

# Using Git within RStudio



Paulina Jedynak, PhD  
[github.com/paujedynak](https://github.com/paujedynak)

Environmental Epidemiology lab  
[gricad-gitlab.univ-grenoble-alpes.fr](https://gricad-gitlab.univ-grenoble-alpes.fr)

# 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](http://happygitwithr.com)

If you prefer less details but straight to the poing, try [cfss.uchicago.edu/setup/git-with-rstudio](http://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](https://github.com)
- [gitlab.com](https://gitlab.com)
- [gricad-gitlab.univ-grenoble-alpes.fr](https://gricad-gitlab.univ-grenoble-alpes.fr) (login with AGALAN credentials)
- ...

NOTE: Git != GitHub





Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



[Overview](#)

[Repositories](#) 26

[Projects](#)

[Packages](#)



## 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)

# Paulina Jedynak

paujedydnak

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

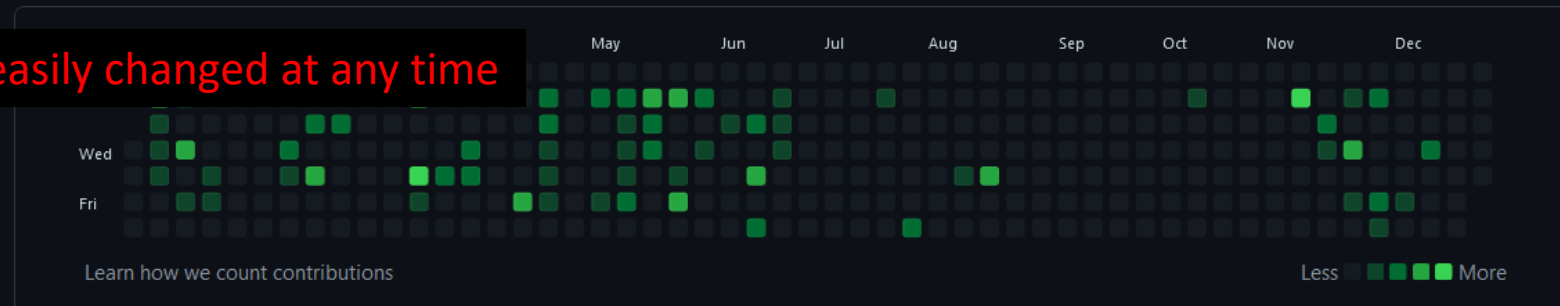
📍 Grenoble

✉ [paulina.jedynak@gmail.com](mailto:paulina.jedynak@gmail.com)

🔗 <https://www.linkedin.com/in/pau-jedyn...>

348 contributions in 2020

[Contribution settings](#) ▼



## Contribution activity

December 2020

Created 22 commits in 1 repository  
[paujedydnak/Jedynak2021EHP](#) 22 commits

2021

2020

2019

2018

7

# 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](https://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](https://git-scm.com/downloads)*

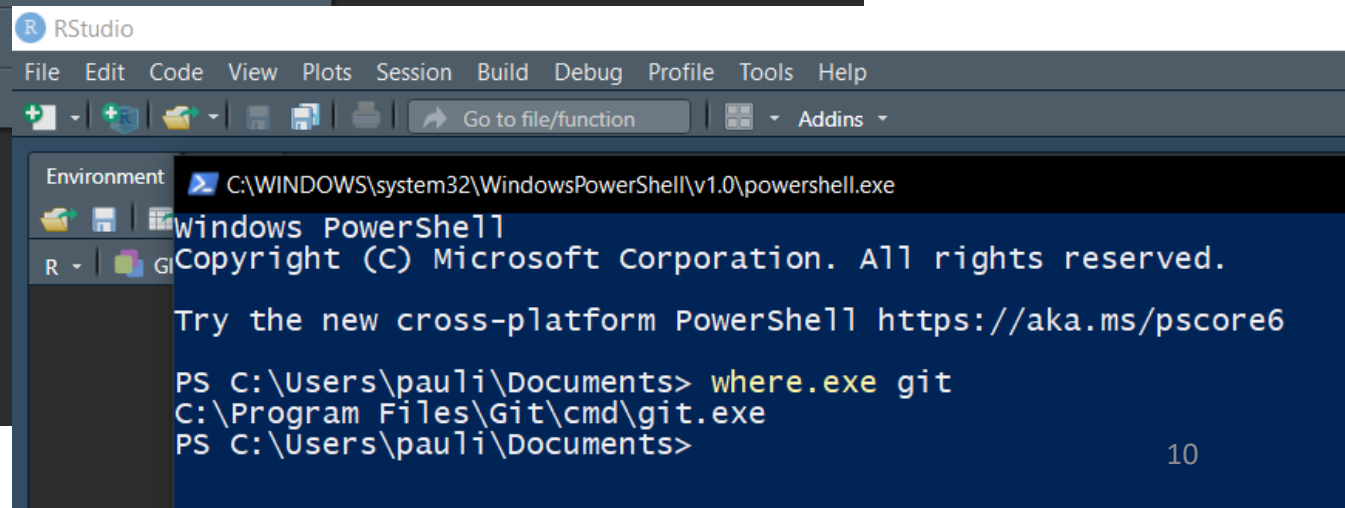
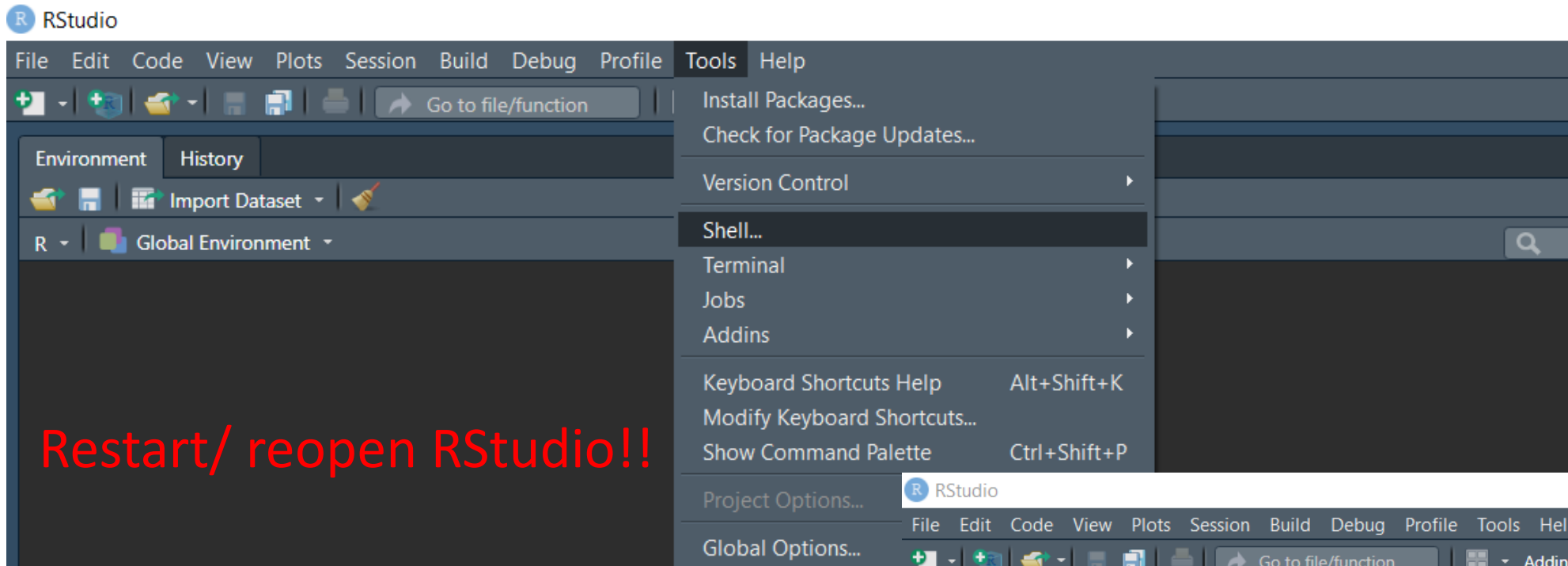
You can check if you already have Git by typing `where.exe git` (Windows)

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

4. Open RStudio

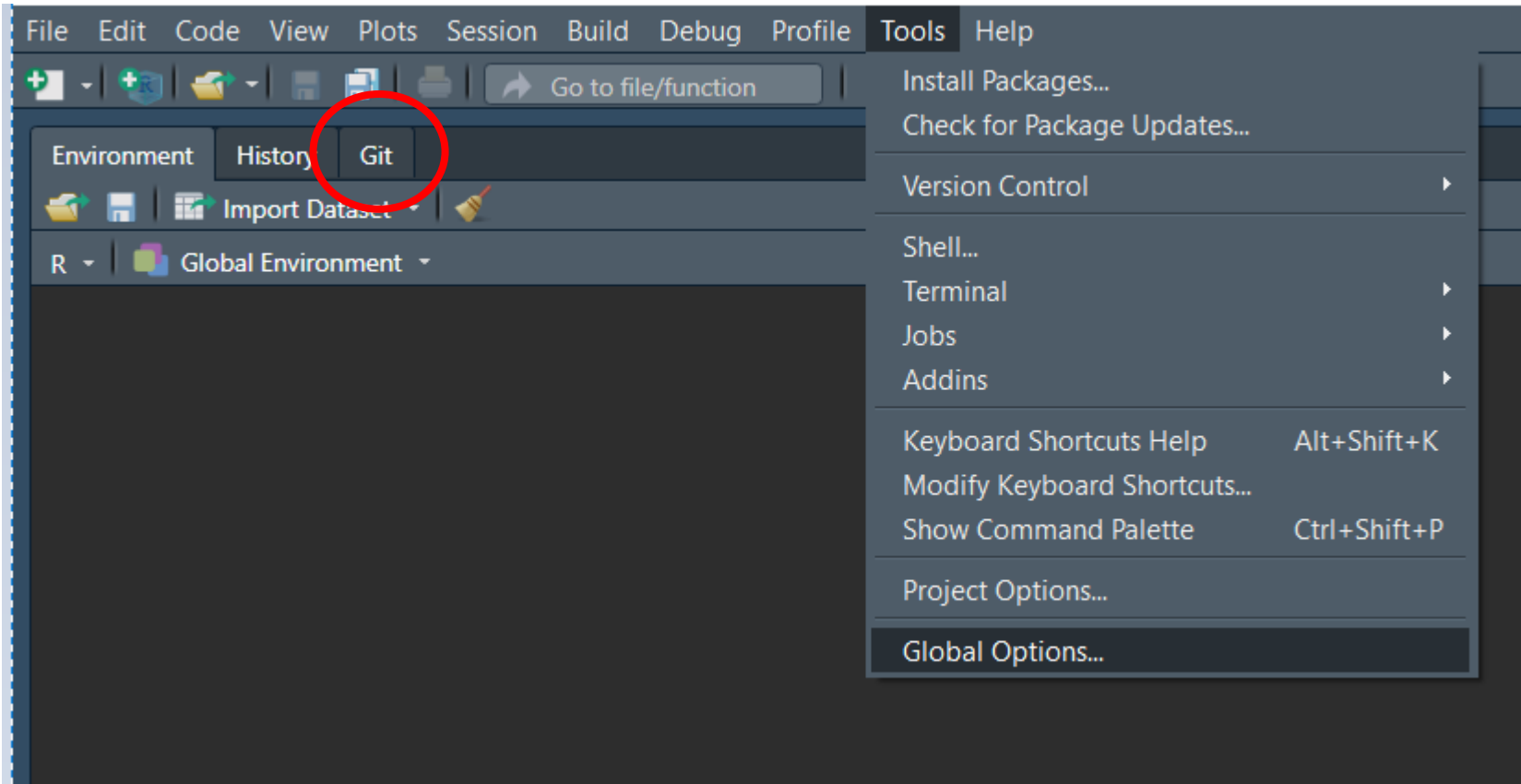


# 1. Initial installations



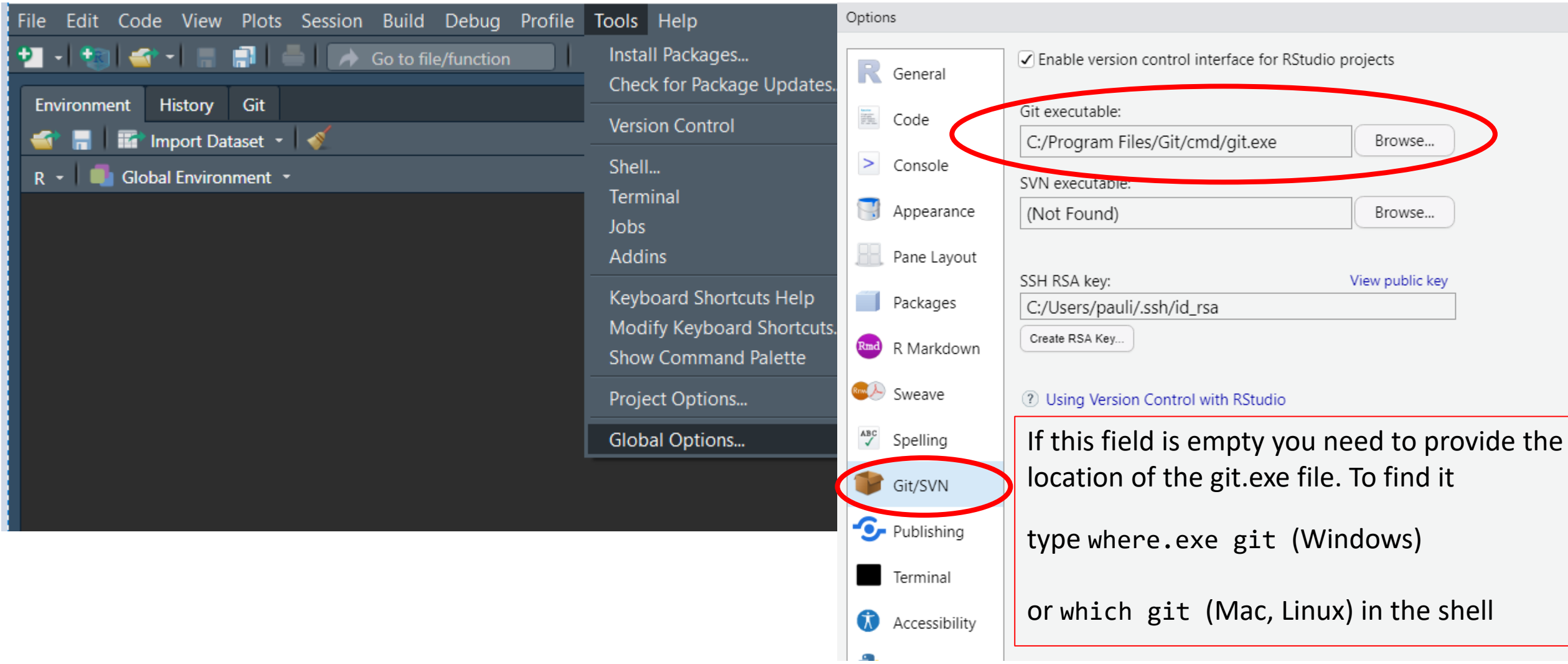
# 1. Initial installations

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)



The screenshot shows the RStudio interface with the 'Tools' menu open and the 'Git/SVN' option selected. The 'Options' dialog box is open, showing the 'Git/SVN' tab. The 'Git executable:' field is highlighted with a red circle and contains the path 'C:/Program Files/Git/cmd/git.exe'. The 'SVN executable:' field is empty and contains the text '(Not Found)'. The 'SSH RSA key:' field contains the path 'C:/Users/pauli/.ssh/id\_rsa'. A red box highlights a text area at the bottom right of the dialog.

Options

☒ Enable version control interface for RStudio projects

Git executable: C:/Program Files/Git/cmd/git.exe Browse...

SVN executable: (Not Found) Browse...

SSH RSA key: C:/Users/pauli/.ssh/id\_rsa View public key

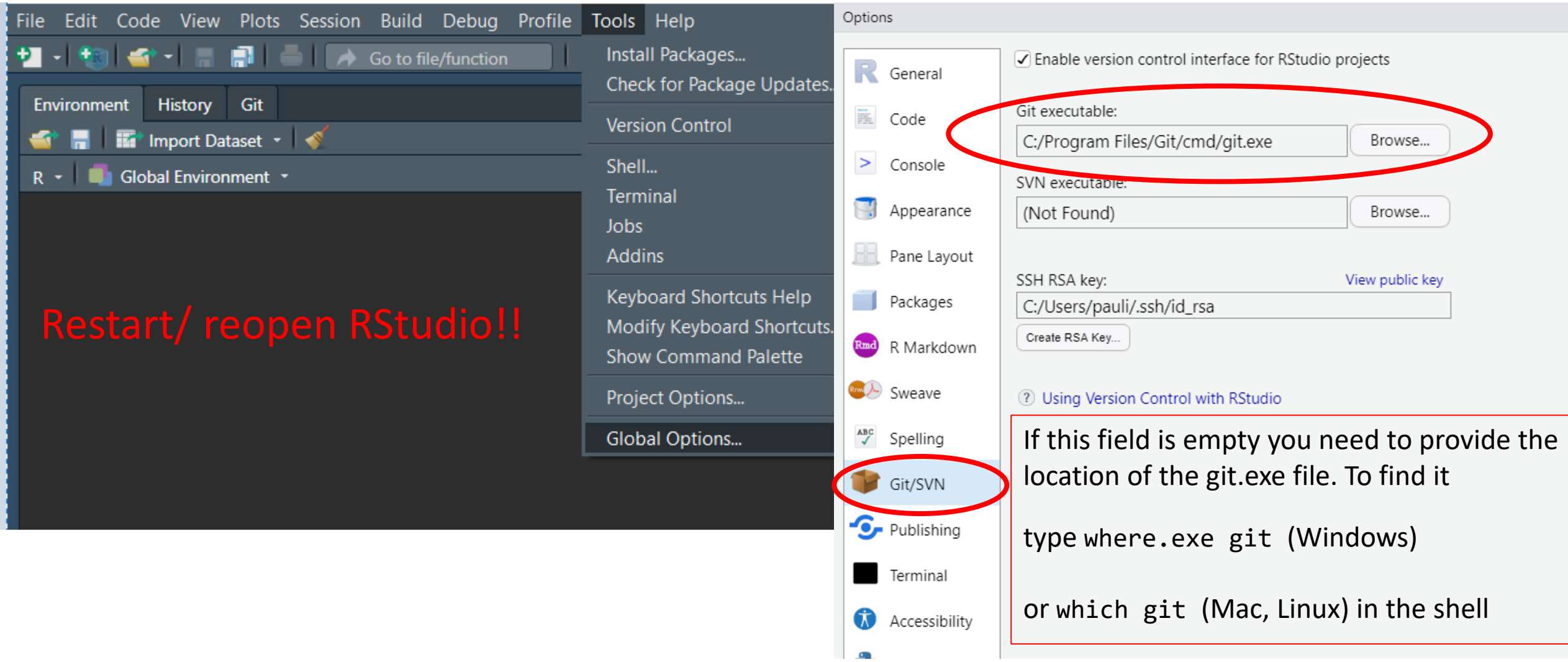
Create RSA Key...

? Using Version Control with 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

# 1. Initial installations

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



The screenshot shows the RStudio interface with the **Tools** menu open and the **Git/SVN** option selected. The **Options** dialog box is displayed on the right, with the **Git executable:** field highlighted by a red oval. The field contains the path `C:/Program Files/Git/cmd/git.exe`. A red box at the bottom right contains instructions for filling this field if it is empty.

**Restart/ reopen RStudio!!**

**Options**

- ☒ Enable version control interface for RStudio projects
- Git executable:**
- SVN executable:**
- SSH RSA key:**  [View public key](#)
- 
- [? Using Version Control with 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")
```

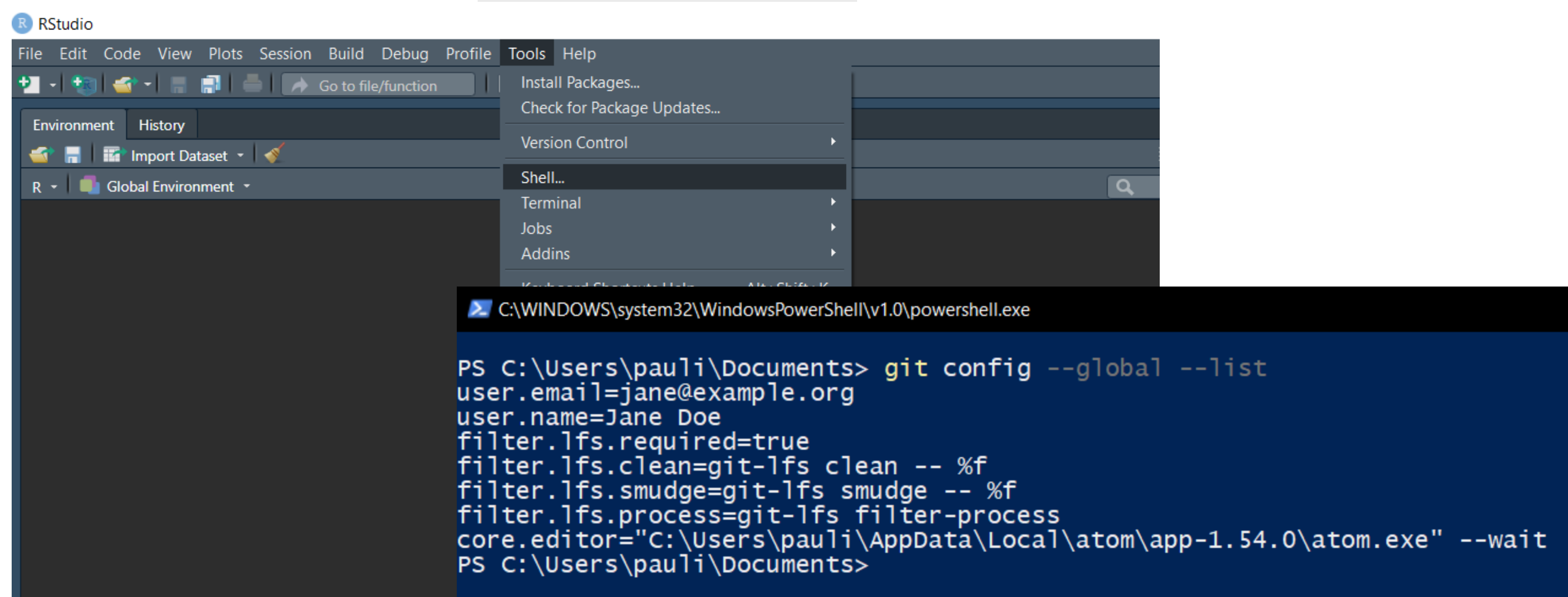
This does not have to be your GitHub user name

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

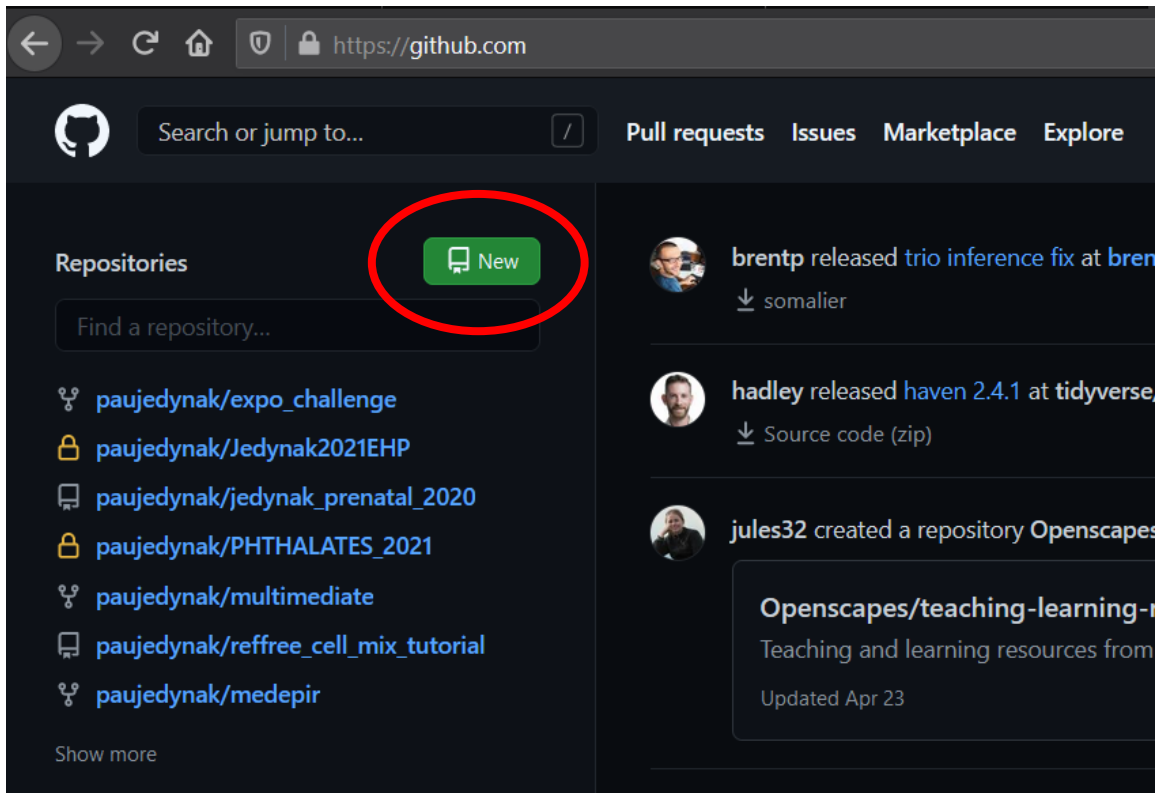
```
git config --global -list (again in the shell)
```





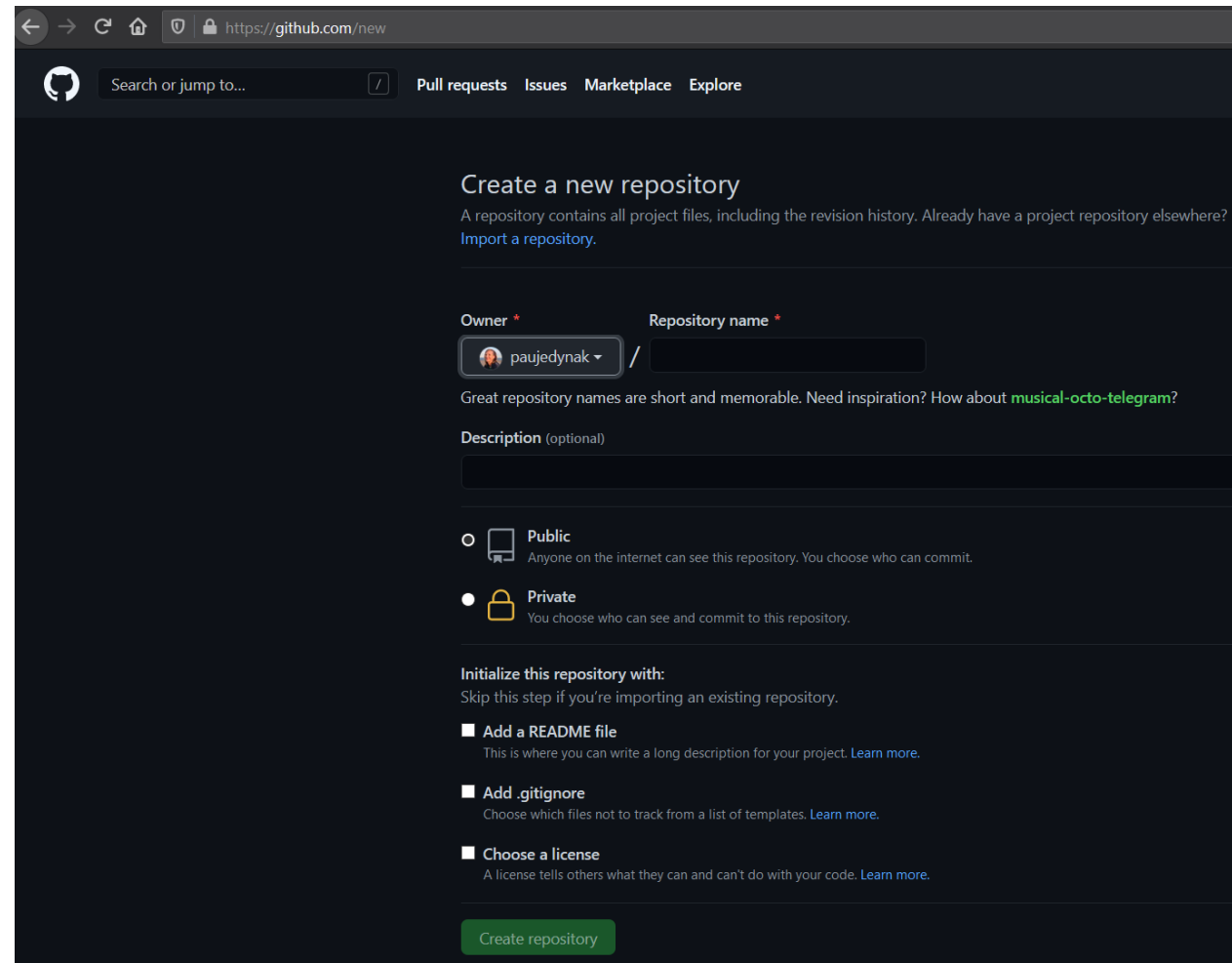
# 3. Connect to GitHub

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



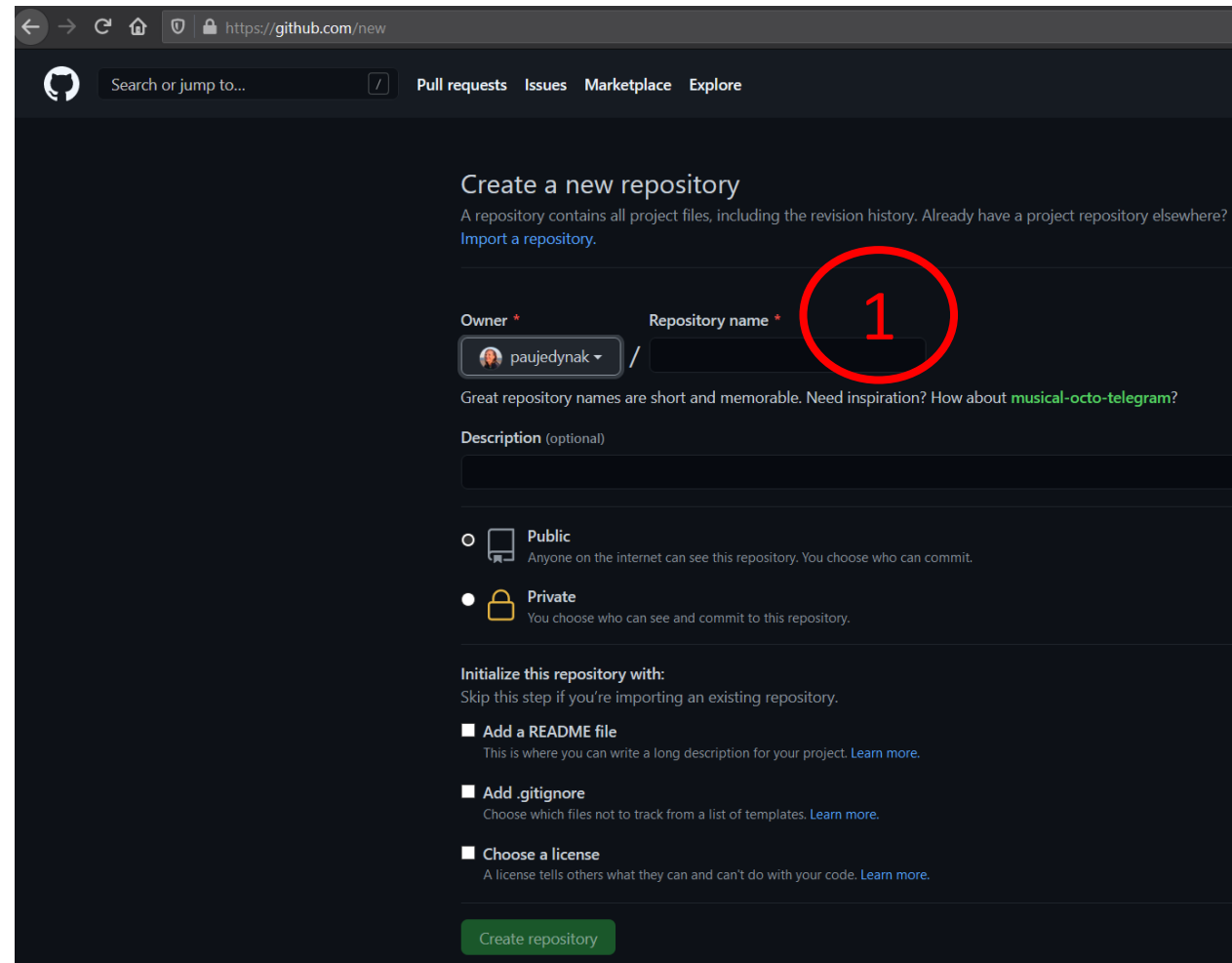
The screenshot shows the GitHub 'Create a new repository' page. The browser address bar displays 'https://github.com/new'. The page header includes the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main content area is titled 'Create a new repository' and includes a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'paujedynak') and 'Repository name'. A suggestion for repository names is provided: 'Great repository names are short and memorable. Need inspiration? How about [musical-octo-telegram](#)?'. There is an optional 'Description' text area. The 'Visibility' section has two radio buttons: 'Public' (selected) and 'Private'. The 'Initialize this repository with:' section has three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. Each checkbox has a brief description and a 'Learn more' link. At the bottom, there is a green 'Create repository' 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



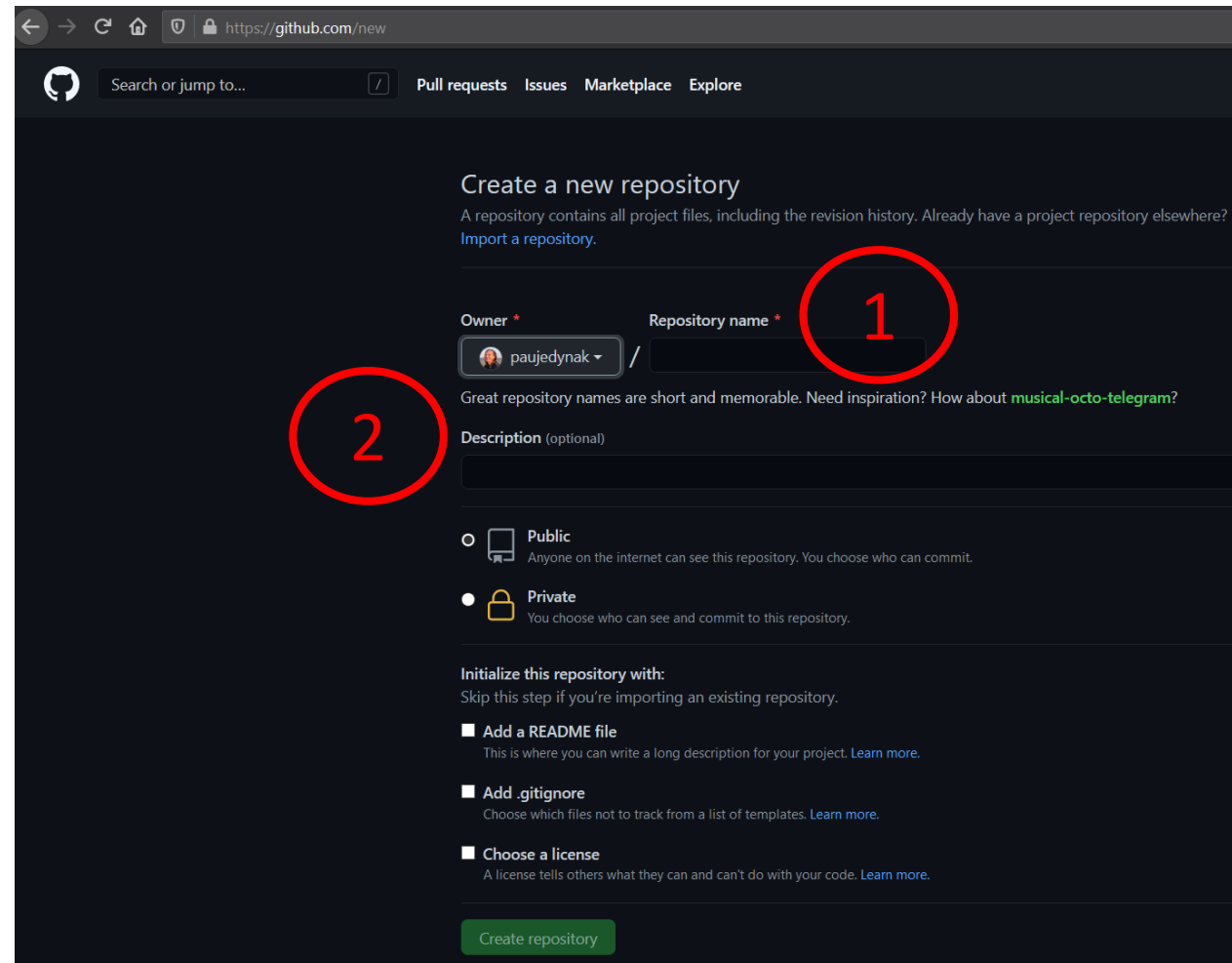
The screenshot shows the GitHub 'Create a new repository' page. The 'Repository name' field is highlighted with a red circle and a red number '1'. The 'Owner' field is set to 'paujedynak'. The 'Description' field is optional. The 'Public' option is selected for the repository's visibility. The 'Initialize this repository with' section includes options to 'Add a README file', 'Add .gitignore', and 'Choose a license'. A green 'Create repository' button is at the bottom.

# 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



The screenshot shows the GitHub 'Create a new repository' page. The browser address bar shows 'https://github.com/new'. The page has a dark theme. At the top, there's a search bar and navigation links: 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository' with a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'paujedynak') and 'Repository name' (with a red circle and the number '1' next to it). Below these fields, there's a text input for 'Description (optional)' with a red circle and the number '2' next to it. Under the description field, there are two radio button options: 'Public' (selected) and 'Private'. At the bottom, there's a section 'Initialize this repository with:' with three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. A green 'Create repository' button is at the bottom right.

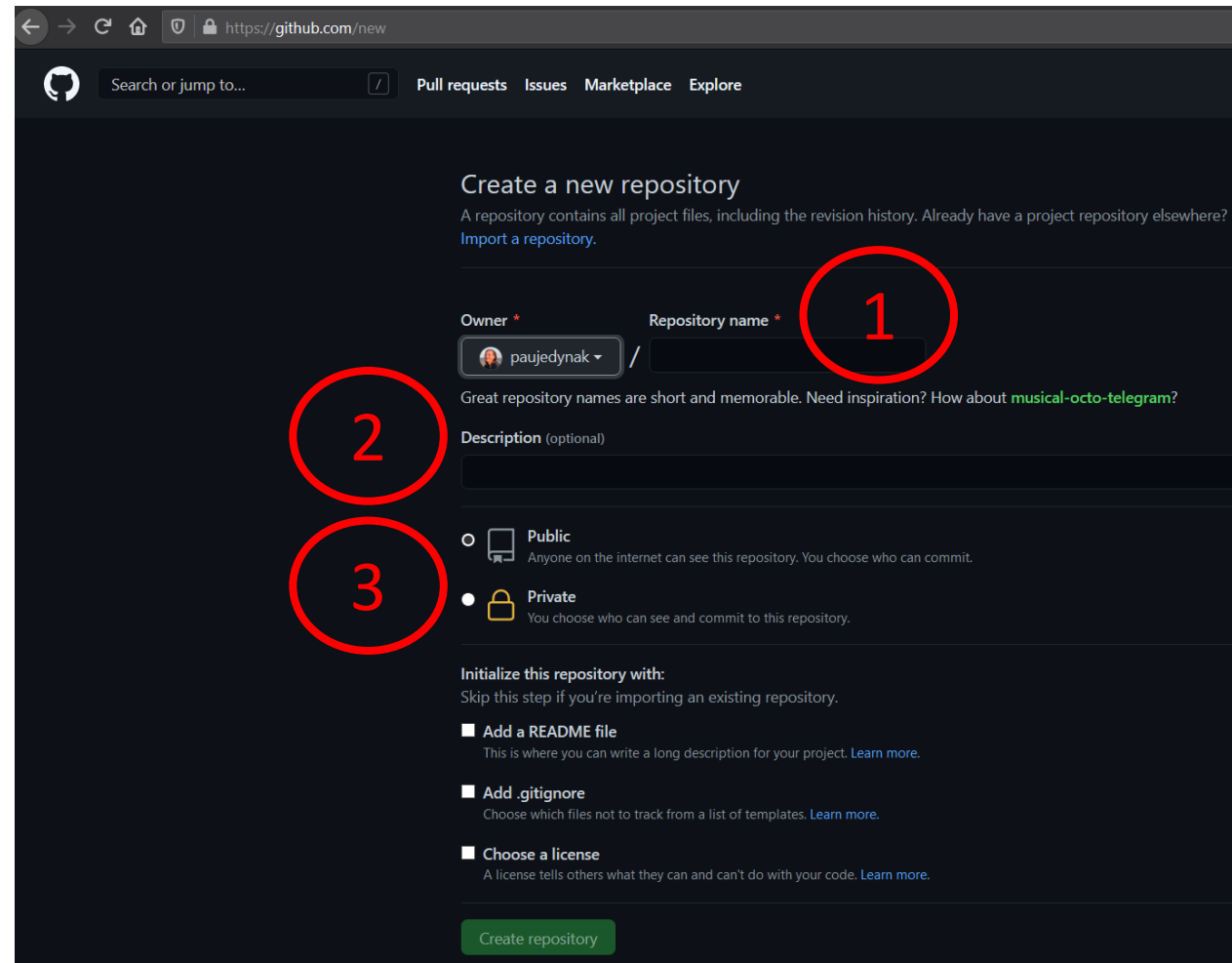
# 3. Connect to GitHub

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



The screenshot shows the GitHub 'Create a new repository' page. The page has a dark theme. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository'. Below it, there's a subheading 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. The form fields are: 'Owner' (a dropdown menu showing 'paujedynak'), 'Repository name' (a text input field with a red circle and the number '1' next to it), 'Description (optional)' (a text input field with a red circle and the number '2' next to it), and 'Visibility' (two radio buttons: 'Public' and 'Private', with 'Public' selected and a red circle and the number '3' next to it). Below the visibility options, there's a section 'Initialize this repository with:' with three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. At the bottom, there's a green 'Create repository' 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
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

The screenshot shows the GitHub 'Create a new repository' page. The page is dark-themed. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository'. Below it, there's a subheading 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. The form has several fields: 'Owner' (a dropdown menu showing 'paujedynak'), 'Repository name' (a text input field with a red circle and the number '1' next to it), 'Description (optional)' (a text input field with a red circle and the number '2' next to it), and 'Visibility' (radio buttons for 'Public' and 'Private', with 'Public' selected and a red circle and the number '3' next to it). Below these, there's a section 'Initialize this repository with:' with three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. The 'Add a README file' checkbox is checked and has a red circle and the number '4' next to it. At the bottom, there's a green 'Create repository' 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
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

The screenshot shows the GitHub 'Create a new repository' page. The interface is dark-themed. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository', followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'paujedynak') and 'Repository name' (with a red circle and the number '1' next to it). A hint below these fields says: 'Great repository names are short and memorable. Need inspiration? How about [musical-octo-telegram?](#)'. There's a 'Description (optional)' text area with a red circle and the number '2' next to it. Below the description, there are two radio button options for visibility: 'Public' (selected) and 'Private'. A red circle with the number '3' is next to the 'Public' option. Below the visibility options, there's a section 'Initialize this repository with:' with the instruction 'Skip this step if you're importing an existing repository.' There are three checkboxes: 'Add a README file' (selected), 'Add .gitignore' (selected), and 'Choose a license' (selected). A red circle with the number '4' is next to the 'Add .gitignore' checkbox. At the bottom, there's a green 'Create repository' button with a red circle and the number '5' next to it.

# 3. Connect to GitHub

The screenshot shows a web browser window with the URL `https://github.com/paujedynak/test`. The GitHub interface is in dark mode. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. Below this, the repository name 'paujedynak / test' is shown with a 'Private' label. A secondary navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The 'Code' tab is active. The main content area shows the 'main' branch with 1 branch and 0 tags. There are buttons for 'Go to file', 'Add file', and a green 'Code' button. Below this, a commit by 'paujedynak' is shown with the message 'Initial commit'. A file named '.gitignore' is listed under this commit. At the bottom, there's a prompt to 'Add a README with an overview of your project.' and a green 'Add a README' 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 ▾**

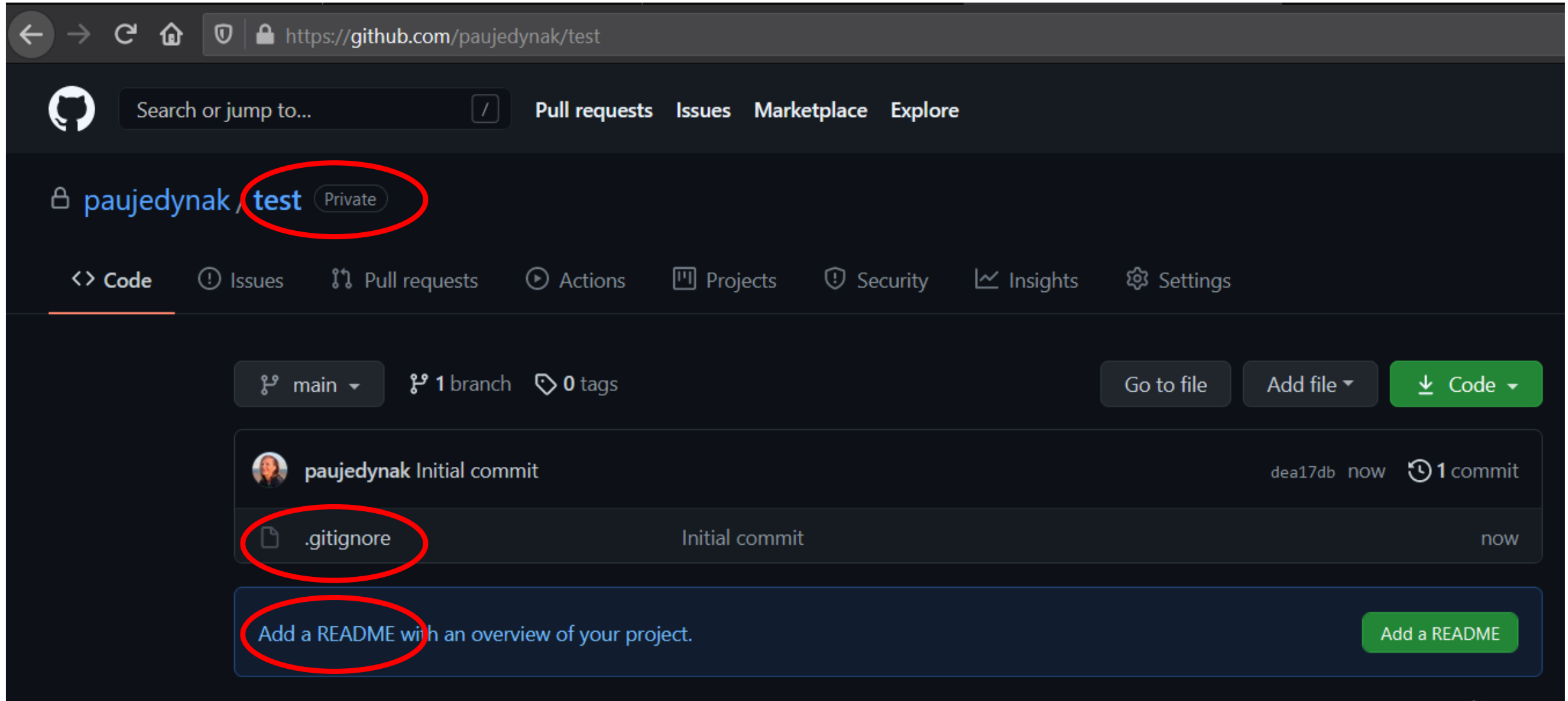
**paujedynak** Initial commit dea17db now ⌚ 1 commit

📄 .gitignore Initial commit now

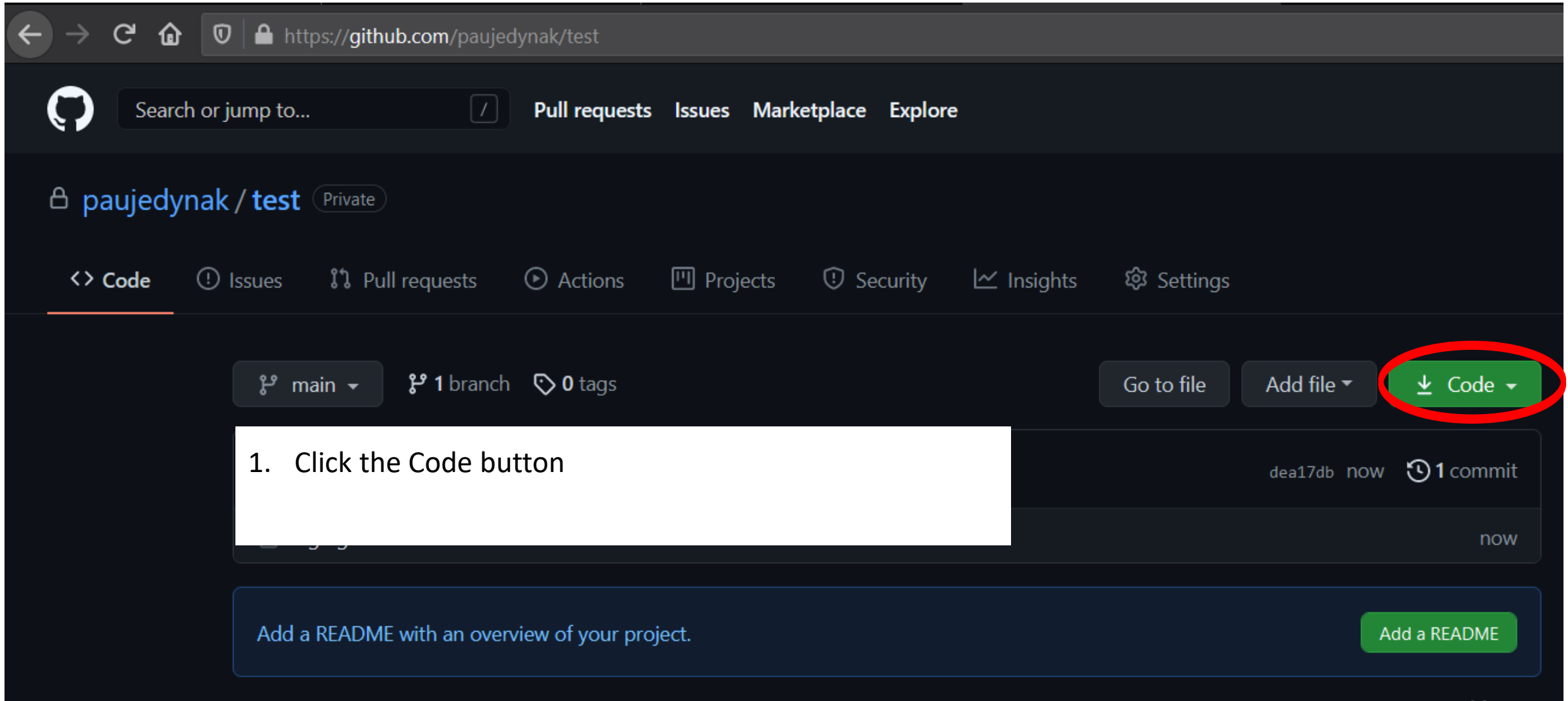
Add a README with an overview of your project. **Add a README**



# 3. Connect to GitHub



# 3. Connect to GitHub



The screenshot shows the GitHub interface for a repository named 'paujedynek/test'. The repository is marked as 'Private'. The 'Code' button is highlighted with a red circle. A white text box with the instruction '1. Click the Code button' is overlaid on the page. The repository has 1 branch and 0 tags. The commit history shows a single commit by 'dea17db' at 'now'.

1. Click the Code button

🔗 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

🔗 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

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

📄 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

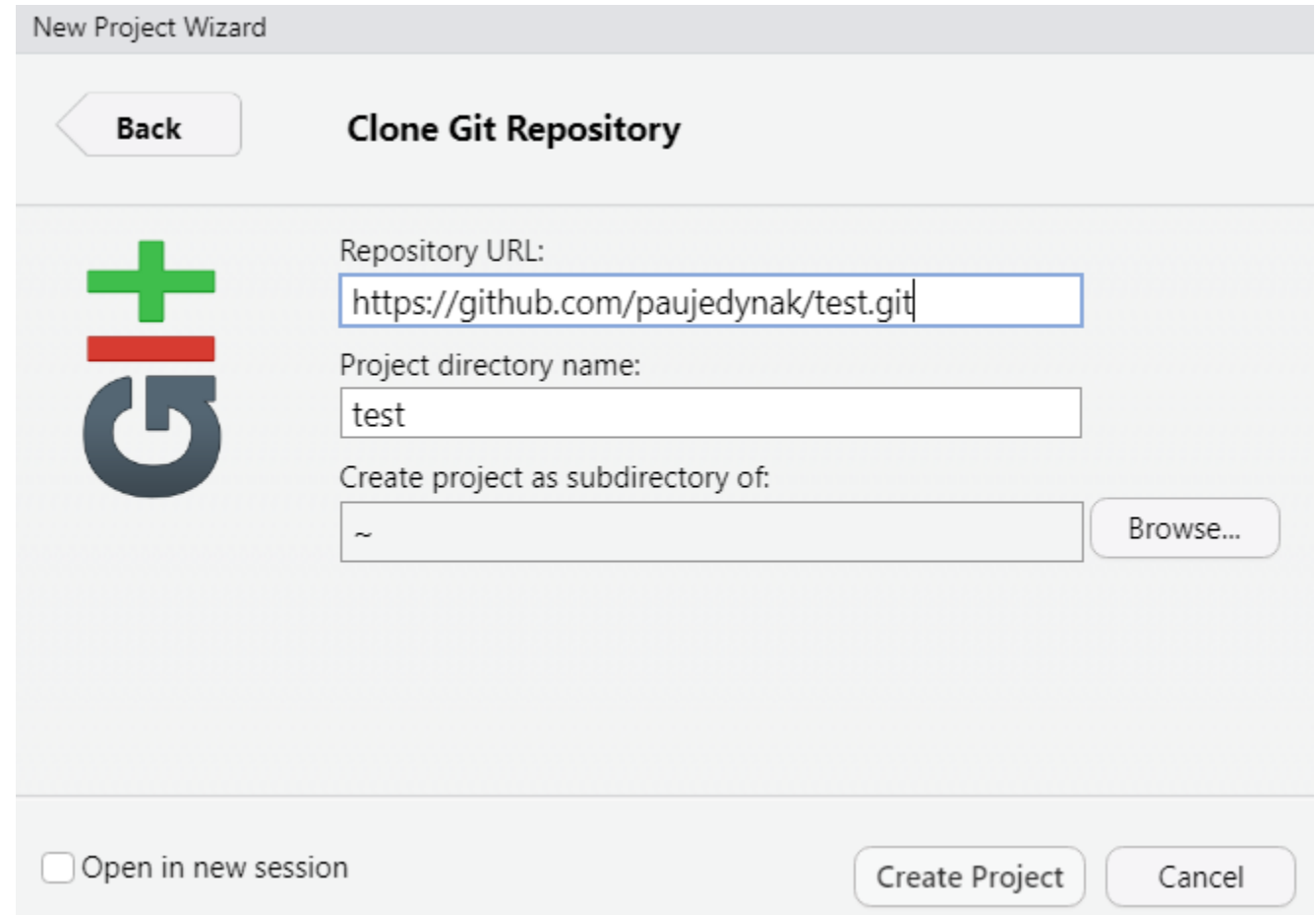
# 4. Clone the repo to your local machine

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



The screenshot shows the 'New Project Wizard' dialog box in RStudio, specifically the 'Clone Git Repository' step. The dialog has a title bar 'New Project Wizard' and a 'Back' button. On the left is the Git logo (a green plus sign over a blue 'G'). The main area contains three text input fields: 'Repository URL:' with the value 'https://github.com/paujedynak/test.git', 'Project directory name:' with the value 'test', and 'Create project as subdirectory of:' with the value '~'. A 'Browse...' button is next to the last field. At the bottom, there is a checkbox 'Open in new session' and two buttons: 'Create Project' and 'Cancel'.

New Project Wizard

Back

Clone Git Repository

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

Project directory name:  
test

Create project as subdirectory of:  
~

Browse...

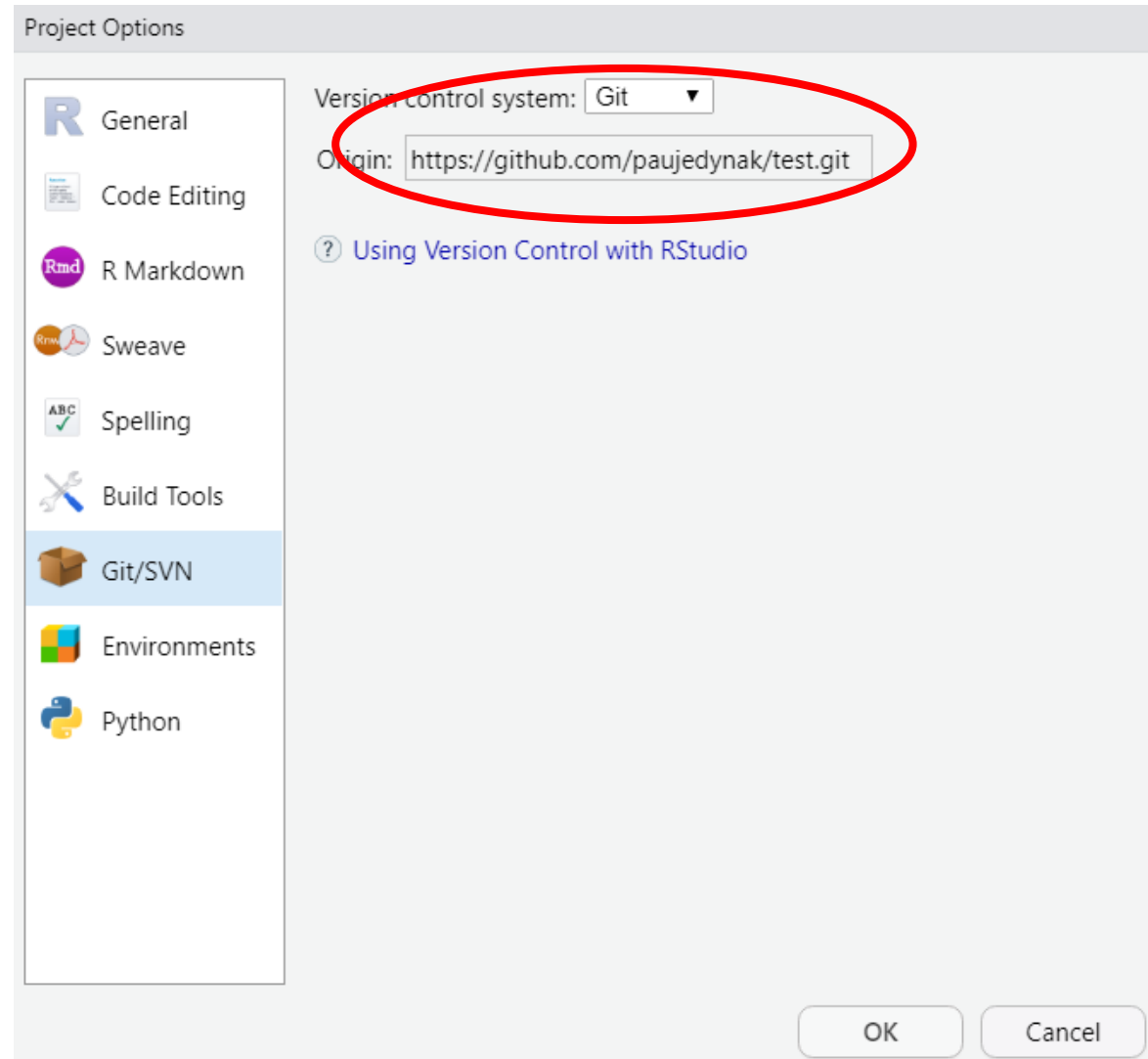
☐ Open in new session

Create Project Cancel

# 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

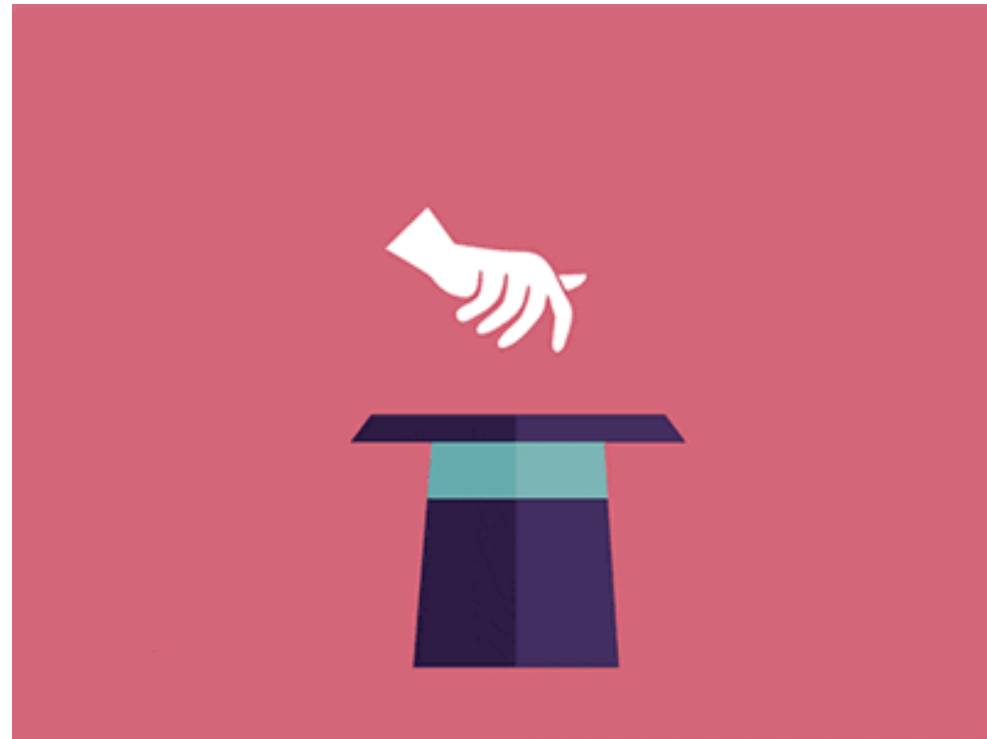


# 4. Clone the repo to your local machine

## Brand new R Project:

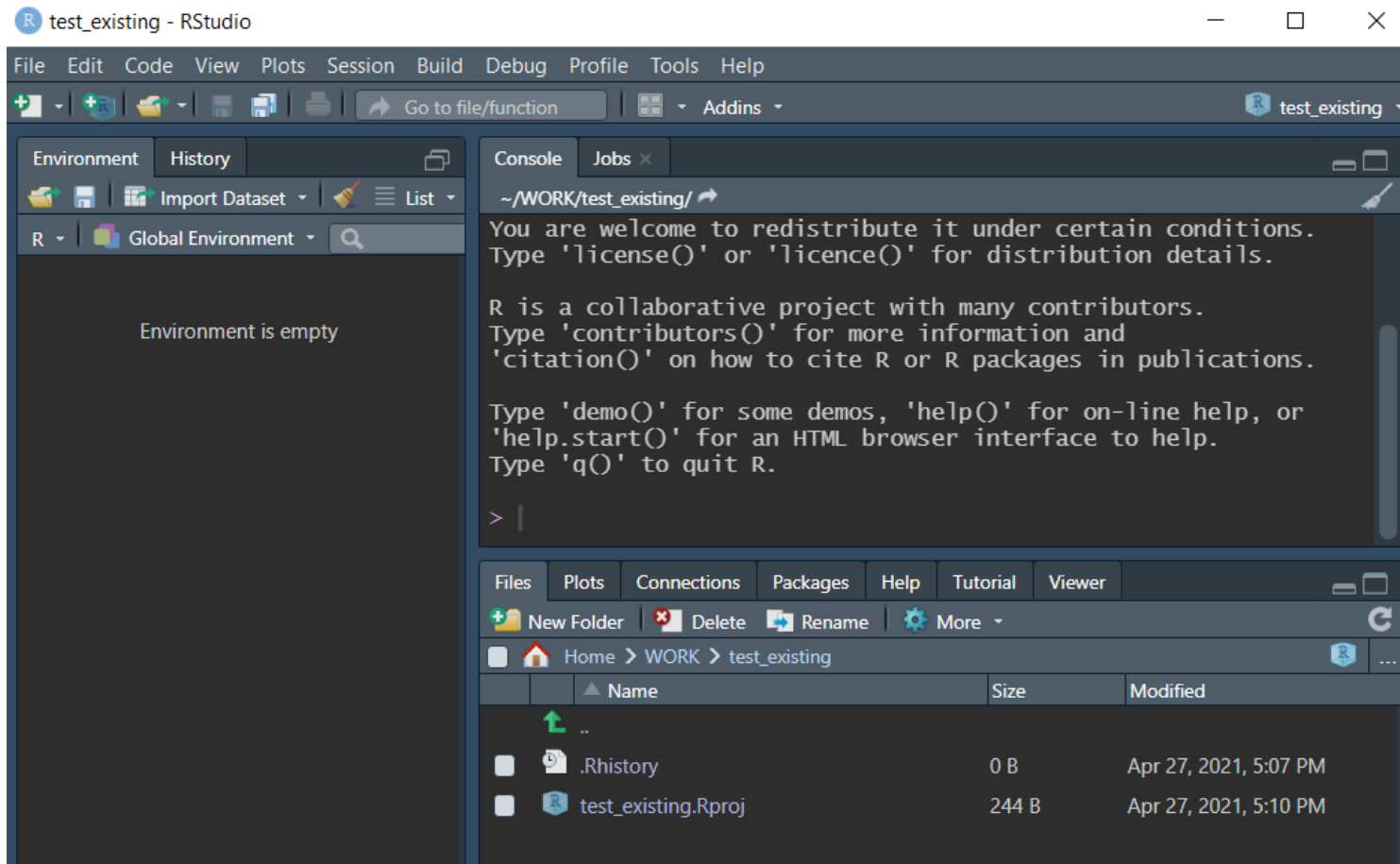
1. In RStudio >> File >> New Project...
2. Choose Version Control
3. Choose Git
4. Paste the link to your repository into the Repository URL field (the Project directory name will be filled automatically)
5. Click Browse... to choose the location where your new Project will be created
6. Click Create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

You can commit, push and pull...





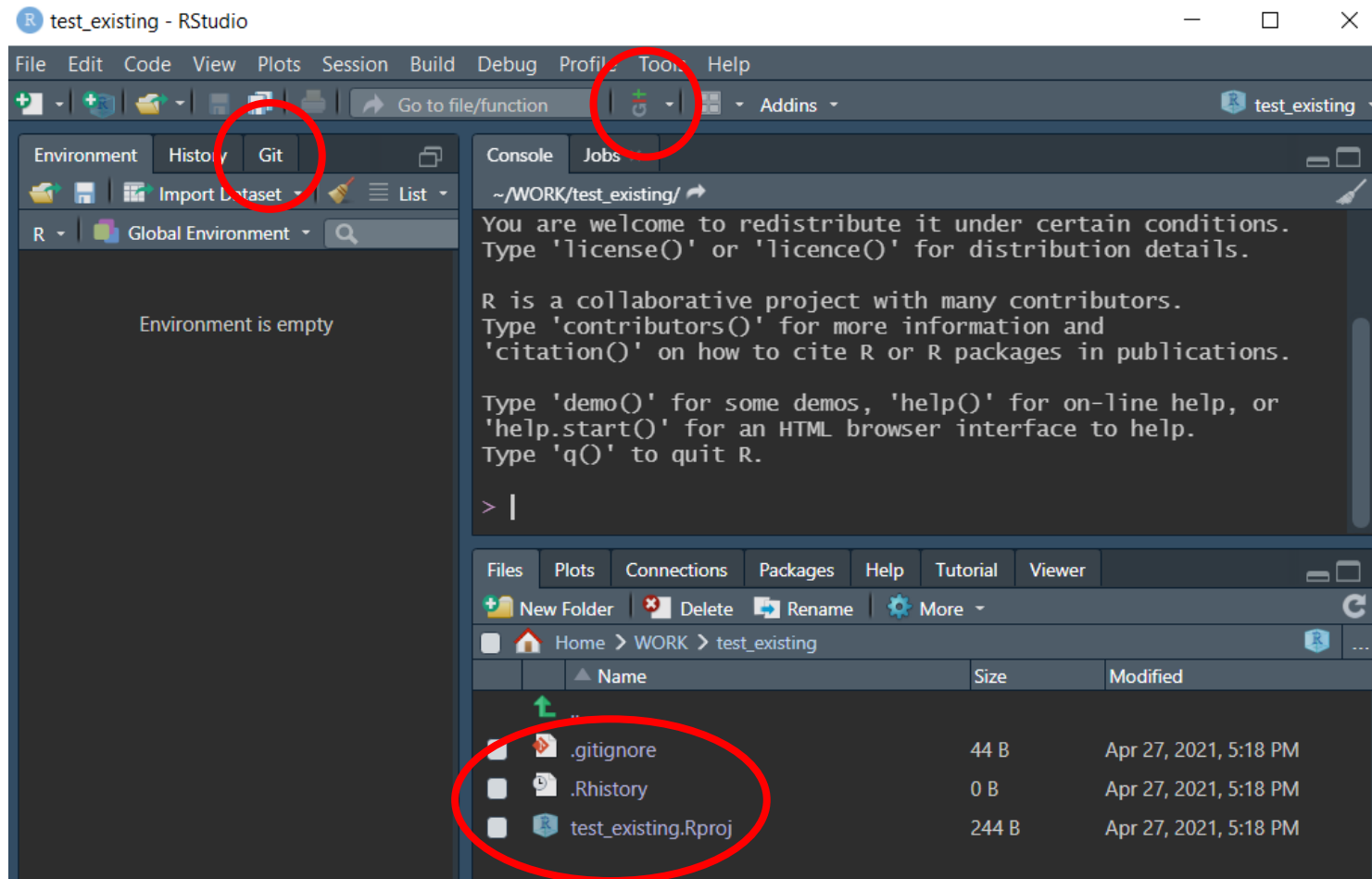
# 5. Connect Git to an existing R project



## Existing R Project:

1. Open an existing Project

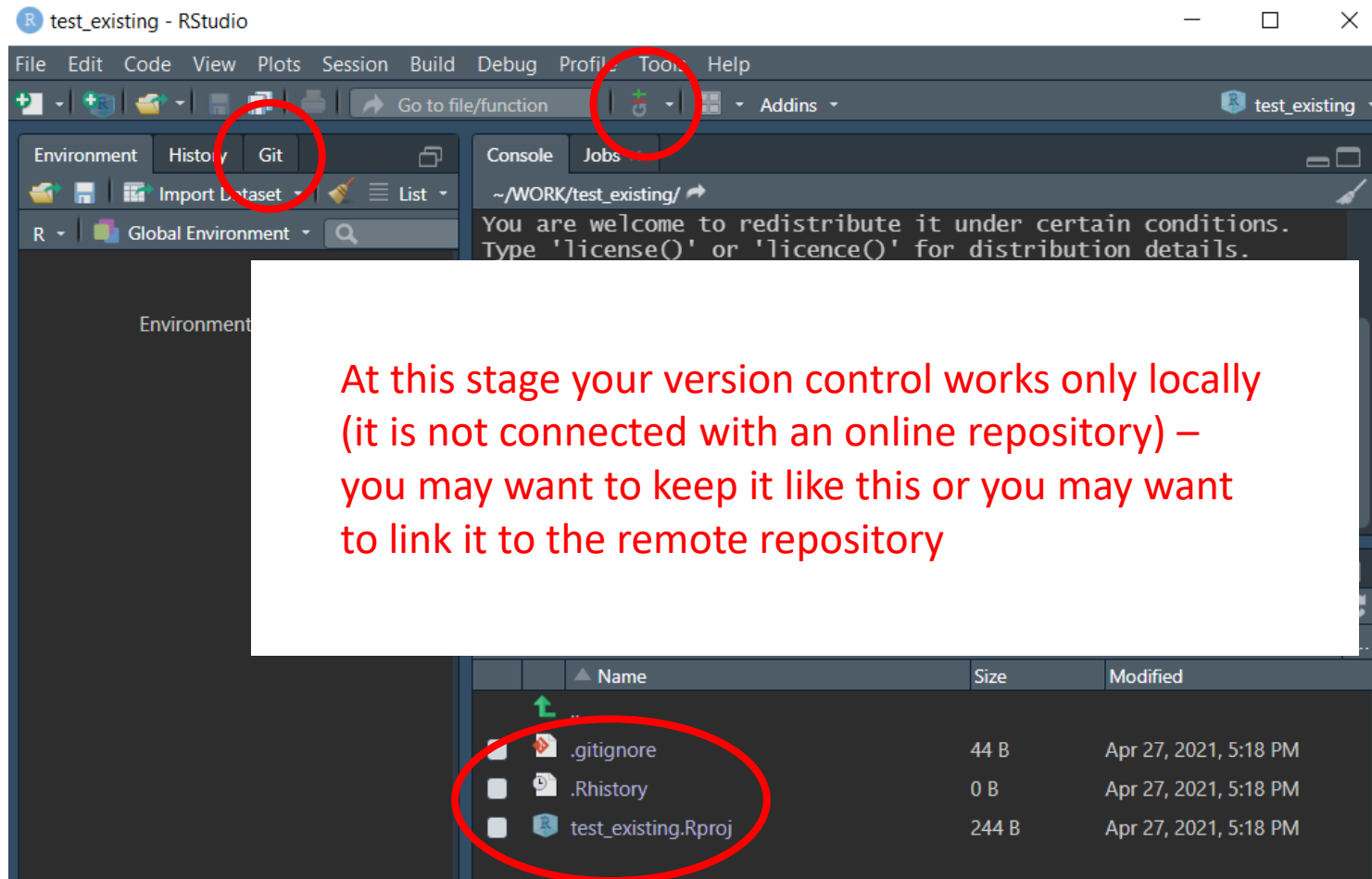
# 5. Connect Git to an existing R project



## 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

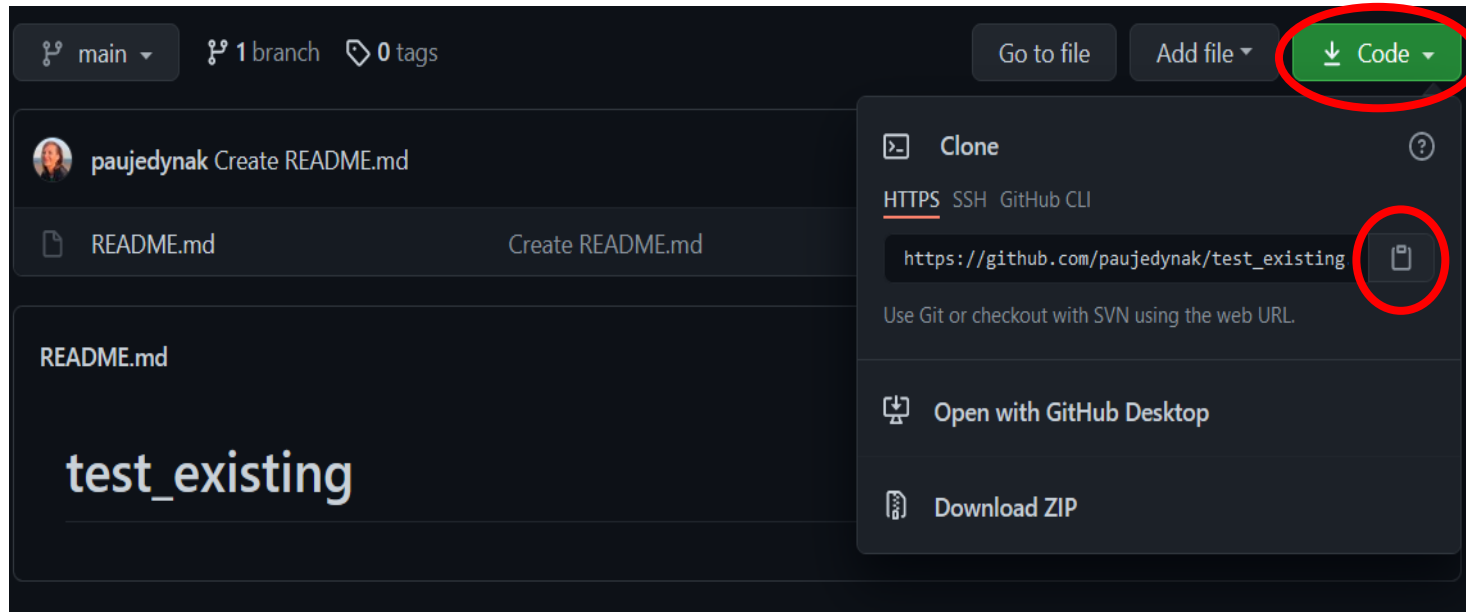
# 5. Connect Git to an existing R project



## 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

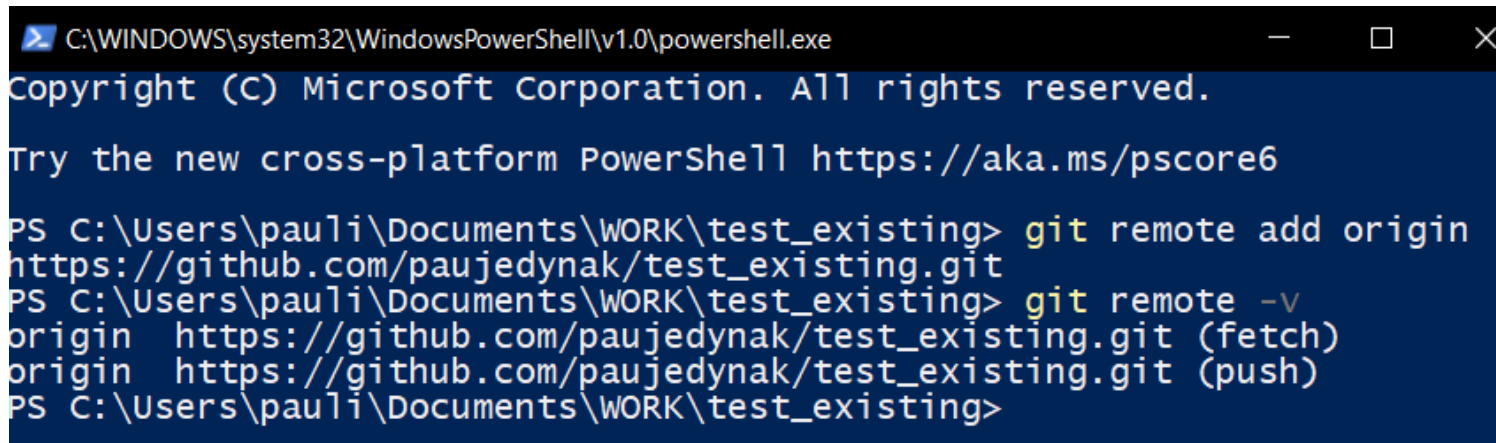
# 5. Connect Git to an existing R project



## 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



```
C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe
Copyright (C) Microsoft Corporation. All rights reserved.
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pauli\Documents\WORK\test_existing> git remote add origin
https://github.com/paujedynak/test_existing.git
PS C:\Users\pauli\Documents\WORK\test_existing> git remote -v
origin https://github.com/paujedynak/test_existing.git (fetch)
origin https://github.com/paujedynak/test_existing.git (push)
PS C:\Users\pauli\Documents\WORK\test_existing>
```

## 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:

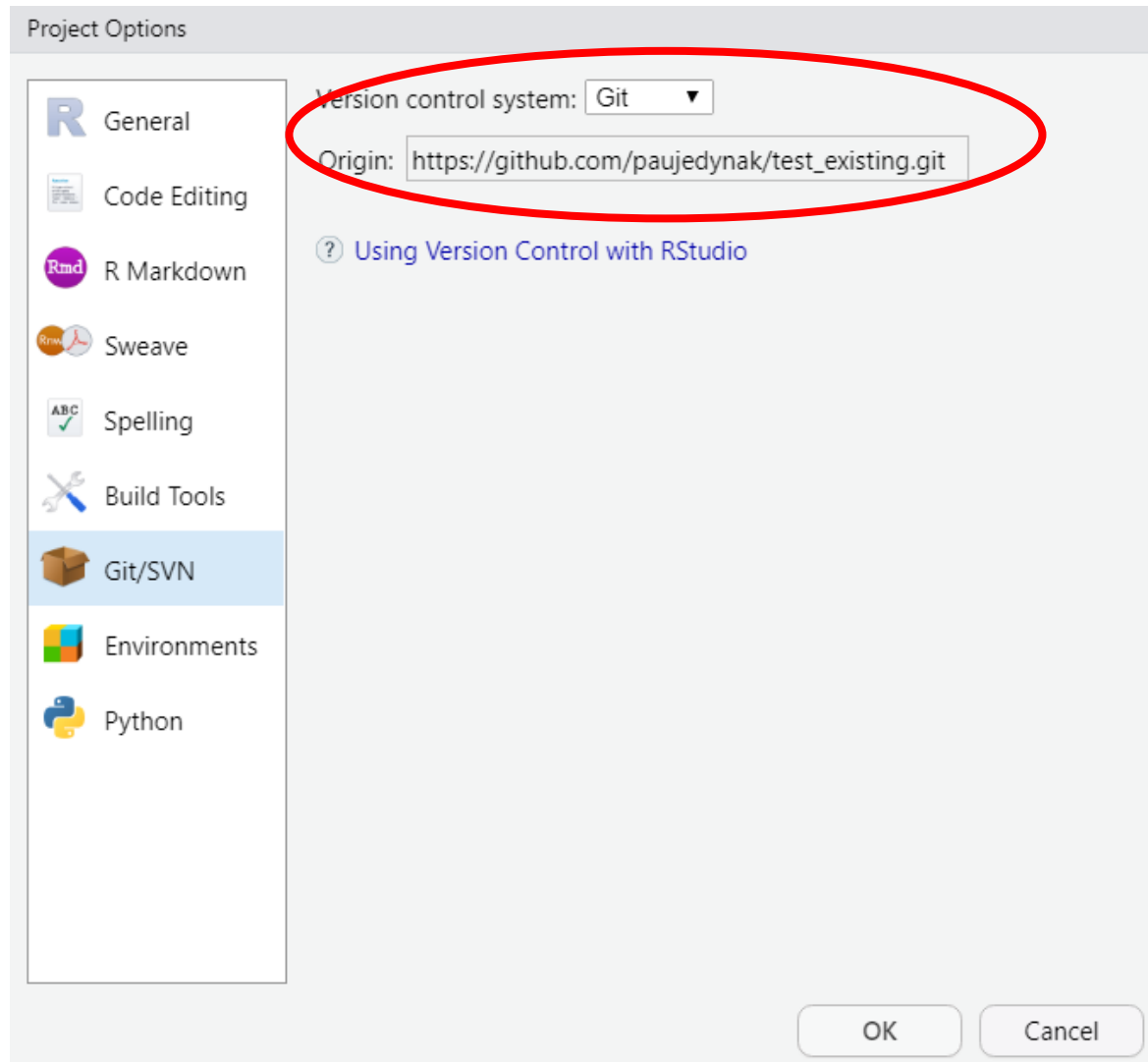
```
git remote add origin https://github.com/username/reponame.git
```

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



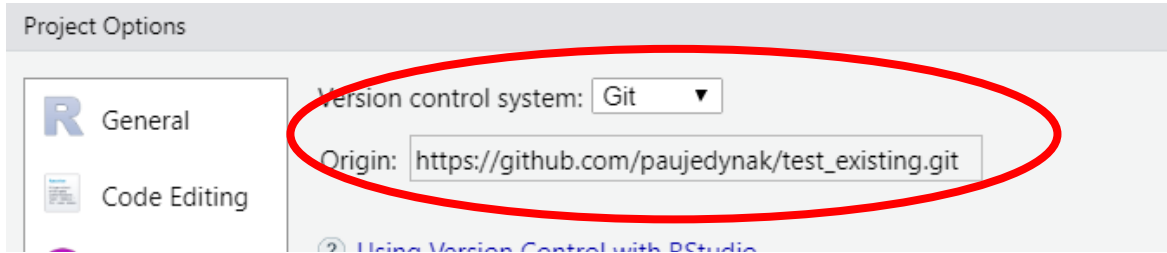
## Connect to an online repository:

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
```

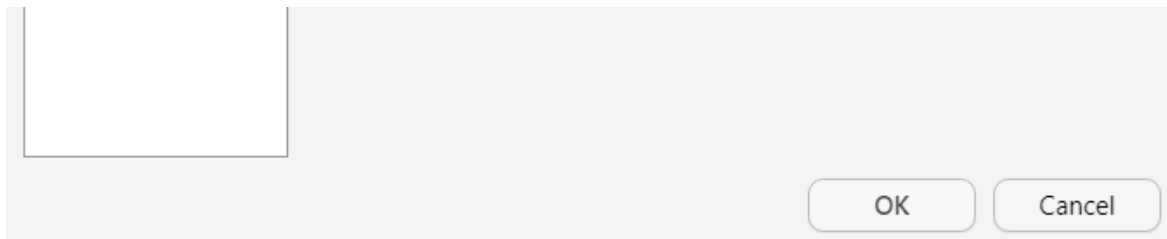
# 5. Connect Git to an existing R project



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
```

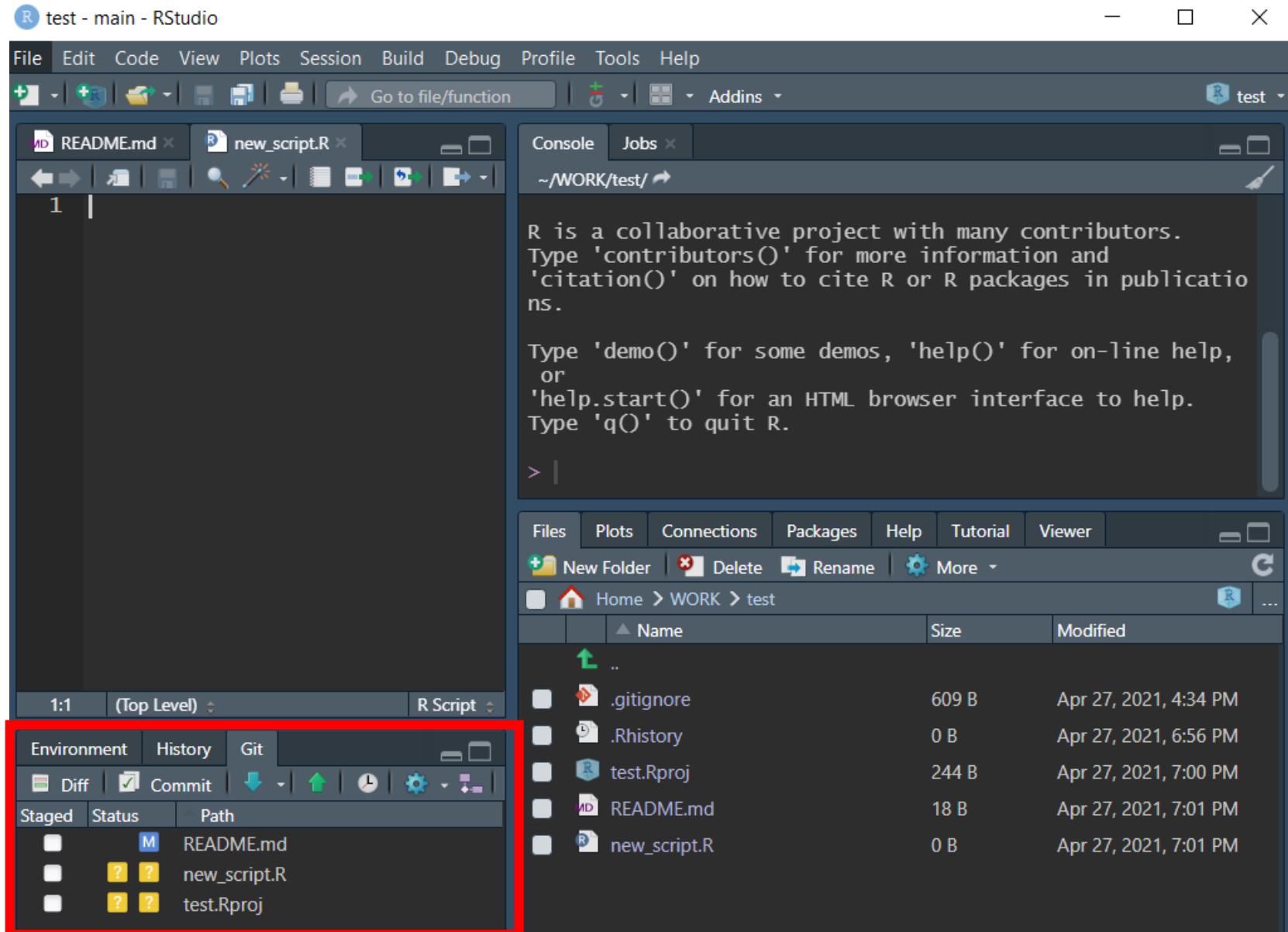


## 6. Commit, push and pull

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



# 6. Commit, push and pull

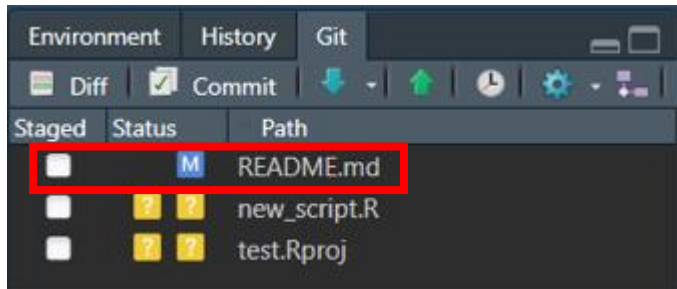


- All files within the Project are 'observed' by Git, unless added to the .gitignore file
- If there are any changes in the Project – new files created (new\_script.R), any SAVED changes made to the content of existing files (README.md) etc., Git will keep notifying you and will let you decide if to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

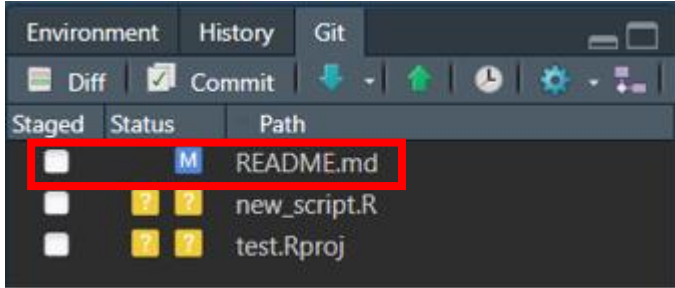
## 6. Stage, commit, push and pull

### CREATING NEW FILES, MODIFYING EXISTING ONES

Git informs you that new files appeared in the Project (? ? = 'unknown' status)



## 6. Stage, commit, push and pull



### CREATING NEW FILES, MODIFYING EXISTING ONES

Git informs you that new files appeared in the Project (? ? = 'unknown' status)

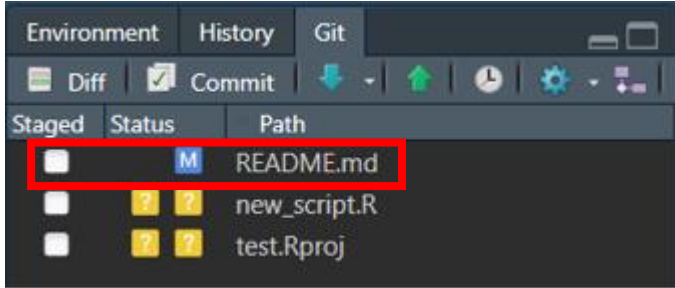
Git informs you that changes were made to a file (M = 'modified' status)

# 6. Stage, commit, push and pull

## CREATING NEW FILES, MODIFYING EXISTING ONES

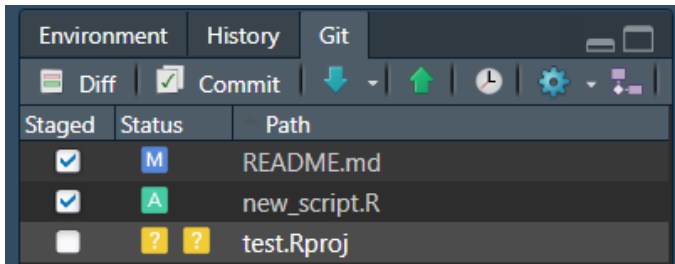
Git informs you that new files appeared in the Project (? ? = 'unknown' status)

Git informs 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

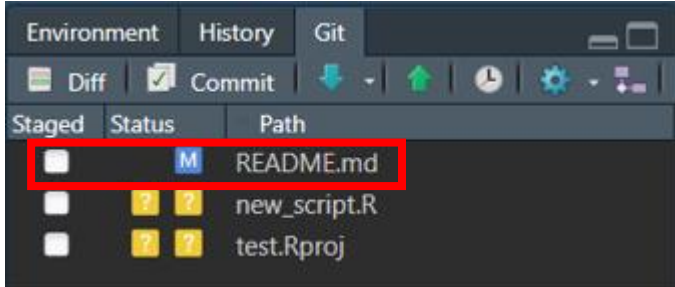


# 6. Stage, commit, push and pull

## CREATING NEW FILES, MODIFYING EXISTING ONES

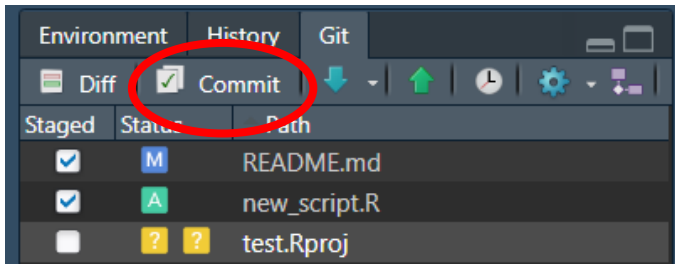
Git informs you that new files appeared in the Project (? ? = 'unknown' status)

Git informs you that changes were made to a file (M = 'modified' status)

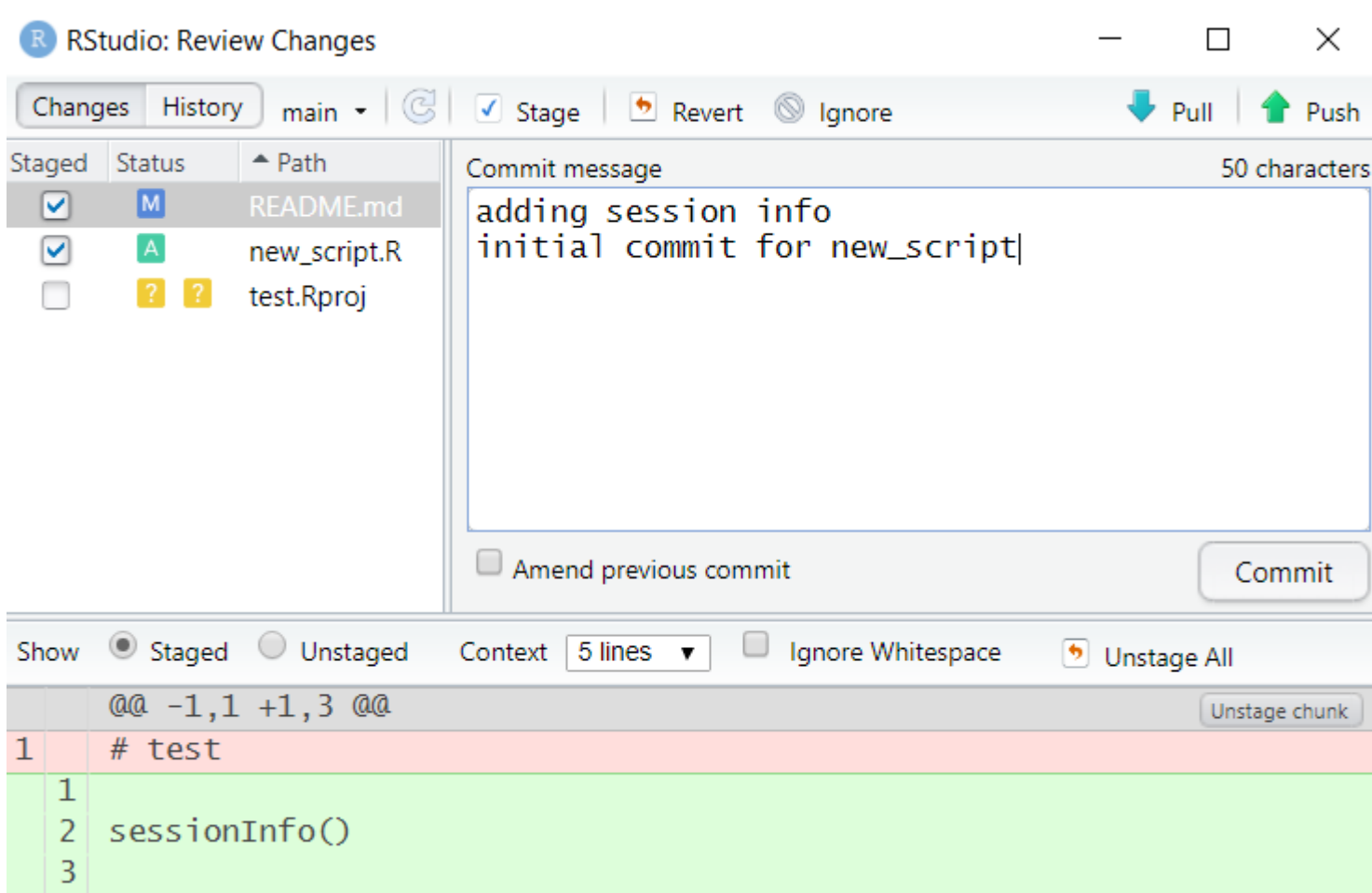


## STAGING

- 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



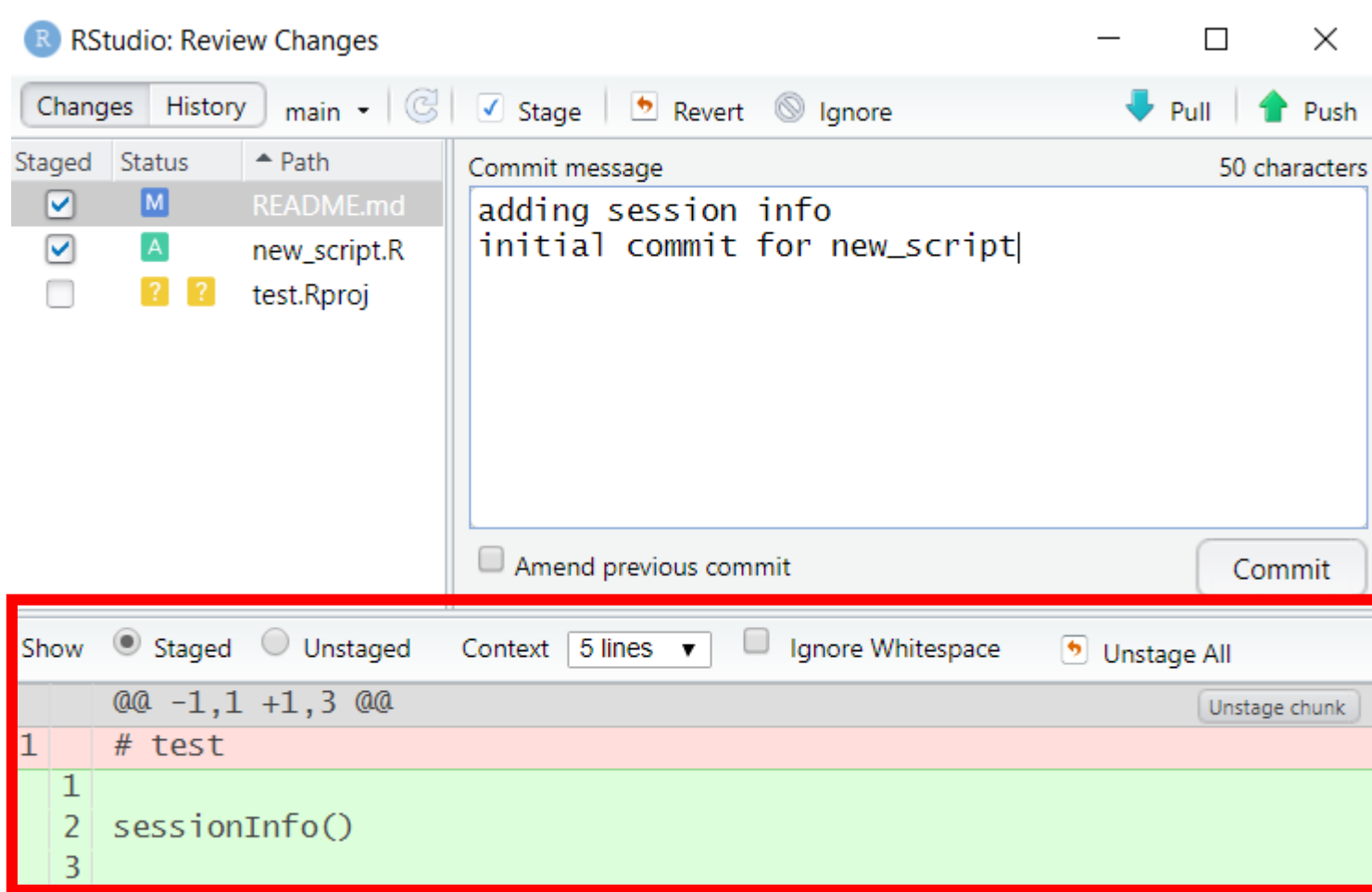
# 6. Stage, commit, push and pull



## 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

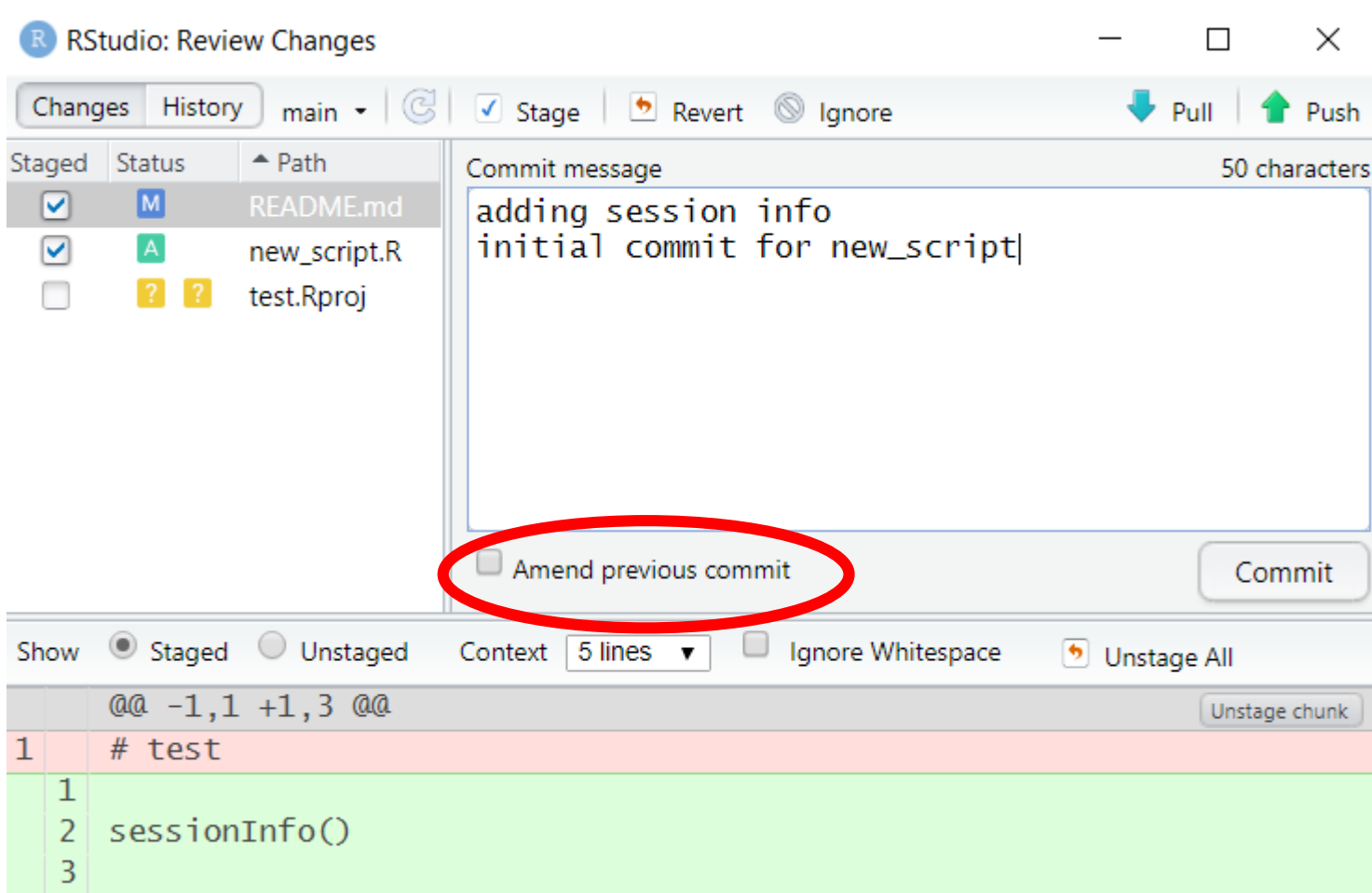


## COMMIT

- 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



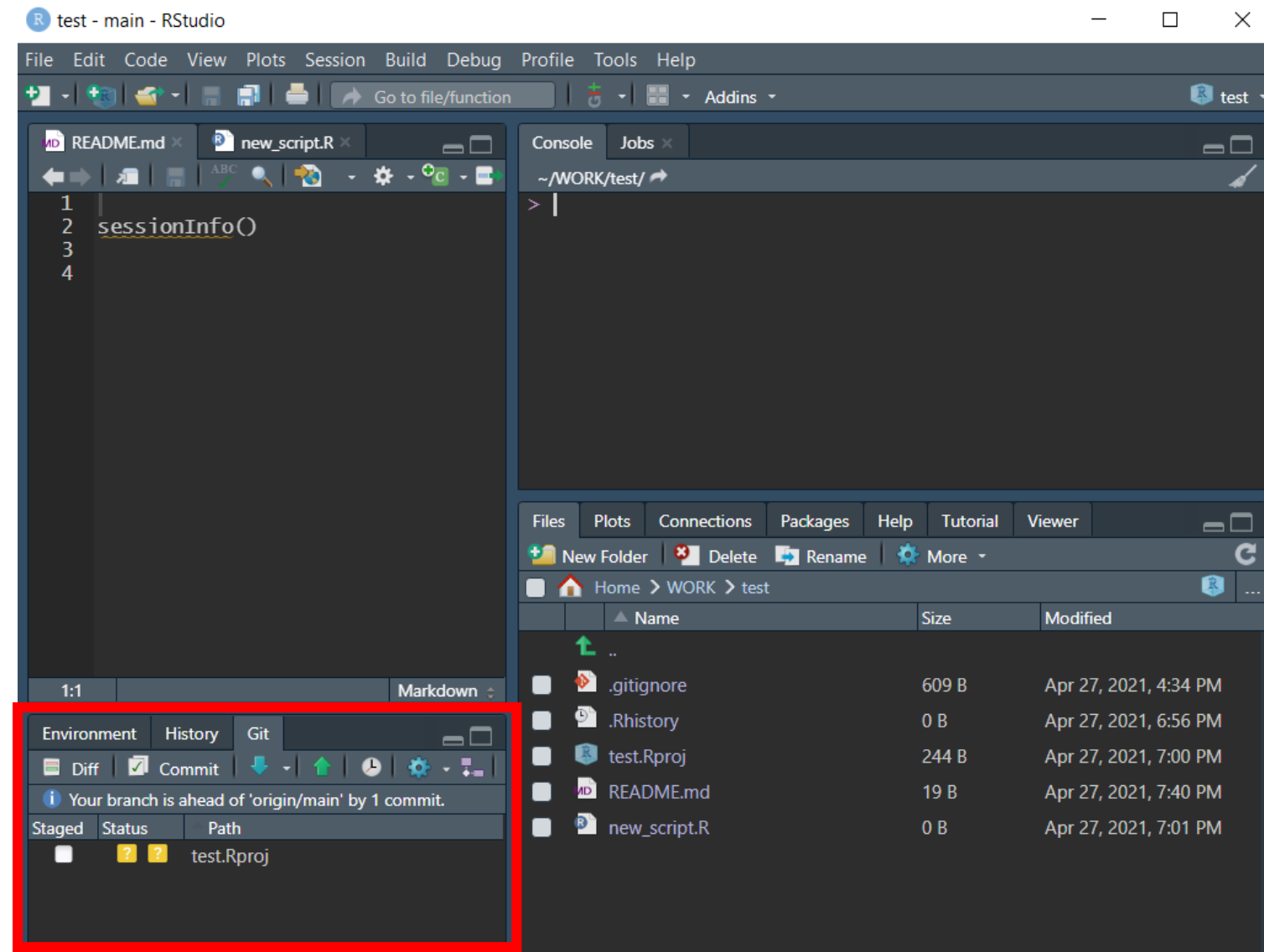
## COMMIT

- 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

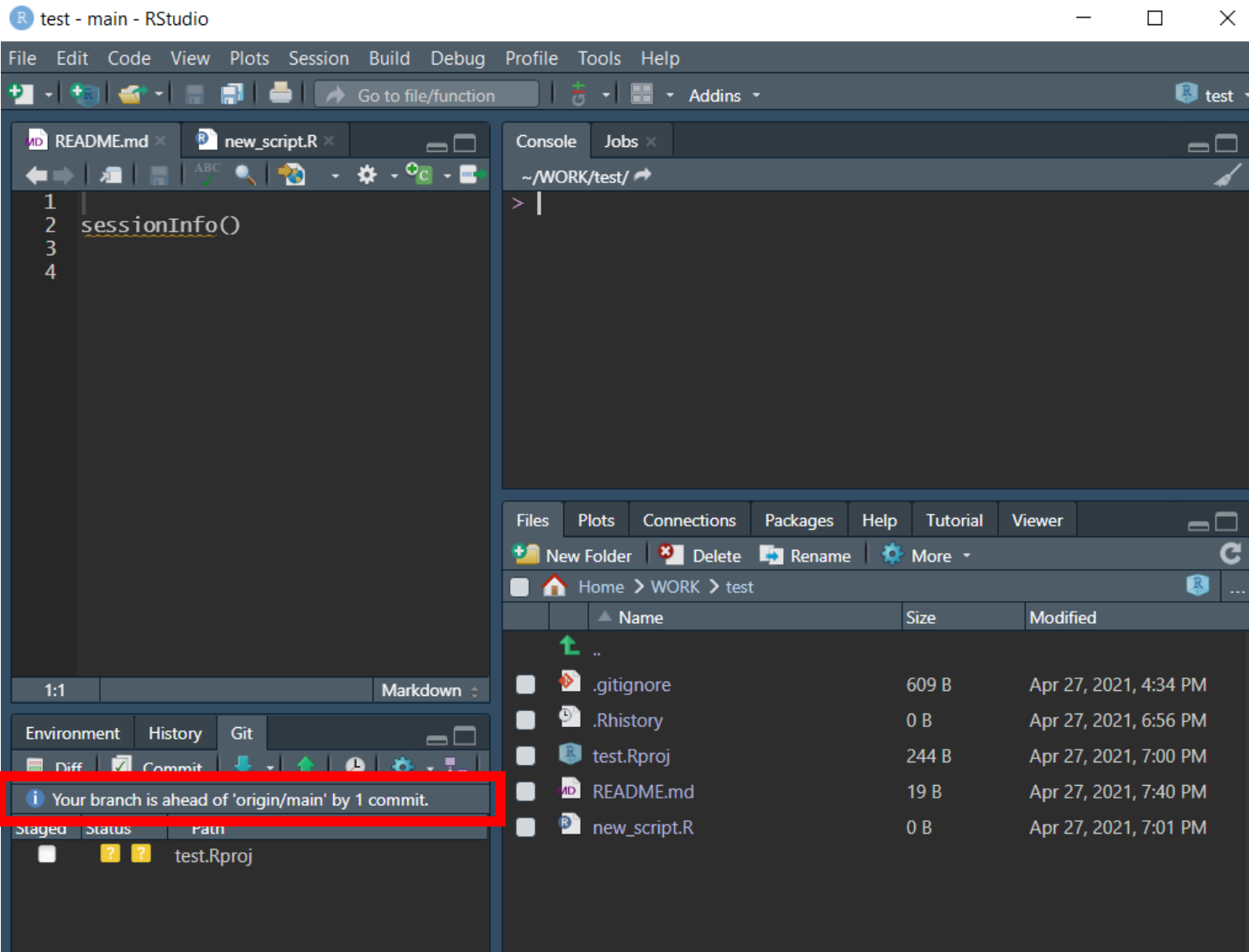
- After the commit, the staged files disappear



# 6. Stage, commit, push and pull

## PUSH

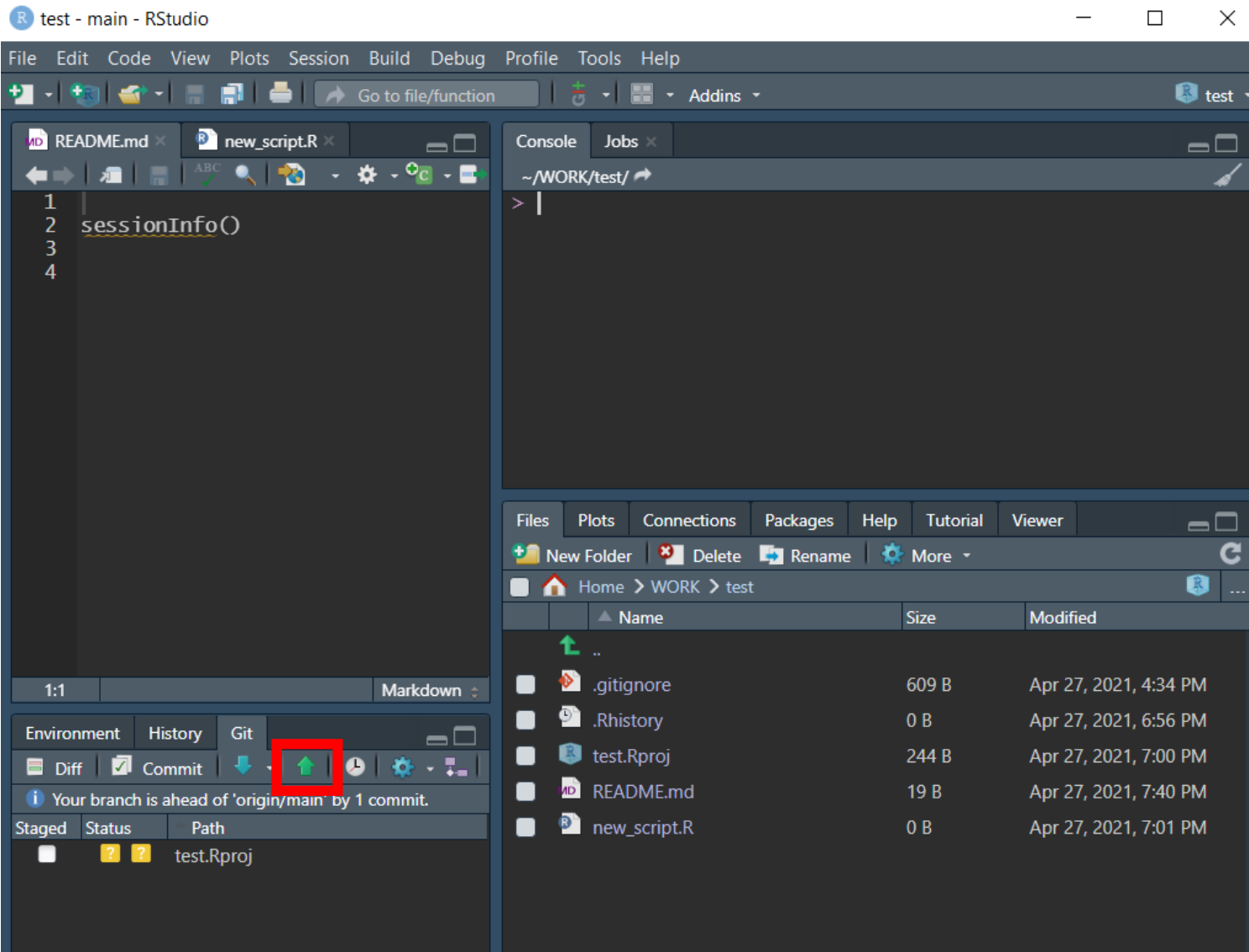
- After the commit, the staged files disappear
- You are also informed that your local repository is ahead of the remote one by one commit (your local version is newer than the online one by one commit)
- You do not need to push each time you make a commit, you can do it e.g. at the end of the day



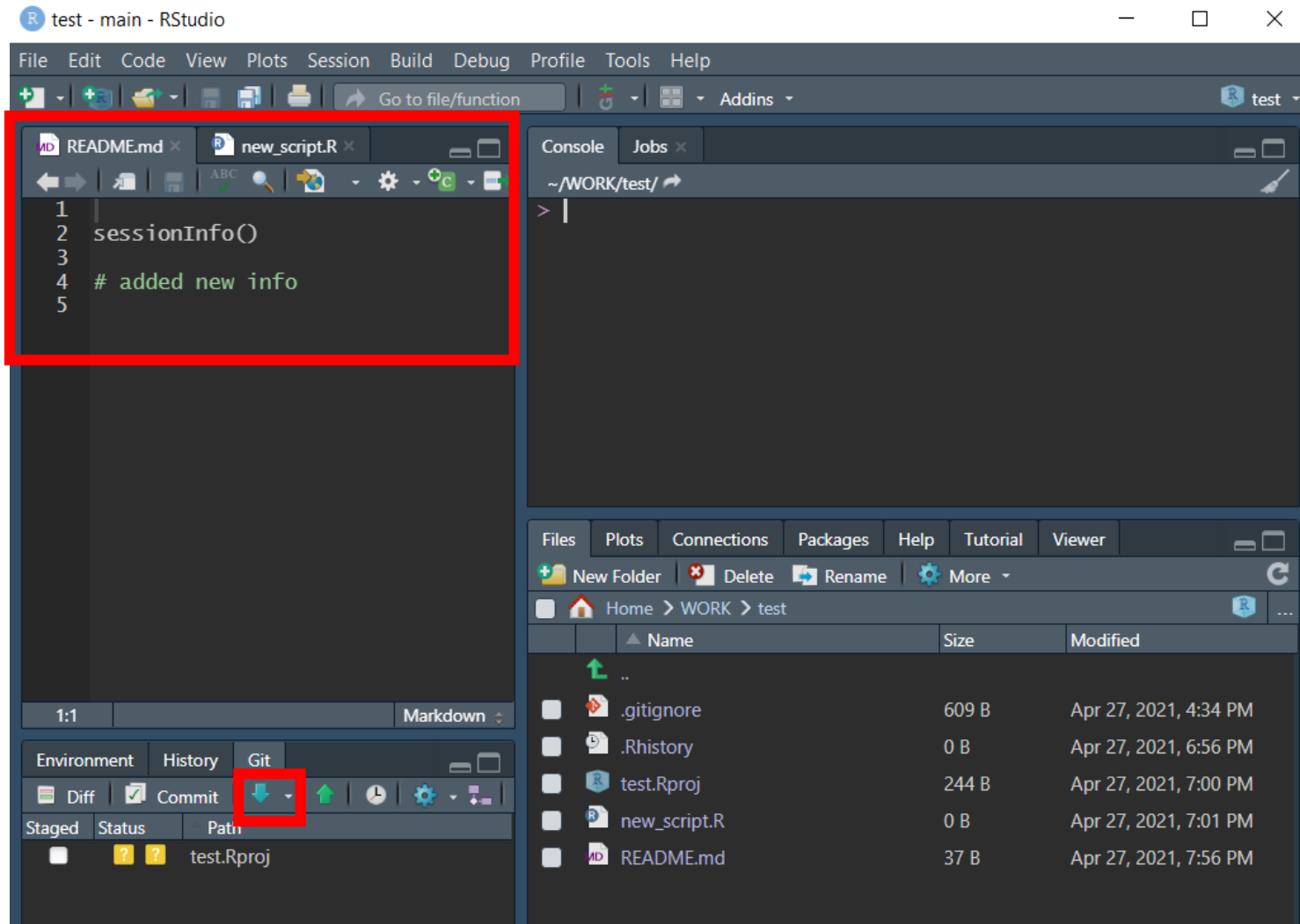
# 6. Stage, commit, push and pull

## 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



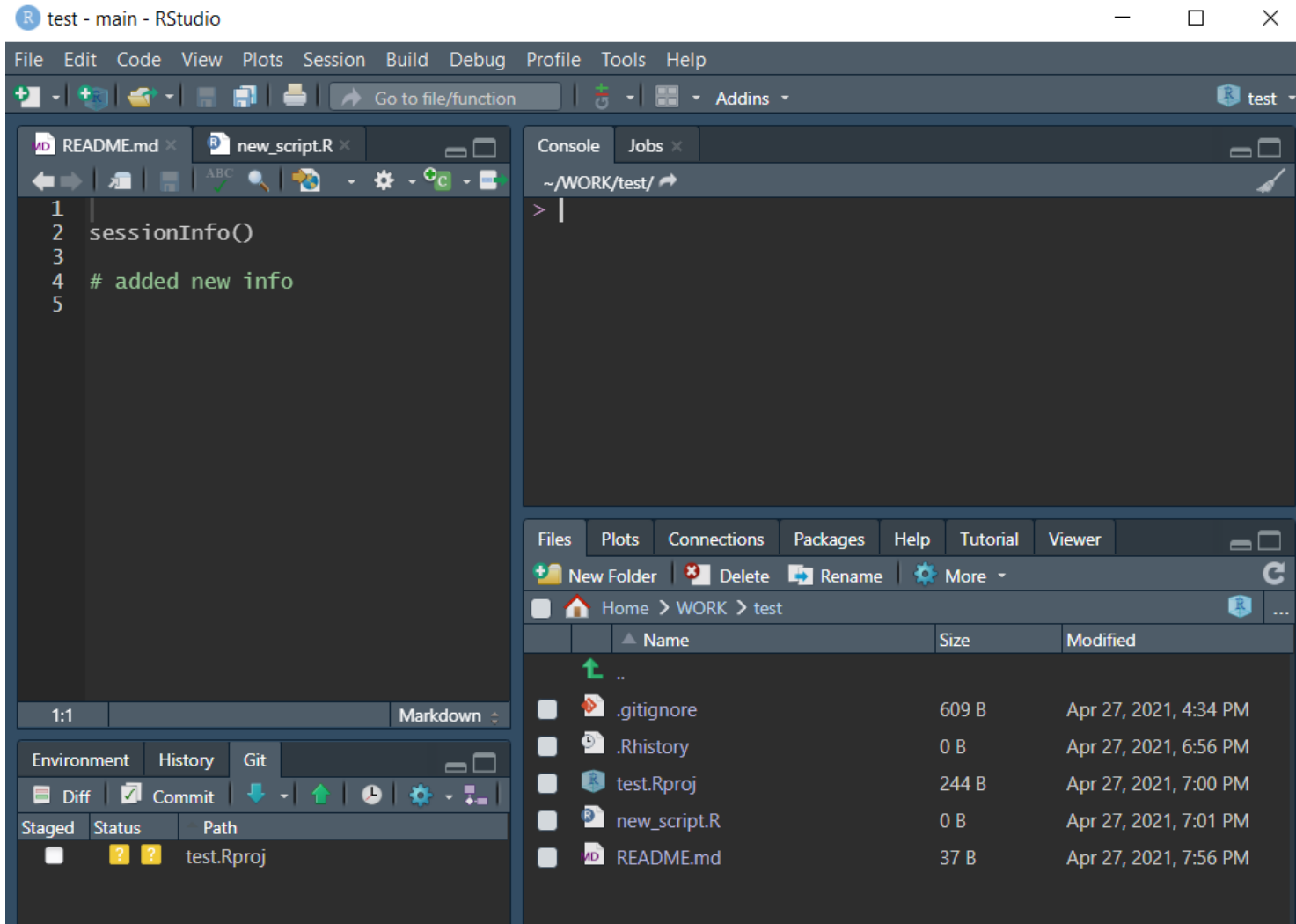
# 6. Stage, commit, push and pull



## 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

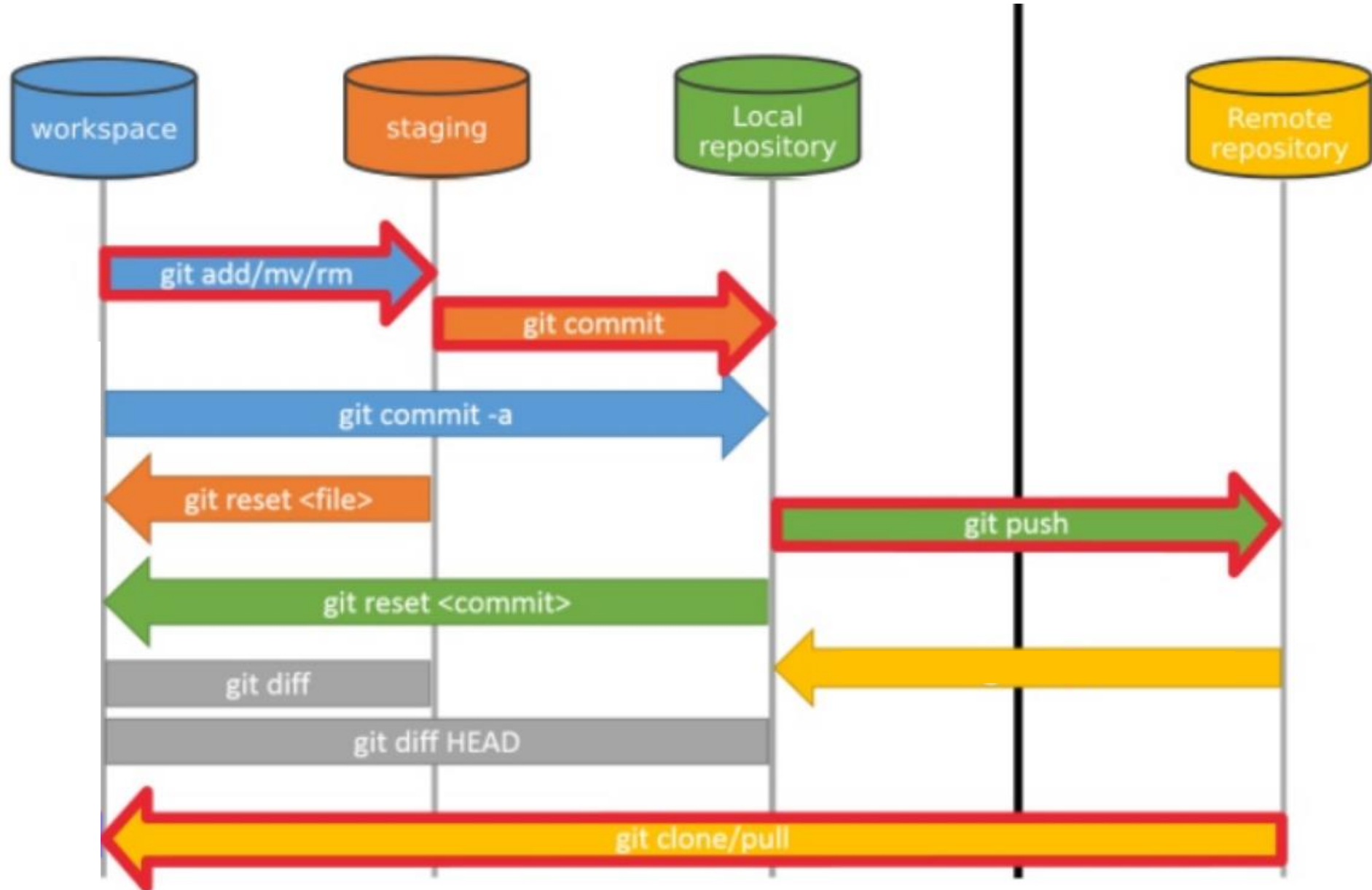
# 6. Stage, commit, push and pull



## PULL

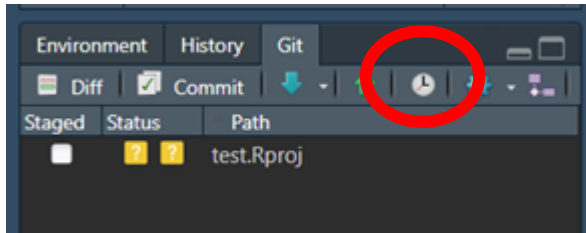
- If you want to e.g. synchronize your work between machines (e.g. your computer at work and another one at home; your machine and your colleague's machine who works with you on a project, etc.), on the machine where the work is outdated, click the blue 'Pull' button
- Pulling will 'download' new versions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



RStudio: Review Changes

Changes History main (all commits) Search Pull

Subject	Author	Date	SHA
HEAD -> refs/heads/main origin/main origin/HEAD Update README.md	Paulina Jedynak <polina.jedynak@ 2021-04-27	2021-04-27	bf5a6458
adding session info	Paulina Jedynak <polina.jedynak@ 2021-04-27	2021-04-27	d825a825
Create README.md	Paulina Jedynak <polina.jedynak@ 2021-04-27	2021-04-27	5599a5ef
Initial commit	Paulina Jedynak <polina.jedynak@ 2021-04-27	2021-04-27	dea17db2

Commits 1-4 of 4

**SHA** bf5a64584e694a0ca9a535eeac862076caac3019

**Author** Paulina Jedynak <polina.jedynak@univ-grenoble-alpes.fr>

**Date** 2021-04-27 17:55

**Subject** Update README.md

**Parent** d825a825d959dda23615eeff4876a08756192e9b

README.md

README.md View file @ bf5a6458

@@ -1,3 +1,4 @@

1	1	
2	2	sessionInfo()
3	3	
4		# added new info



## 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](https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things)

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

RStudio: Review Changes

Changes History main (all commits) Search Pull

Subject	Author	Date	SHA
HEAD -> refs/heads/main origin/main origin/HEAD Update README.md	Paulina Jedynak <paulina.jedynak@ 2021-04-27	bf5a6458	
adding session info	Paulina Jedynak <paulina.jedynak@ 2021-04-27	d825a825	
Create README.md	Paulina Jedynak <paulina.jedynak@ 2021-04-27	5599a5ef	
Initial commit	Paulina Jedynak <paulina.jedynak@ 2021-04-27	dea17db2	

## 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

Commits 1-4 of 4

SHA d825a825d959dda23615eff4876a08756192e9b

Author Paulina Jedynak <paulina.jedynak@gmail.com>

Date 2021-04-27 17:46

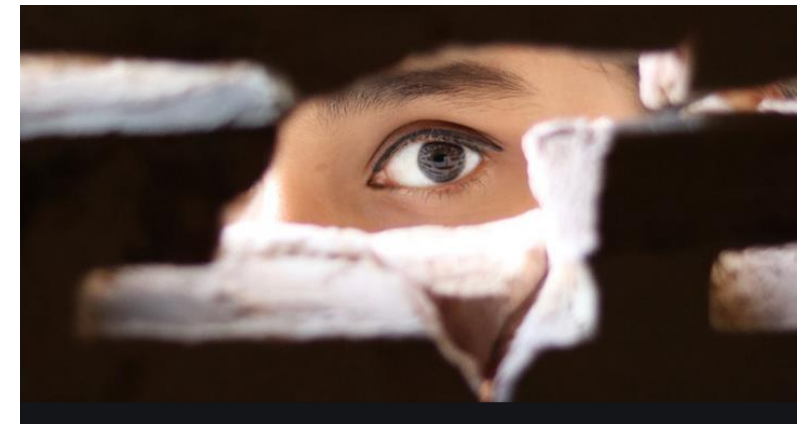
Subject adding session info

Parent 5599a5ef318b18156b0c1d1d140425d4285c6680

MD README.md

MD README.md View file @ d825a825

```
@@ -1,1 +1,3 @@
1 # test
2 sessionInfo()
3
```



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

The most common scenarios when I go back in time:

The screenshot shows the RStudio 'Review Changes' interface. At the top, there's a 'Changes' tab and a 'History' tab. The 'History' tab is active, showing a list of commits. The commit 'adding session info' is selected, and its details are shown below. A red circle highlights the 'View file @ d825a825' link. The diff view shows the changes to 'README.md'.

Subject	Author	Date	SHA
Update README.md	Paulina Jedynak <paulina.jedynak@>	2021-04-27	bf5a6458
adding session info	Paulina Jedynak <paulina.jedynak@>	2021-04-27	d825a825
Create README.md	Paulina Jedynak <paulina.jedynak@>	2021-04-27	5599a5ef
Initial commit	Paulina Jedynak <paulina.jedynak@>	2021-04-27	dea17db2

SHA: d825a825d959dda23615eeff4876a08756192e9b  
Author: Paulina Jedynak <paulina.jedynak@gmail.com>  
Date: 2021-04-27 17:46  
Subject: adding session info  
Parent: 5599a5ef318b18156b0c1d1d1404254d285c6680

View file @ d825a825

```
@@ -1,1 +1,3 @@
1 # test
2 sessionInfo()
3
```

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 newer 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](https://github.com/paujedynak)  
[paulina.jedynak@gmail.com](mailto:paulina.jedynak@gmail.com)

Environmental Epidemiology lab  
[gricad-gitlab.univ-grenoble-alpes.fr](https://gricad-gitlab.univ-grenoble-alpes.fr)

