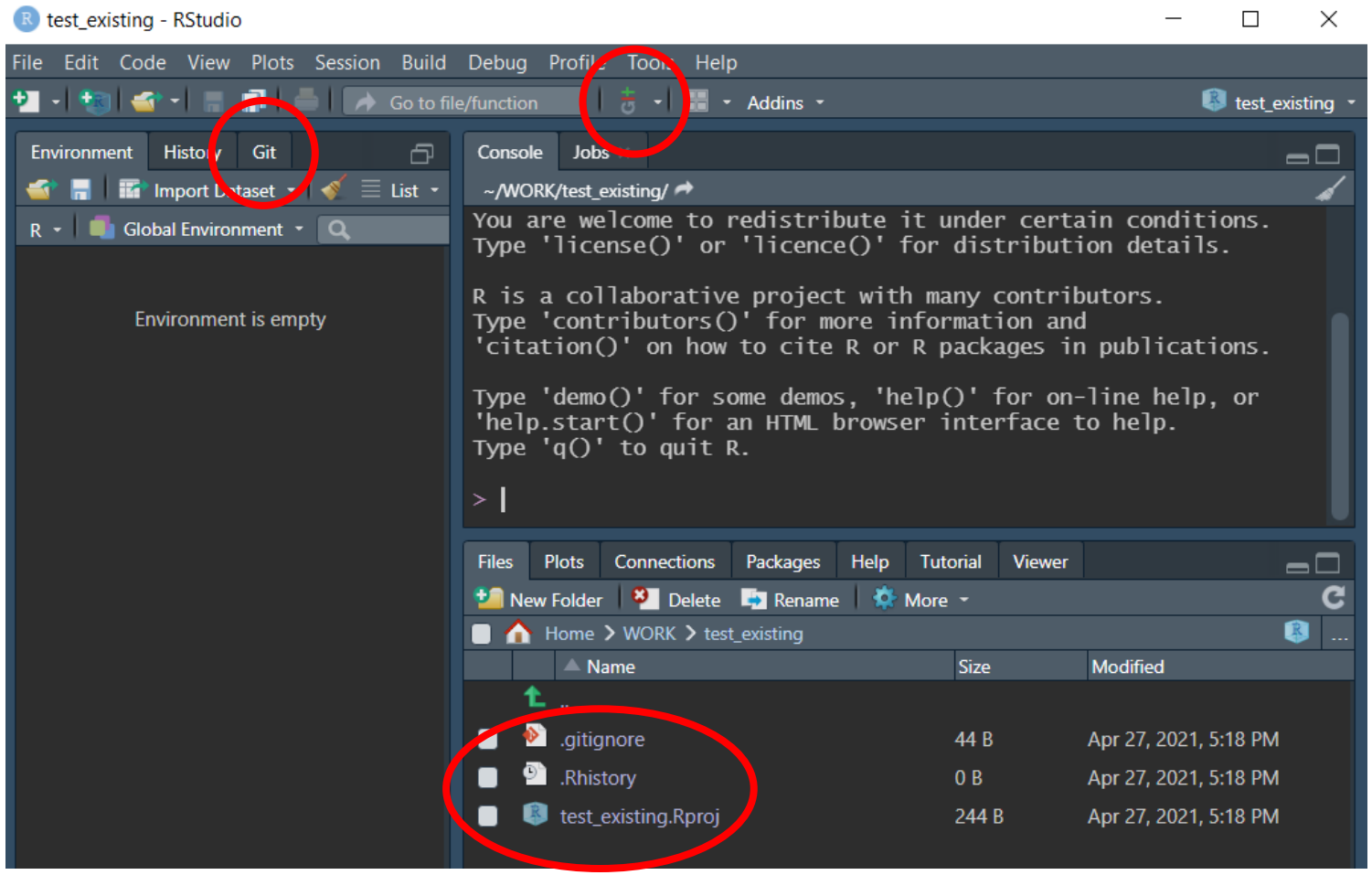
5. Connect Git to an existing R project



Existing R Project:

1. Open an existing Project

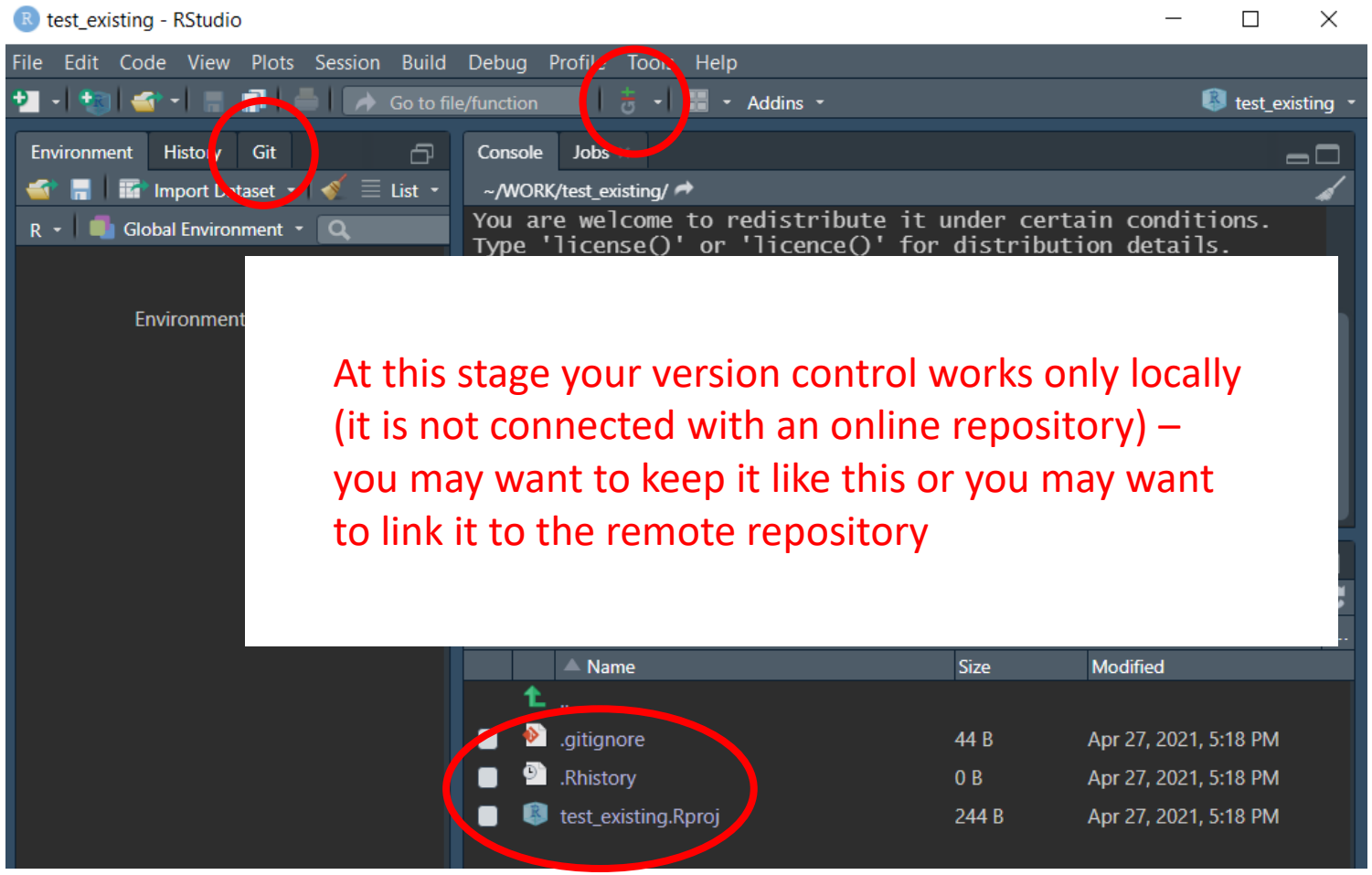
5. Connect Git to an existing R project



Existing R Project:

1. Open an existing Project
2. Tools >> Version Control >> Project Setup…
3. Choose Git as Version control system
4. Accept the following prompts

5. Connect Git to an existing R project

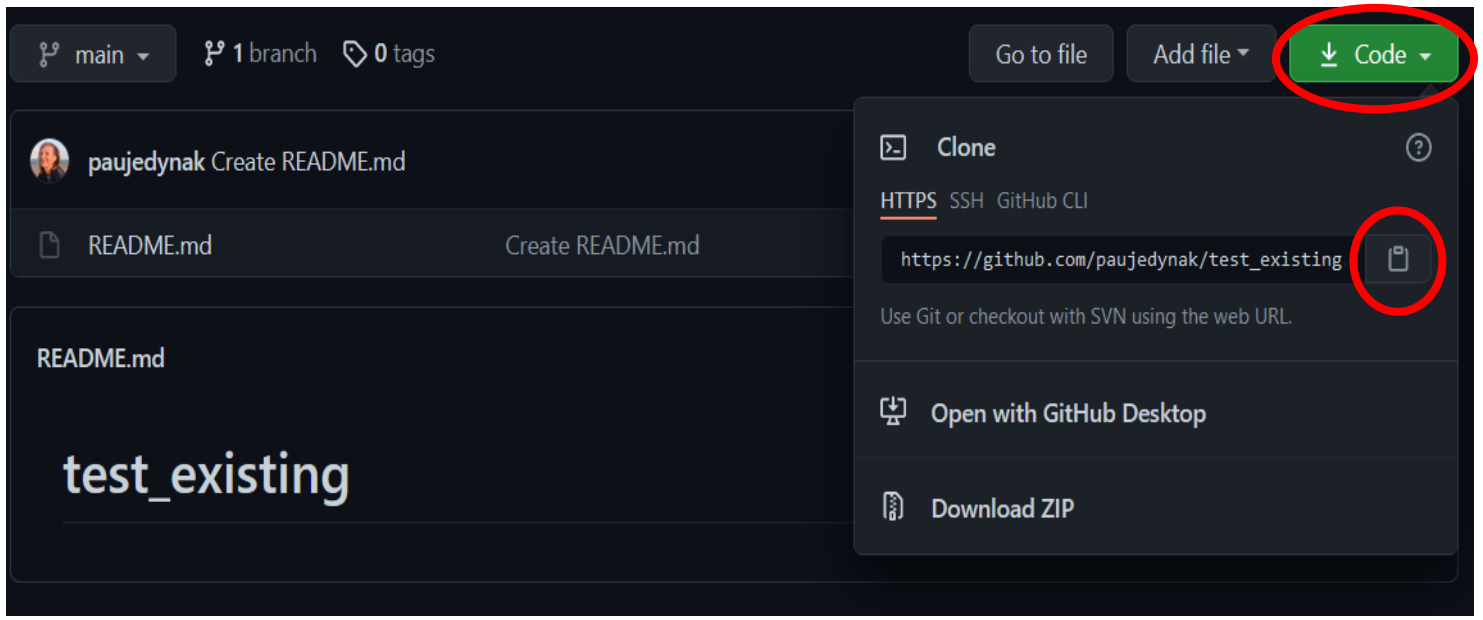


At this stage your version control works only locally (it is not connected with an online repository) – you may want to keep it like this or you may want to link it to the remote repository

Existing R Project:

1. Open an existing Project
2. Tools >> Version Control >> Project Setup…
3. Choose Git as Version control system
4. Accept the following prompts

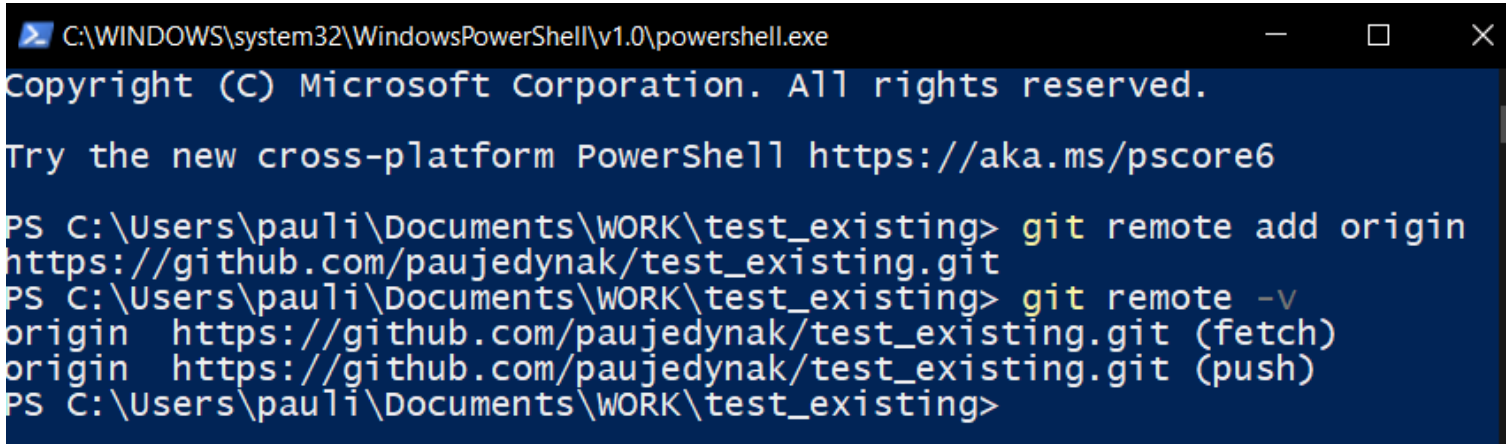
5. Connect Git to an existing R project



Connect to an online repository:

1. Create a remote repository as explained earlier in this presentation (3. Connect to GitHub). IMPORTANT: repo name must match your existing Project name
2. Copy the repo link to your Clipboard

5. Connect Git to an existing R project



Connect to an online repository:

1. Create a remote repository as explained earlier in this presentation (3. Connect to GitHub). IMPORTANT: repo name must match your existing Project name
2. Copy the repo link to your Clipboard
3. Add remote URL to your existing Project using shell by typing:

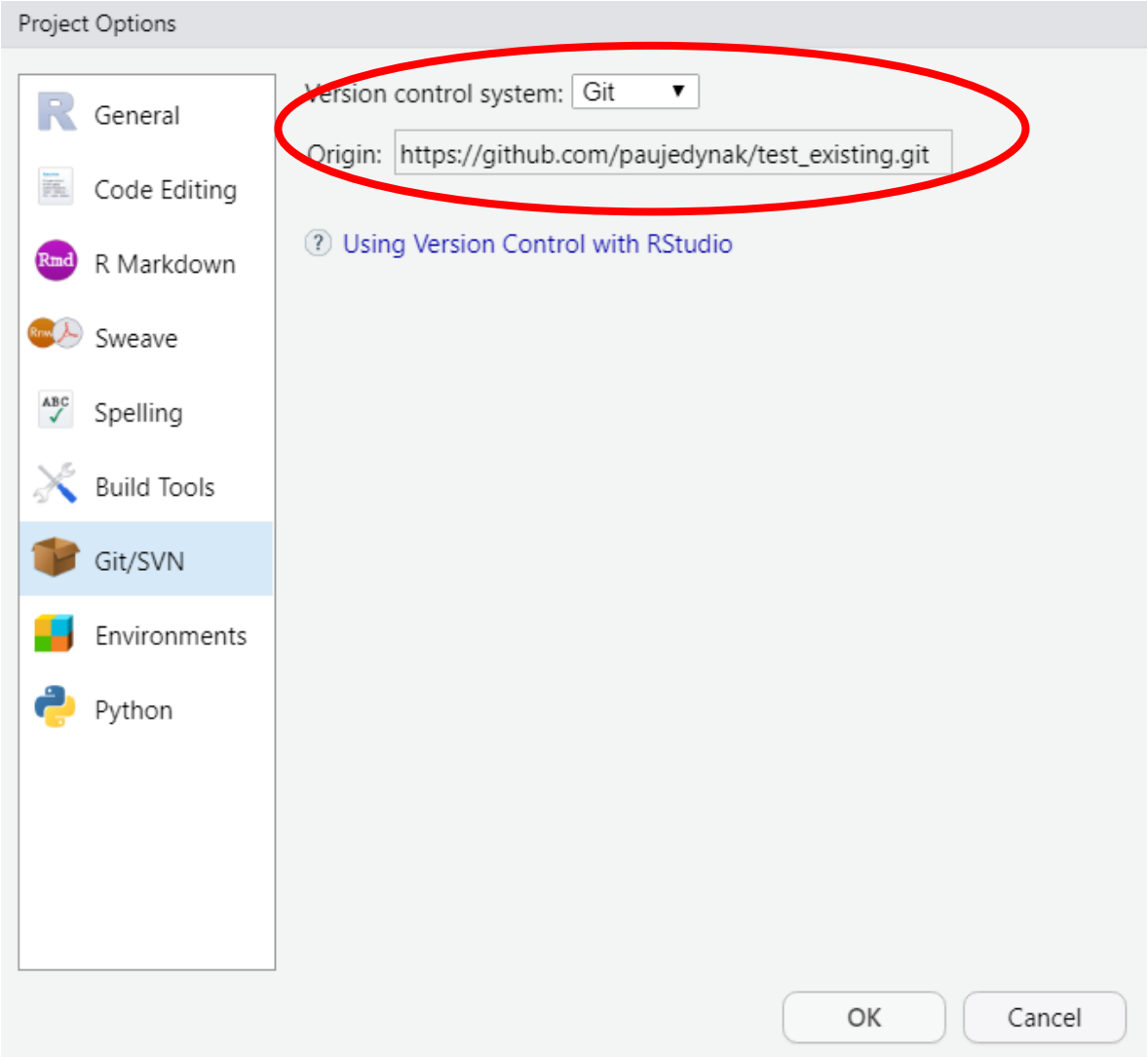
```
git remote add origin https://github.com/username/reponame.git
```

where *username* is your GitHub username and *reponame* is your GitHub repository name

4. Check the remote URL using shell by typing:

```
git remote -v
```

5. Connect Git to an existing R project



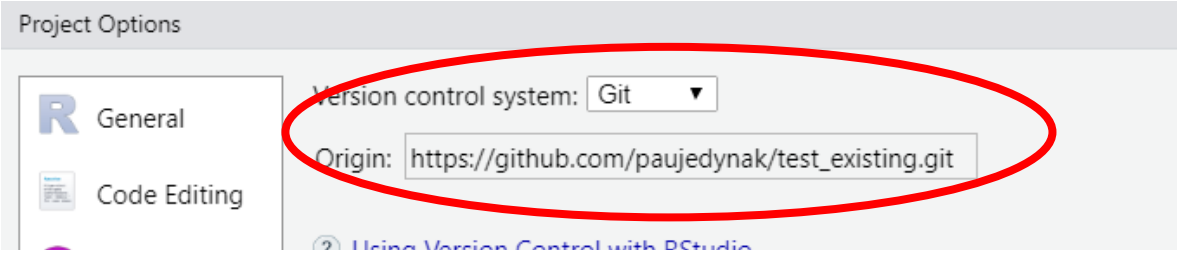
Connect to an online repository:

1. Create a remote repository as explained earlier in this presentation (3. Connect to GitHub). IMPORTANT: repo name must match your existing Project name
2. Copy the repo link to your Clipboard
3. Add remote URL to your existing Project using shell by typing:

4. Check the remote URL using shell by typing:

```
git remote -v
```

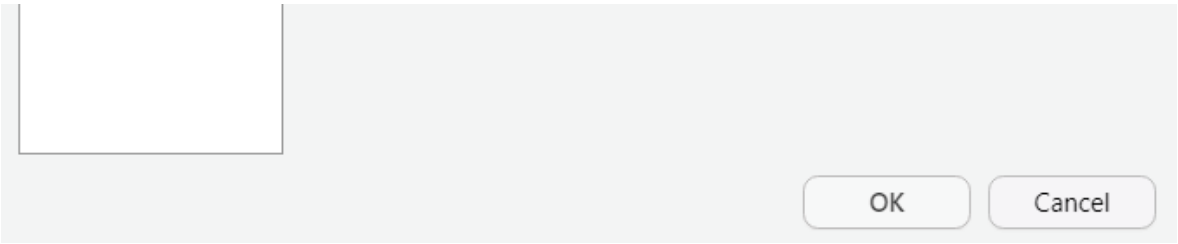
5. Connect Git to an existing R project



IF THIS FIELD IS EMPTY OR 'PUSH' AND 'PULL' BUTTONS ARE GREYED OUT (INACTIVE), DON'T PANIC!

Restart / reopen RStudio. If the problem persists, make some commit and make a push setting the upstream by typing in the shell:

```
git push origin master -u
```



Connect to an online repository:

1. Create a remote repository as explained earlier in this presentation (3. Connect to GitHub). IMPORTANT: repo name must match your existing Project name
2. Copy the repo link to your Clipboard
3. Add remote URL to your existing Project using shell by typing:

4. Check the remote URL using shell by typing:

```
git remote -v
```

5. Connect Git to an existing R project

You can commit, push and pull…

Connect to an online repository:

1. Create a remote repository as explained earlier in this presentation (3. Connect to GitHub). IMPORTANT: repo name must match your existing Project name
2. Copy the repo link to your Clipboard
3. Add remote URL to your existing Project using shell by typing:

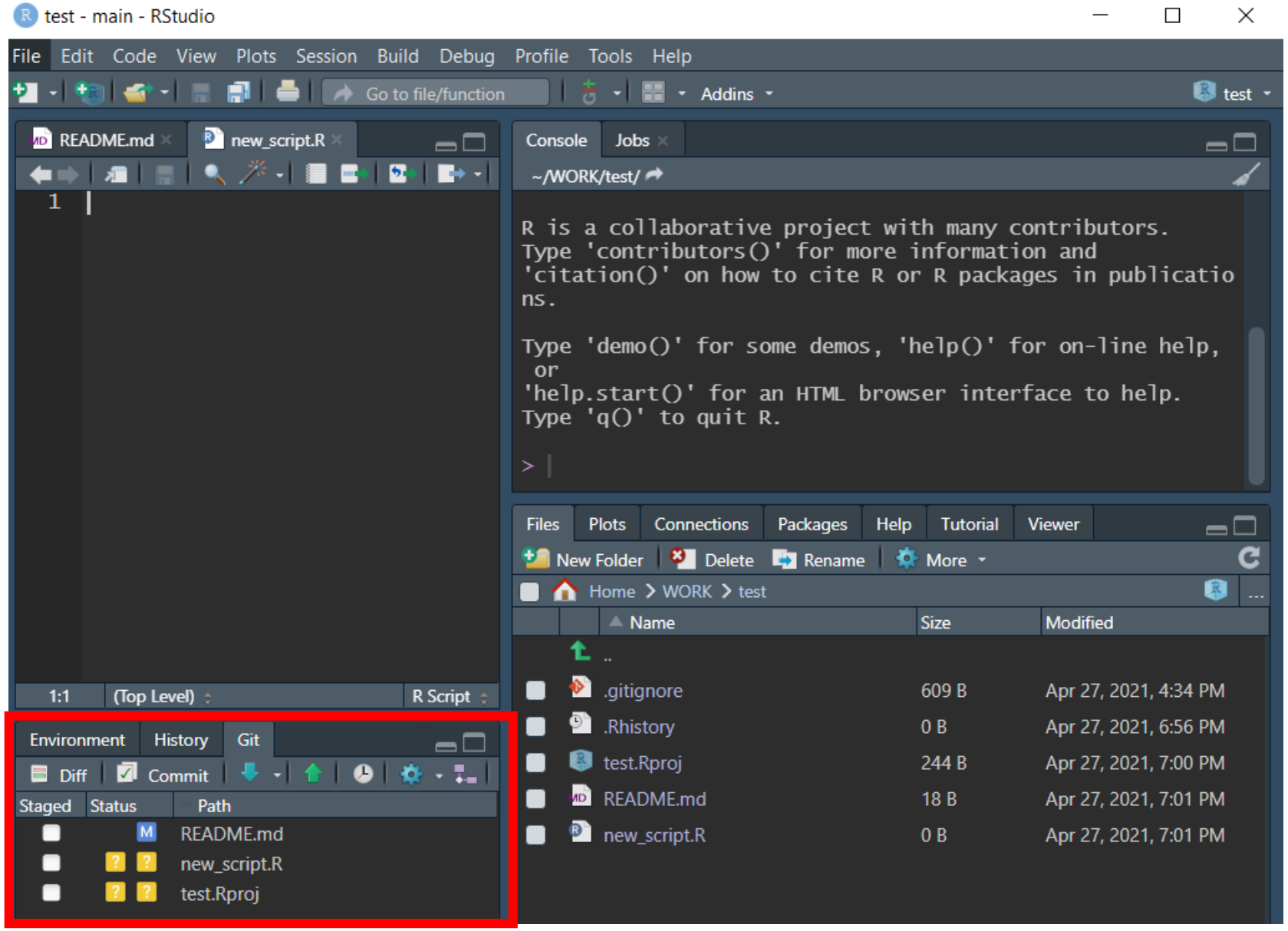
4. Check the remote URL using shell by typing:

```
git remote -v
```

6. Commit, push and pull

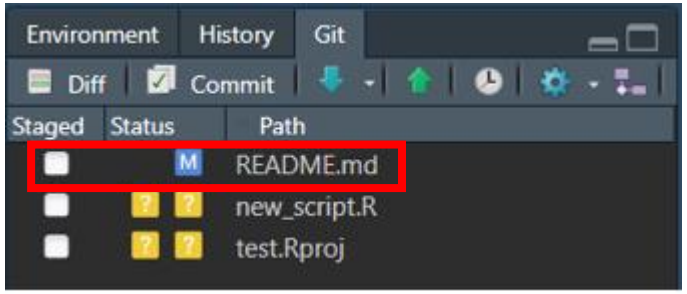
MAKE A COMMIT every time you finish a valuable chunk of work, probably many times a day

6. Commit, push and pull



- All files within the Project are 'observed' by Git, unless added to the .gitignore file
- If there are any changes in the Project – new files created (new_script.R), any SAVED changes made to the content of existing files (README.md) etc., Git will keep notifying you and will let you decide if to start tracking them (new files) or commit changes (modified files)

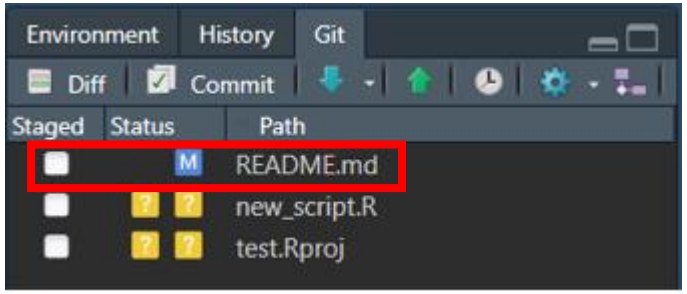
6. Stage, commit, push and pull



CREATING NEW FILES, MODYFYING EXISTING ONES

Git informes you that new files appeared in the Project (? ? = ‘unknown’ status)

6. Stage, commit, push and pull

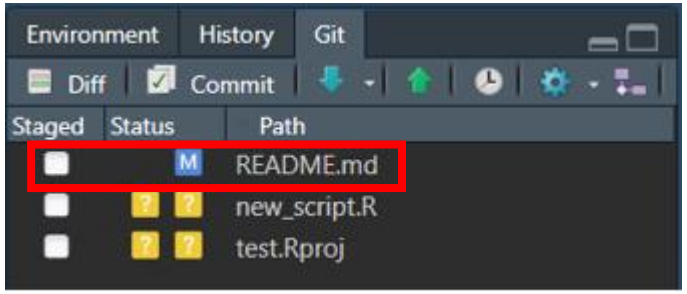


CREATING NEW FILES, MODYFYING EXISTING ONES

Git informes you that new files appeared in the Project (? ? = ‘unknown’ status)

Git informes you that changes were made to a file (M = ‘modified’ status)

6. Stage, commit, push and pull

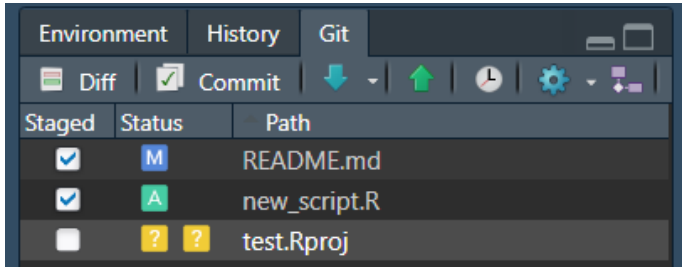


CREATING NEW FILES, MODYFYING EXISTING ONES

Git informes you that new files appeared in the Project (? ? = ‘unknown’ status)

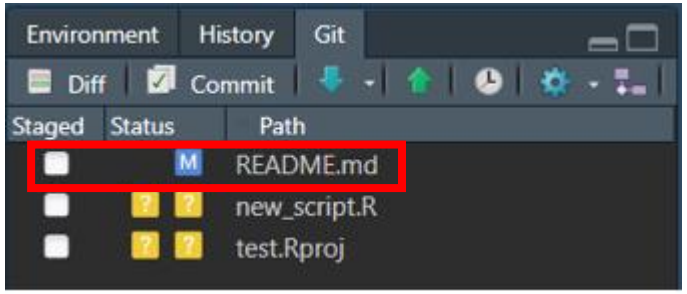
Git informes you that changes were made to a file (M = ‘modified’ status)

STAGING



- By ticking files you *stage* them (it is a necessary step before a commit) so you let Git know which files will be included in your next commit
- By staging a new file (new_script.R) you let Git know you want it to start tracking this file (A = ‘adding for tracking’ status)
- By staging a modified file (README.md) you let Git know you want to capture the file content modification

6. Stage, commit, push and pull

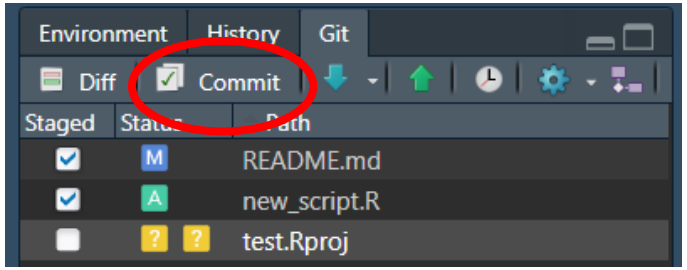


CREATING NEW FILES, MODYFYING EXISTING ONES

Git informes you that new files appeared in the Project (? ? = ‘unknown’ status)

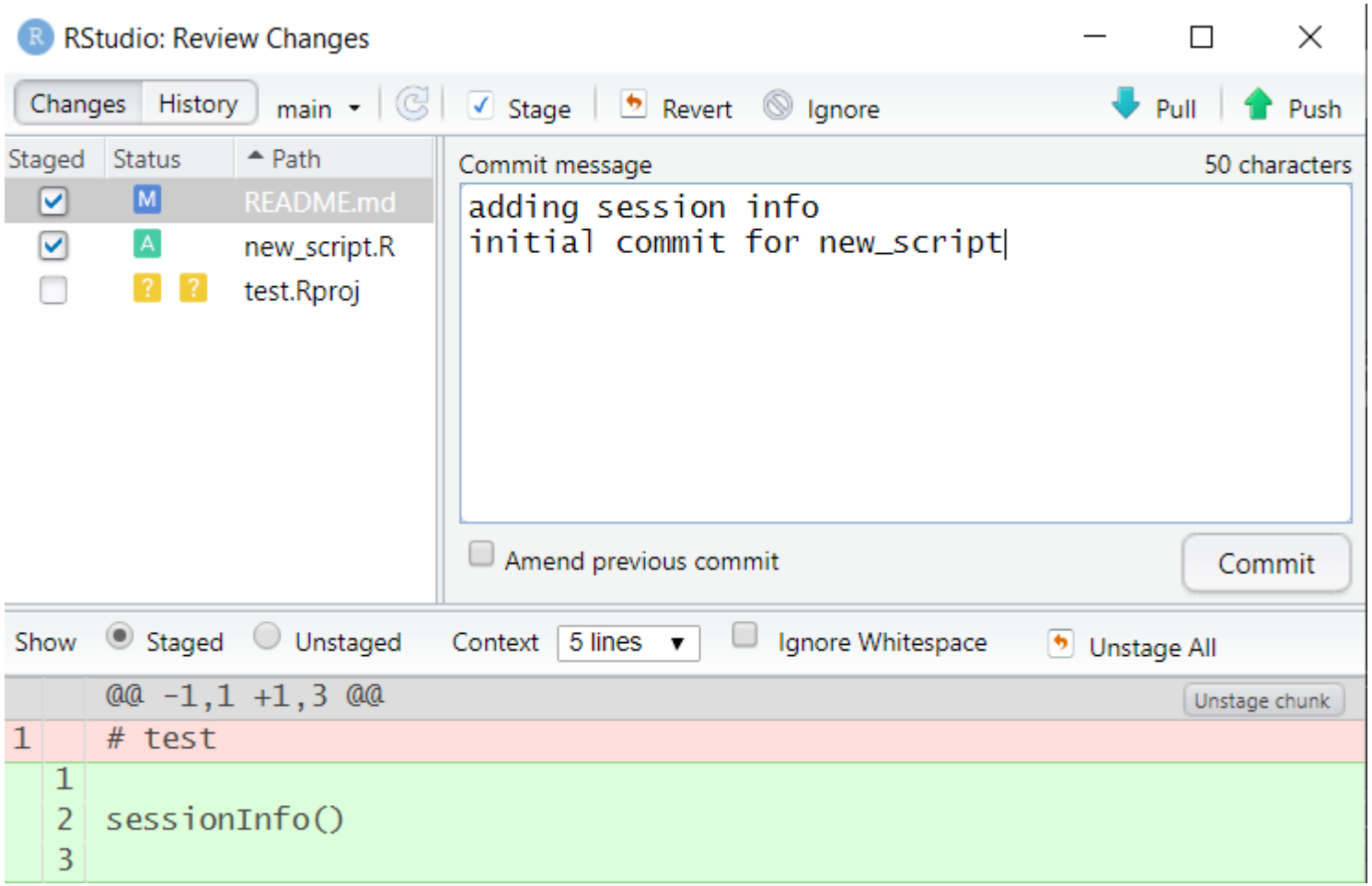
Git informes you that changes were made to a file (M = ‘modified’ status)

STAGING



- By ticking files you *stage* them (it is a necessary step before a commit) so you let Git know which files will be included in your next commit
- By staging a new file (new_script.R) you let Git know you want it to start tracking this file (A = ‘adding for tracking’ status)
- By staging a modified file (README.md) you let Git know you want to capture the file content modification
- After you staged all the files, click ‘Commit’ button

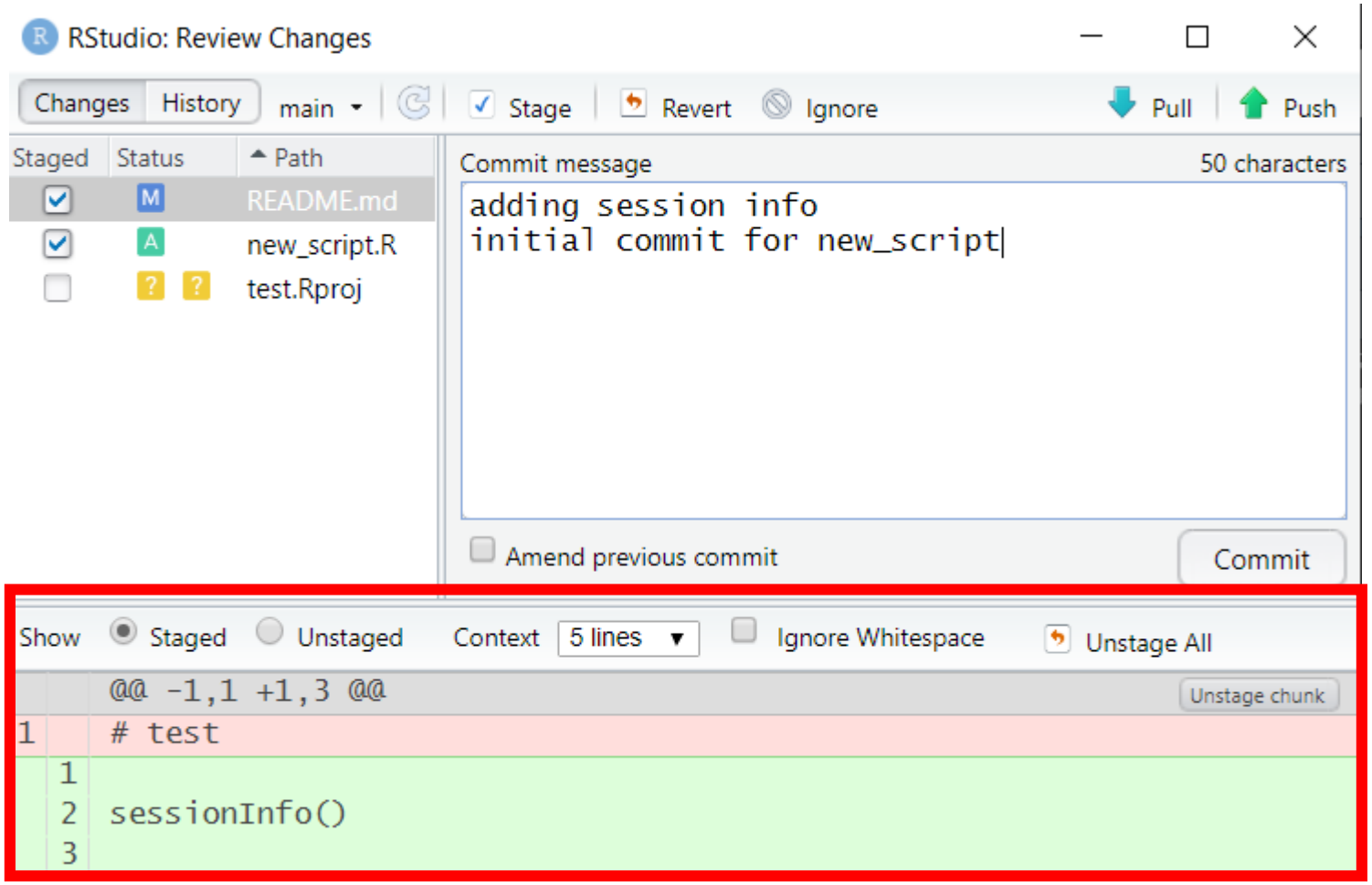
6. Stage, commit, push and pull



COMMIT

- A new window pops-out where you need to provide an **informative** description of the commit (Commit message) so in the future you can easily find the commit by name
- No need to include date in the commit, it is automatically dated

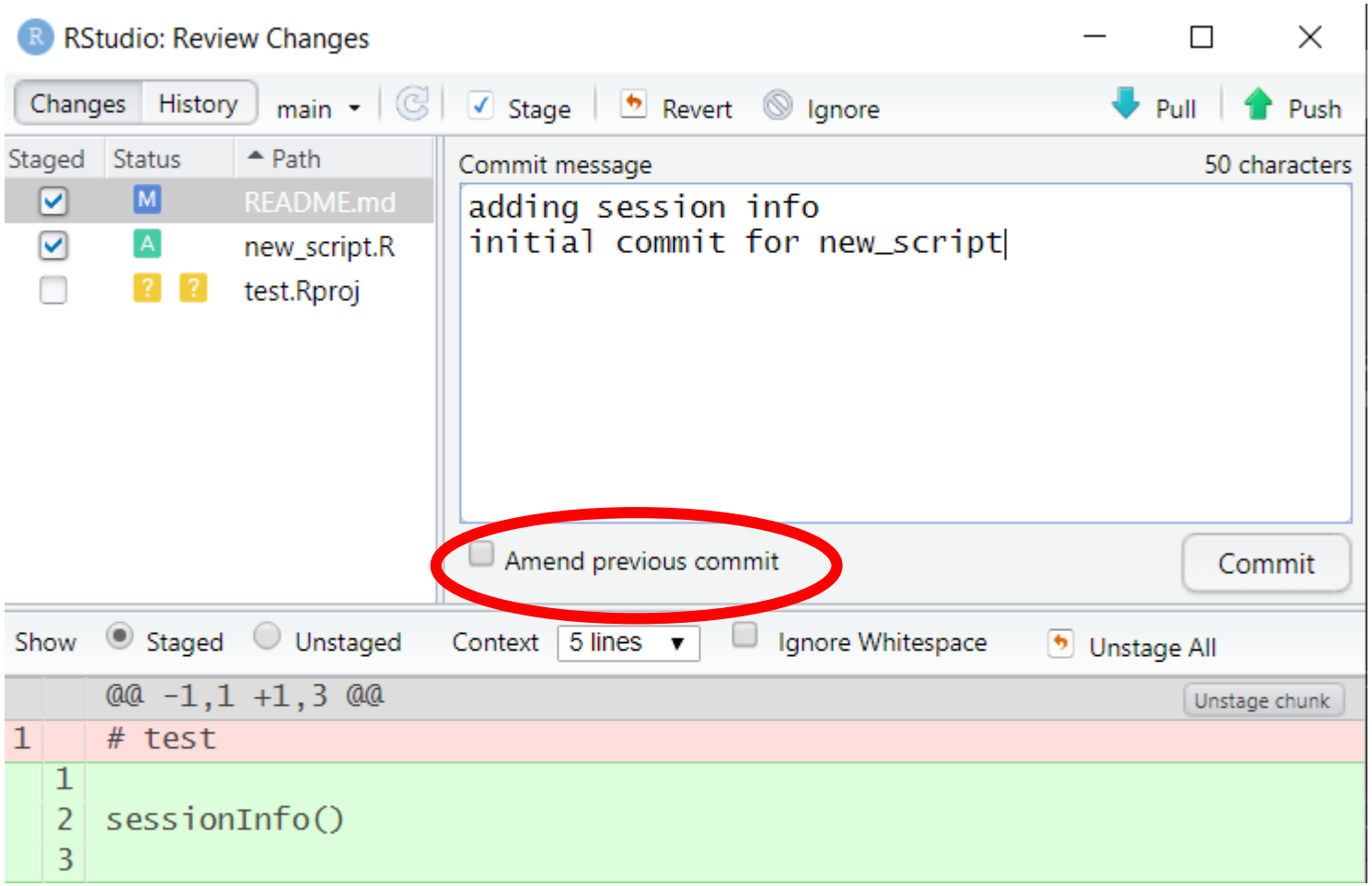
6. Stage, commit, push and pull



COMMIT

- A new window pops-out where you need to provide an **informative** description of the commit (Commit message) so in the future you can easily find the commit by name
- No need to include date in the commit, it is automatically dated
- Clicking on a staged file (README.md) we can see the changes ('diff') made in the file (deleted lines in red, modified/new lines in green)
- Clicking 'Commit' button captures an image of the code in a particular time point

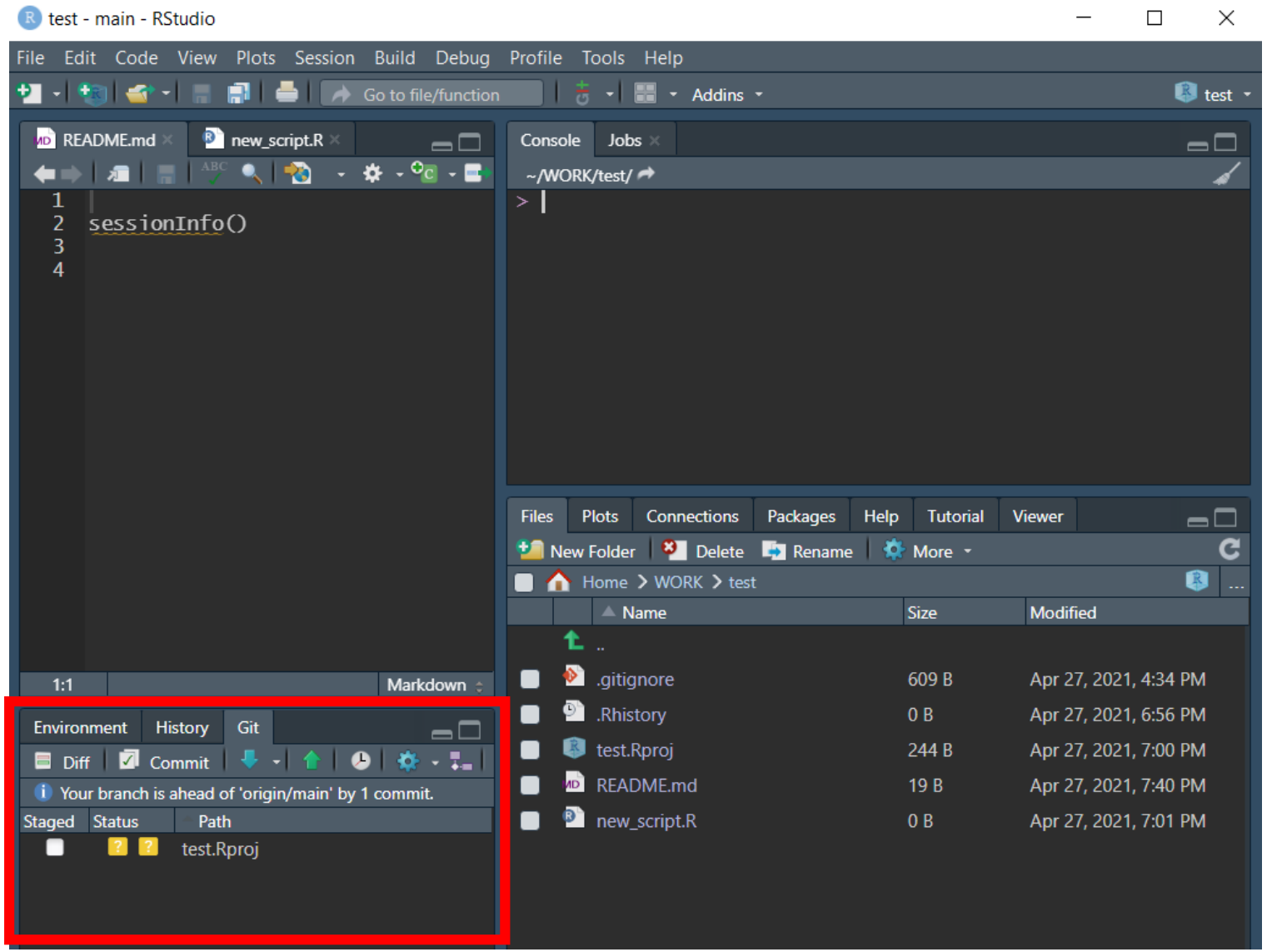
6. Stage, commit, push and pull



COMMIT

- A new window pops-out where you need to provide an **informative** description of the commit (Commit message) so in the future you can easily find the commit by name
- No need to include date in the commit, it is automatically dated
- Clicking on a staged file (README.md) we can see the changes ('diff') made in the file (deleted lines in red, modified/new lines in green)
- Clicking 'Commit' button captures an image of the code in a particular time point

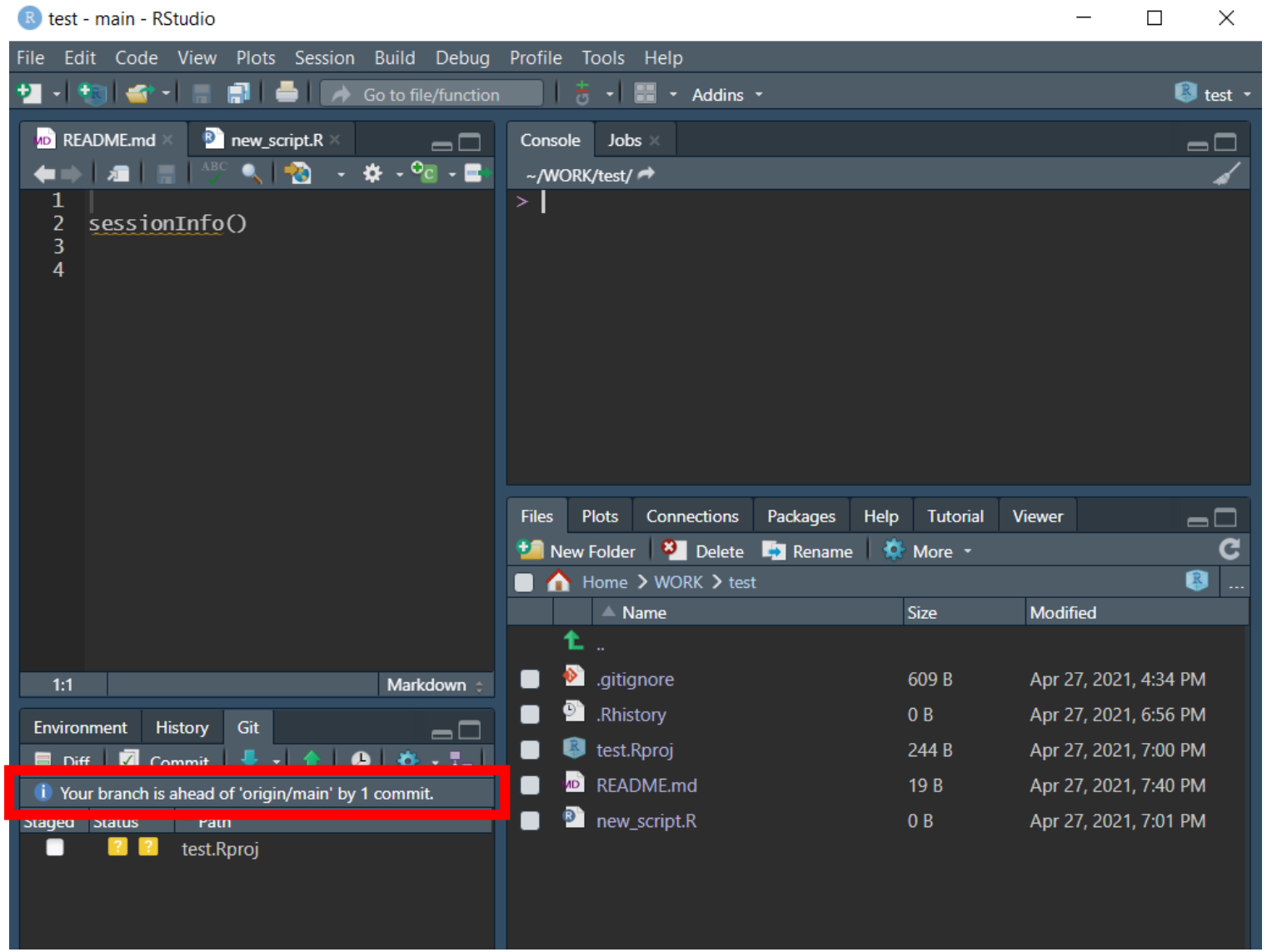
6. Stage, commit, push and pull



PUSH

- After the commit, the staged files disappear

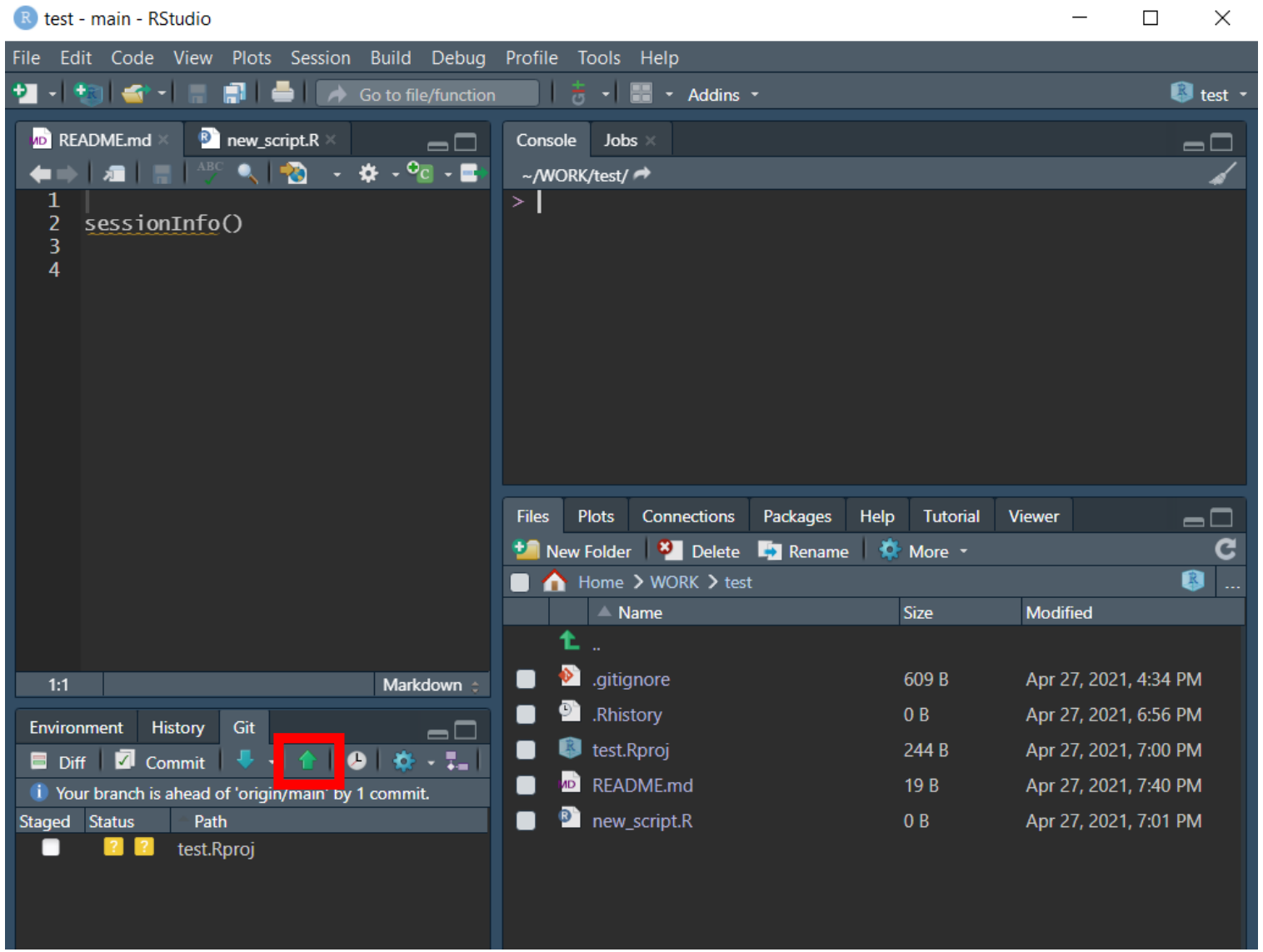
6. Stage, commit, push and pull



PUSH

- After the commit, the staged files disappear
- You are also informed that your local repository is ahead of the remote one by one commit (your local version is newer than the online one by one commit)
- You do not need to push each time you make a commit, you can do it e.g. at the end of the day

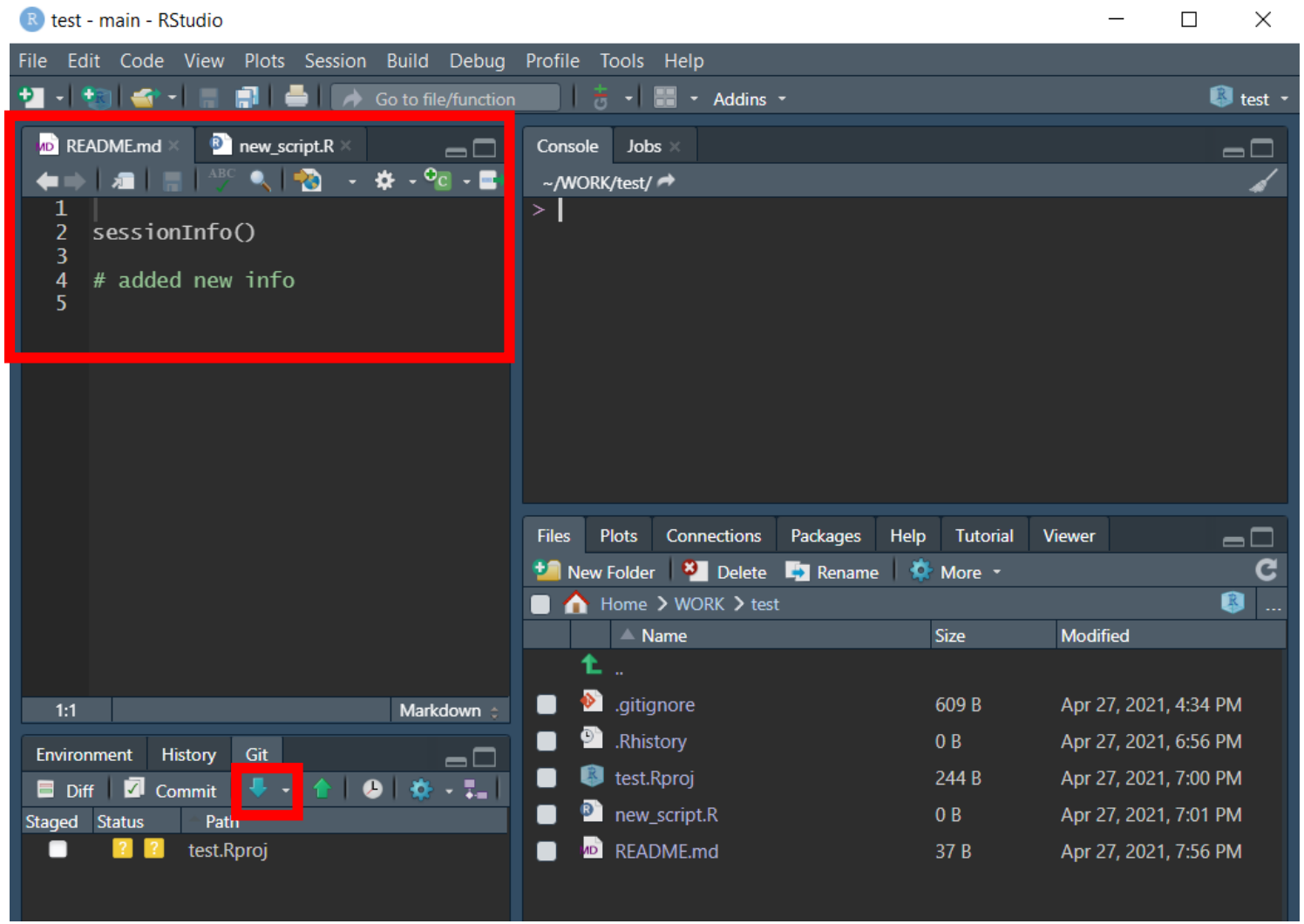
6. Stage, commit, push and pull



PUSH

- After the commit, the staged files disappear
- You are also informed that your local repository is ahead of the remote one by one commit (your local version is newer than the online one by one commit)
- You do not need to push each time you make a commit, you can do it e.g. at the end of the day
- To push your code to the online repository, click on the green ‘Push’ button

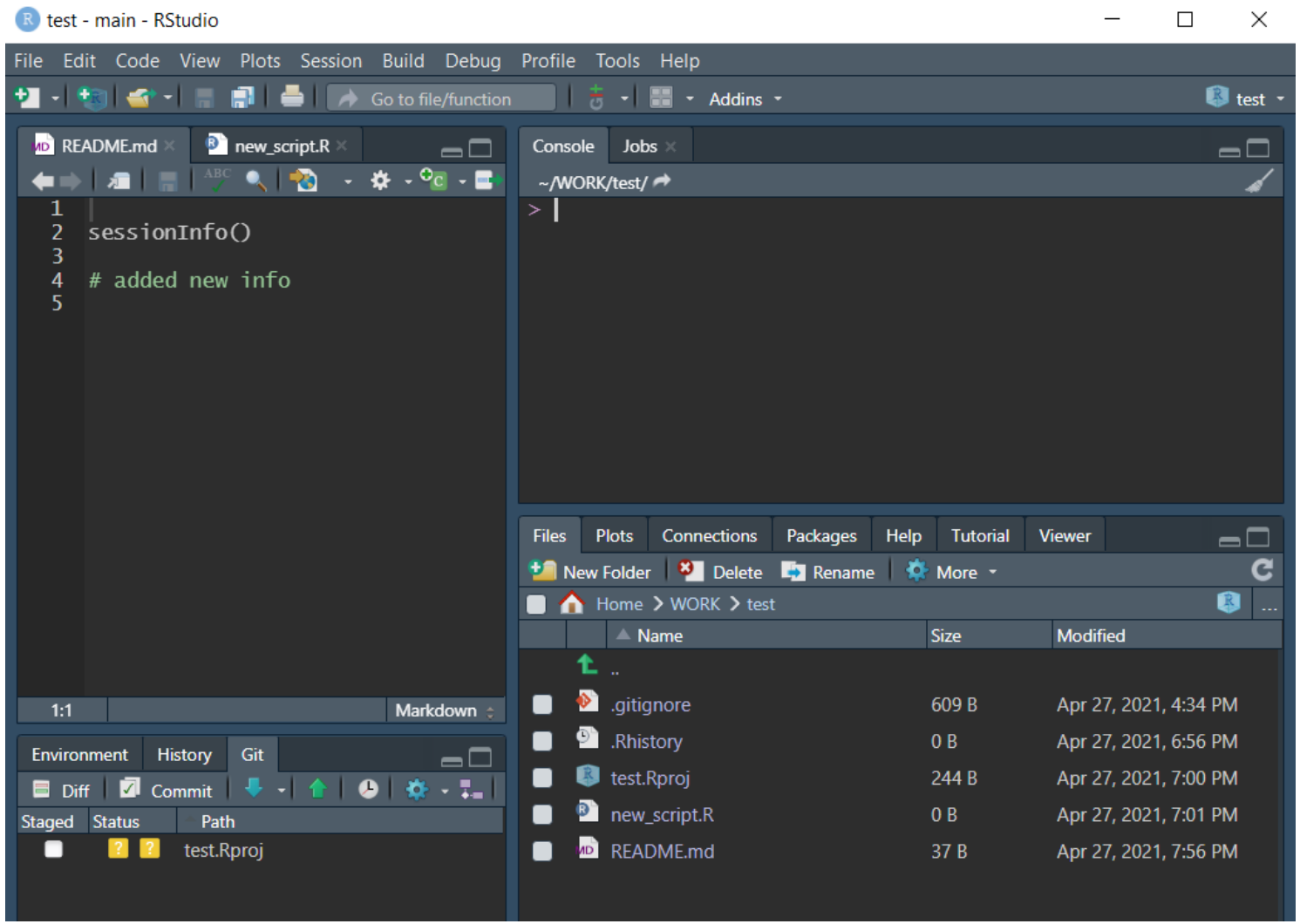
6. Stage, commit, push and pull



PULL

- If you want to e.g. synchronize your work between machines (e.g. your computer at work and another one at home; your machine and your colleague's machine who works with you on a project, etc.), on the machine where the work is outdated, click the blue 'Pull' button
- Pulling will 'download' new versions of your tracked files and replace/ merge with the old ones

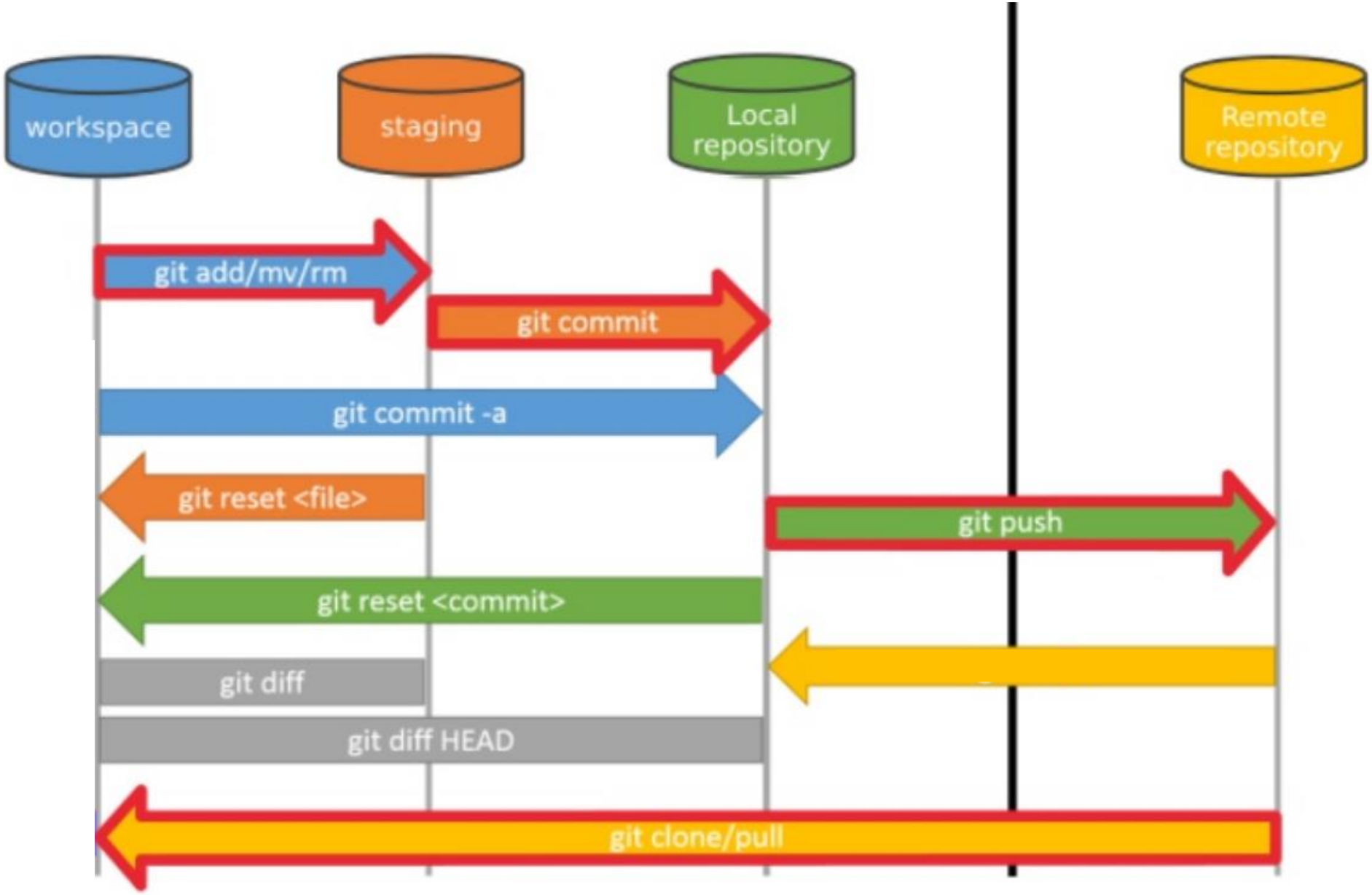
6. Stage, commit, push and pull



PULL

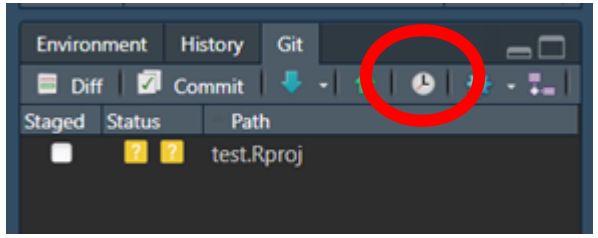
- If you want to e.g. synchronize your work between machines (e.g. your computer at work and another one at home; your machine and your colleague's machine who works with you on a project, etc.), on the machine where the work is outdated, click the blue 'Pull' button
- Pulling will 'download' new versions of your tracked files and replace/ merge with the old ones
- Do not worry (too much) about overwriting some important files – Git will inform you if there are any conflicts
- If you work on several machines and do Push and Pull frequently, get used to starting your work by making a Pull (you will avoid conflicts)

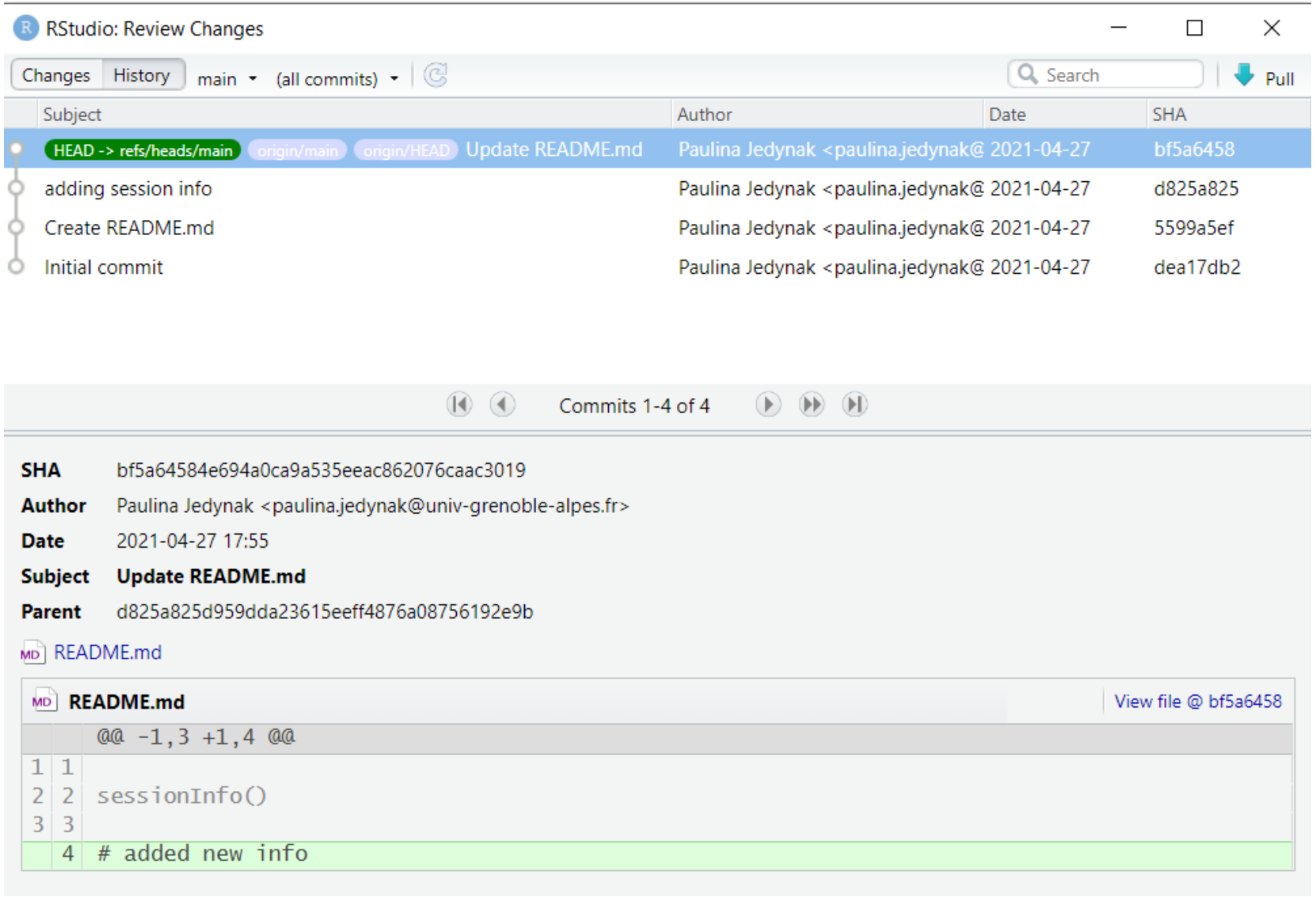
6. Stage, commit, push and pull



6. Stage, commit, push and pull

History





7. Reversing a commit (or peeking to the history)

7. Reversing* a commit (or peeking to the history)

*revert != reset != checkout… (the most confusing part of Git)

The need to revert the changes in the code in a particular way will be specific for the working environment (e.g., if you need to get the last working version in the matter of hours/minutes – common in big scale production setups; if you work alone or with several developers of the same code, etc.)

Undoing things: basic level git-scm.com/book/en/v2/Git-Basics-Undoing-Things

Undoing things: level pro git-scm.com/book/en/v2/Git-Tools-Reset-Demystified

7. Reversing a commit (or peeking to the history)

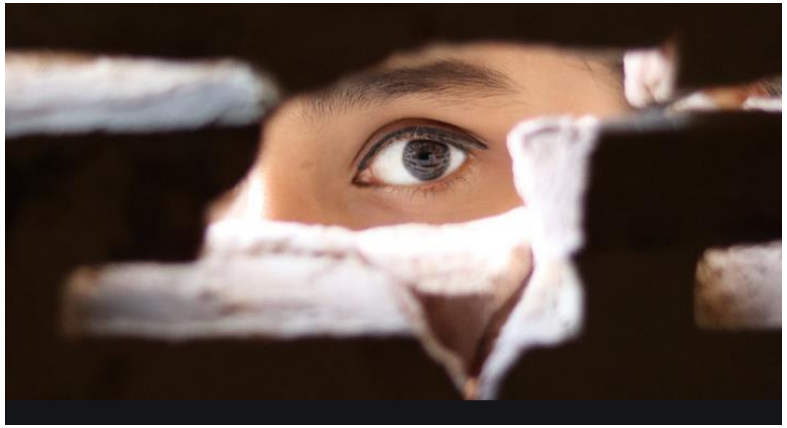
The most common scenarios when I go back in time:

1. **Reversing the code to its version in the particular point in the past**

   Happens very rarely (once per project and rather towards its end, when the code is close to its final version)

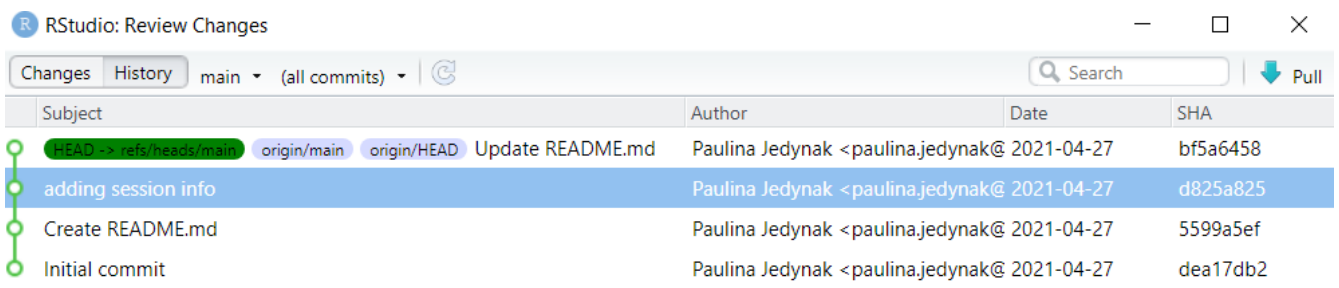
2. **Peeking in previous versions of the code to retrieve specific function or chunk of code**

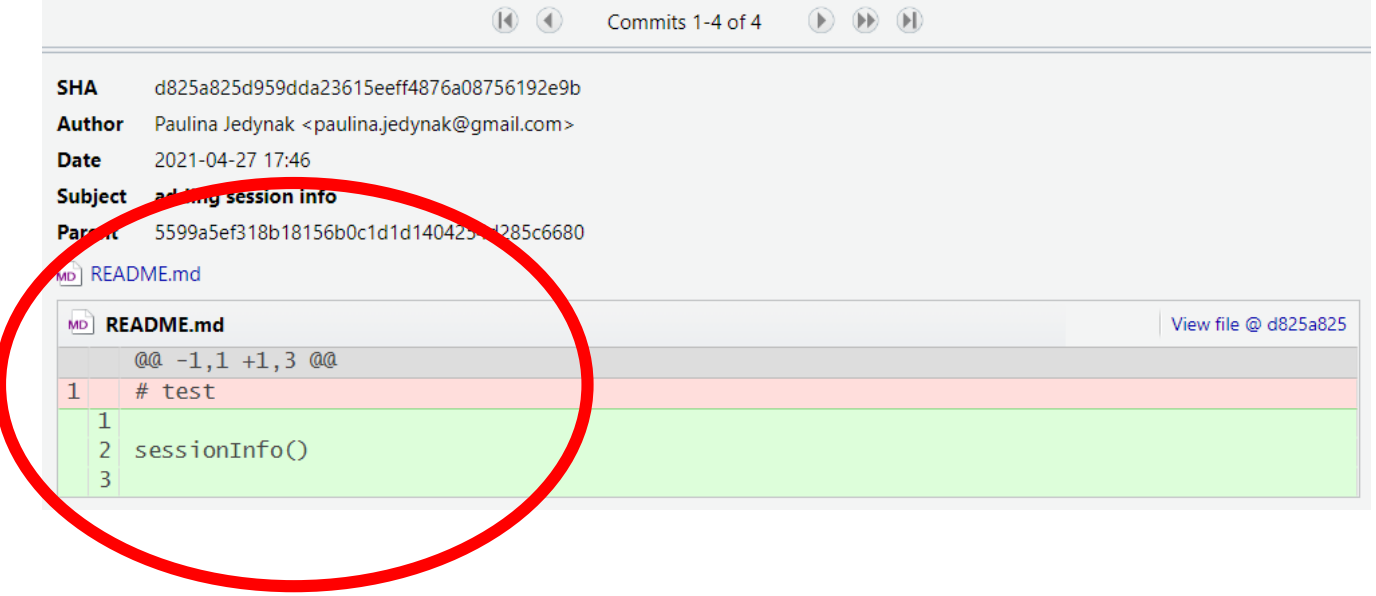
   Happens frequently (several times per project, at any development stage)



7. Reversing a commit (or peeking to the history)

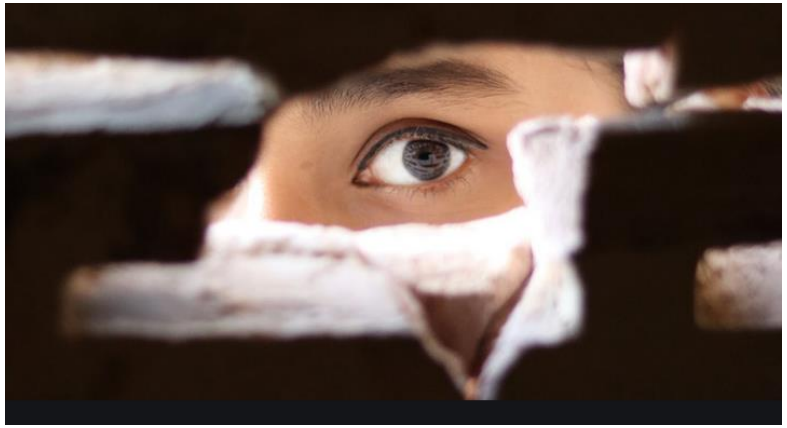
The most common scenarios when I go back in time:





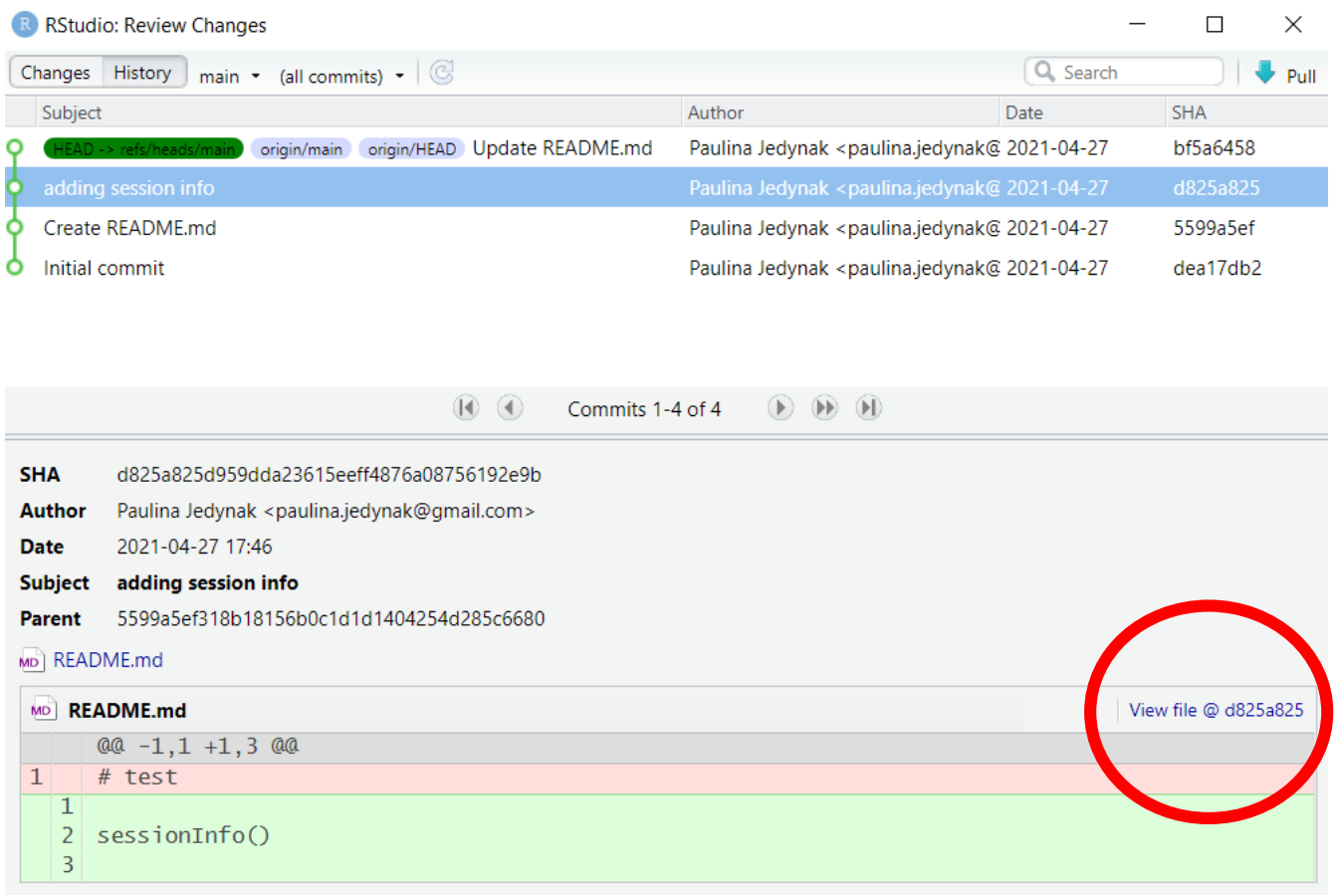
2. **Peeking in previous versions of the code to retrieve specific function or chunk of code**

   Happens frequently (several times per project, at any development stage) –> GoogleDocs version history



7. Reversing a commit (or peeking to the history)

The most common scenarios when I go back in time:



You may restore the old version of the script, in the same form as it was in the past (.R, .Rmd, .html…) and copy paste the chunk of code or save the entire script (but if you want to save the old file and use it to replace the never files – do not go there, reversing a commit is the tool you need!)

Summary

- Git is a great tool if you want to share work within projects (e.g. several persons committing to the same project – NO MORE SENDING CODE BY EMAIL!!!)
- Different collaborators (or your different machines) have the same version of the code plus they have access to all the changes made by any user (or by you in different time points)
- Ideally, when using Git, you should be consequent: make frequent commits that are named in an informative way and contain logically linked chunks of work. At the beginning it may not be easy, but it pays off
- Using Git allows you to reverse an error/ retrieve deleted files or chunks of code relatively easily (the more disciplined you are, the easier it becomes)
- The longer you work with Git, the less often you discover that all the files disappeared from your workspace and you have no idea where they went. At the end, in the 90% of cases they reappear ☺

Thank you for your attention

Paulina Jedynak, PhD
github.com/paujedynak
paulina.jedynak@gmail.com

Environmental Epidemiology lab
gricad-gitlab.univ-grenoble-alpes.fr