

# Introduction to GitHub with R




## Workshop Setup:

- ▶ Wi-Fi

Network Name: N/A

Password: N/A

- ▶ Resources


R (version 3.6.3) 

RStudio (version 1.3.1073)  RStudio

Git (version 2.28.0) 

# What is GitHub?

Git  is an open source version control system and is free to use!

 **GitHub** is a code hosting platform that offers a user-friendly web-based interface to use Git.

You can work on public or private **repositories** which means that you can collaborate on projects with others from anywhere in the world!




Repository: A storage space where your project files and code live

# Topics

► Workshop aim:

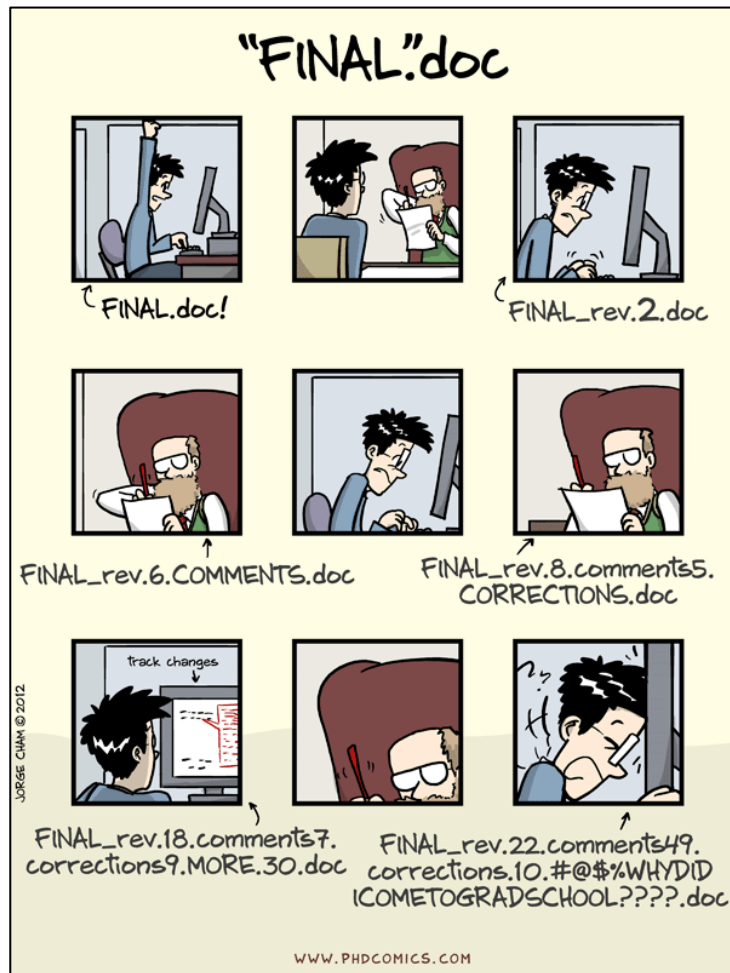
Learn how to version control an R project using GitHub within Rstudio and the basics in using Git.

► Topics:

- Install Git 
- Setup a GitHub account
- Create a GitHub repository
- Connect repository with R project
- Basic version control processes



# Why version control?



Because I want to...

- ▶ have a backup of the code
- ▶ see what has changed in the code over time
- ▶ be able to roll back my code to a previous version
- ▶ collaborate effectively with helpful documentation
- ▶ avoid multiple copies or versions

# Workshop tasks



For this workshop we will take the step by step approach to exercises. I will cover one step and demonstrate it on my machine and then allow you a few minutes to repeat the step.

This will ensure that everyone is on the same page and troubleshoot early than when it is too late!

# Install Git

In order to start using the Git version control system we need to install it on our machine.

**To download Git you need to go to:** <https://git-scm.com/downloads>

and select your platform.

We then follow the next steps to install Git ...



# Install Git



## Downloading Git



### Your download is starting...

You are downloading the latest (2.28.0) 64-bit version of Git for Windows. This is the most recent [maintained build](#). It was released **about 1 month ago**, on 2020-07-28.

[Click here to download manually](#), if your download hasn't started.

### Other Git for Windows downloads

Git for Windows Setup

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

Git for Windows Portable ("thumbdrive edition")

[32-bit Git for Windows Portable.](#)

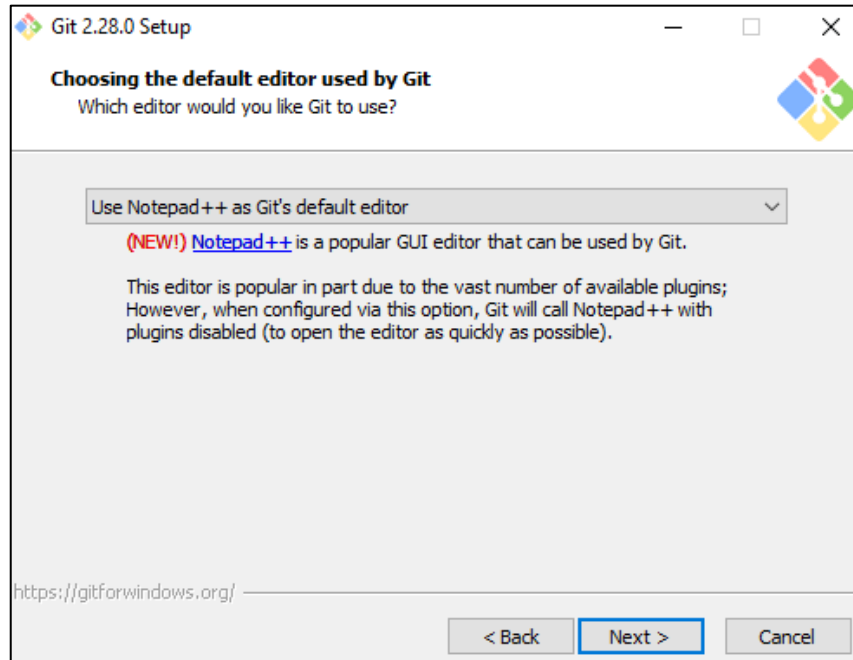
[64-bit Git for Windows Portable.](#)

The current source code release is version 2.28.0. If you want the newer version, you can build it from [the source code](#).

Open the executable file that you just downloaded and follow the instructions (by clicking next) to complete the installation.



# Install Git



There are several editors to choose from. Notepad++ is a simple editor that can be useful in general to have installed on your machine. However please note that when coding in R we use RStudio to edit our scripts.

- ☒ **Git from the command line and also from 3rd-party software**  
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Window PowerShell as well as any third-party software looking for Git in PATH.

Follow the recommended setup and follow the instructions (by clicking next) to complete the installation.

# Setup a GitHub account

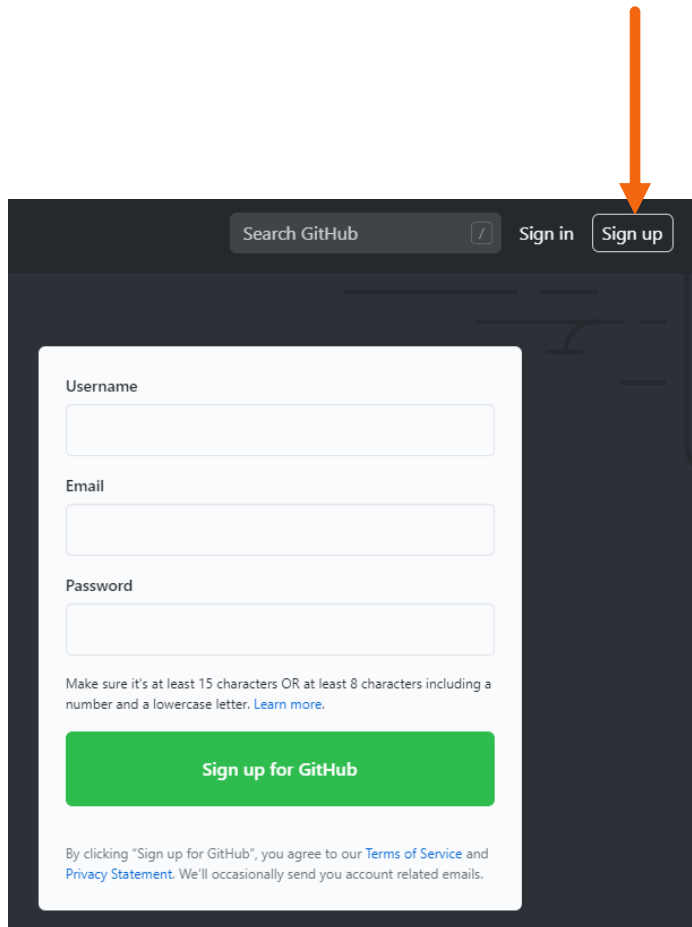
In order to start version controlling your projects online or even start collaborating with others you will need a GitHub account.

- ▶ **For this workshop you need a GitHub account.**
- ▶ **If you do not have one then you need to create an account at:**  
<https://github.com/>



You do not need to download a software, however it's useful to know that there is a GitHub desktop version: <https://desktop.github.com/>

# Setup a GitHub account



Search GitHub [icon] Sign in Sign up

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.



Join GitHub

## Create your account

Username \*

Email address \*

Password \*

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences  
☐ Send me occasional product updates, announcements, and offers.

Verify your account

Please solve this puzzle so we know you are a real person

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

It is a good idea to use a username that can easily be associated with you.

For example:

first name initial + last name....

nicolas attalides



**nattalides**

# Create a GitHub repository

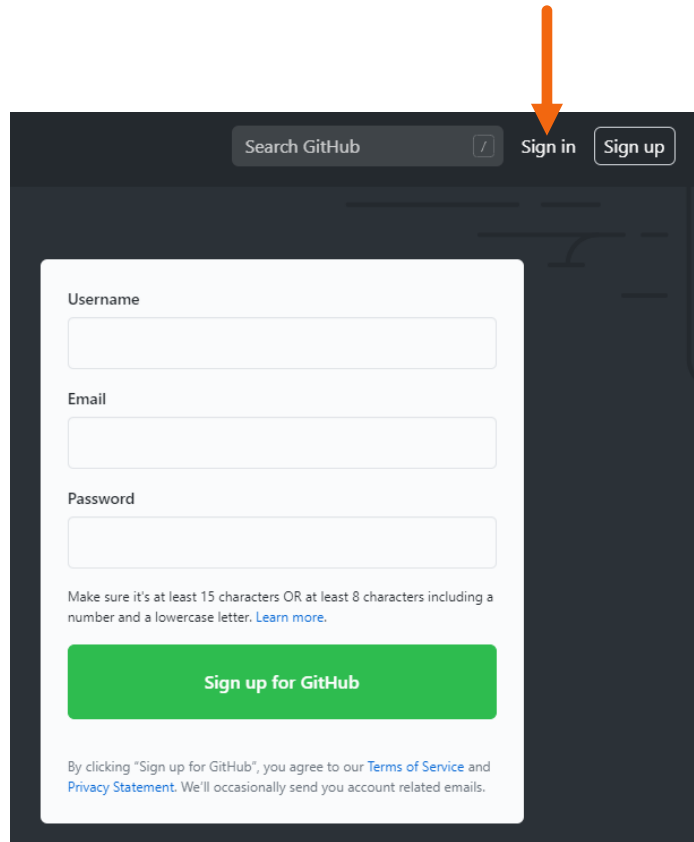
Once we have our GitHub account up and running the next step is to create a new repository. This is the space where your folder and files such R scripts are stored and version controlled. It is good practice for each project that you work on to have its own dedicated repository.

- ▶ **Sign in to our GitHub account**
- ▶ **Create a new public repository**

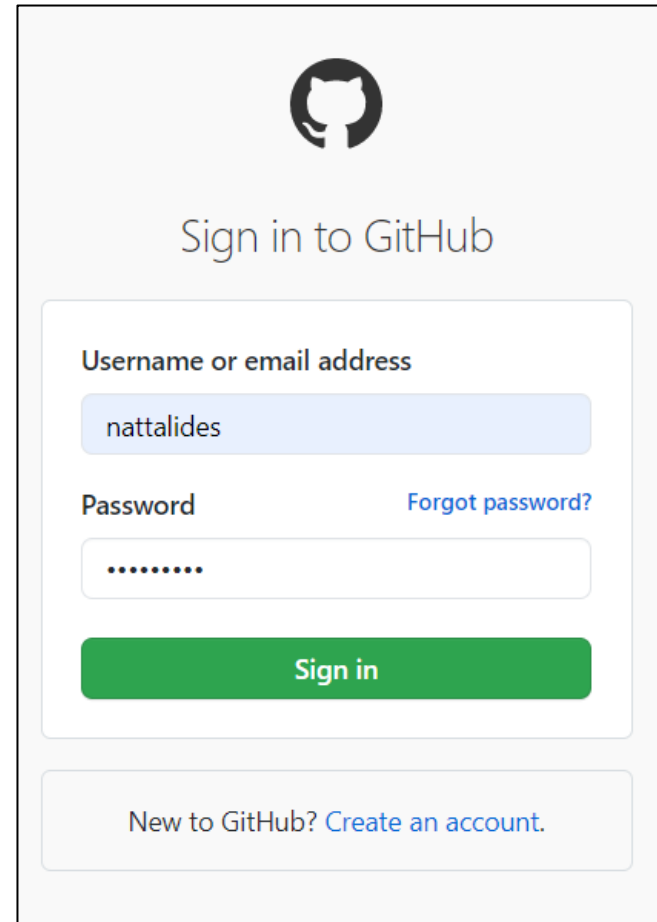


You can control who can access your project by creating a **private repository**, you can also convert a public to private and vice versa

# Create a GitHub repository



A screenshot of the GitHub sign-up page. At the top, there is a search bar labeled "Search GitHub" and two buttons: "Sign in" and "Sign up". An orange arrow points to the "Sign up" button. Below the search bar, there is a form with three input fields: "Username", "Email", and "Password". Below the "Password" field, there is a note: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)". At the bottom of the form is a green button labeled "Sign up for GitHub". Below the button, there is a small text: "By clicking 'Sign up for GitHub', you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails."



A screenshot of the GitHub sign-in page. At the top, there is the GitHub logo. Below it, the text "Sign in to GitHub" is displayed. In the center, there is a form with two input fields: "Username or email address" (containing the text "nattalides") and "Password" (containing seven dots). To the right of the "Password" field is a link labeled "Forgot password?". Below the form is a green button labeled "Sign in". At the bottom, there is a button labeled "New to GitHub? [Create an account.](#)". An orange arrow points from the "Sign up" button in the previous screenshot to this page.

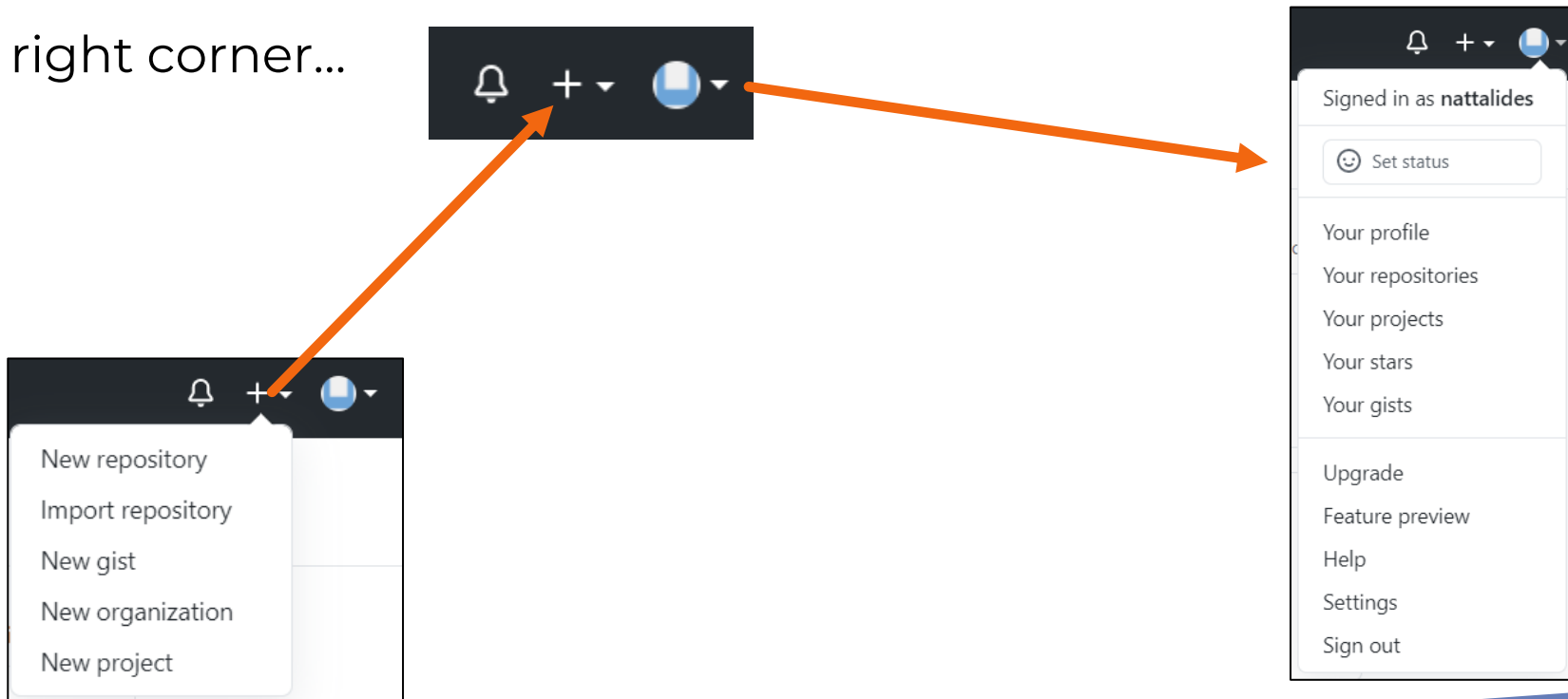


<https://github.com/nattalides>

This is also the URL that others can go to and see all of my public repositories!

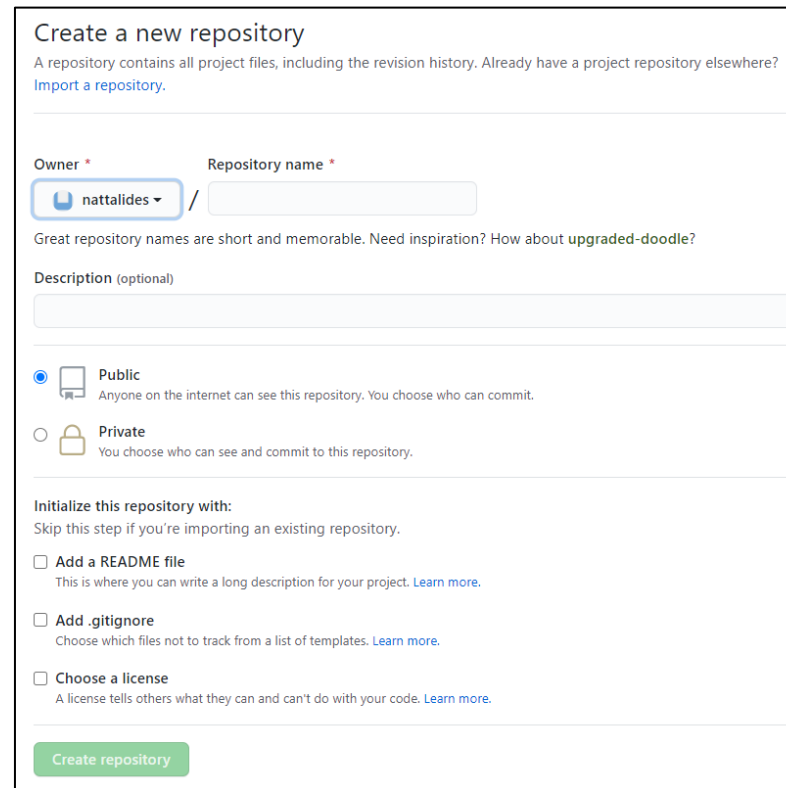
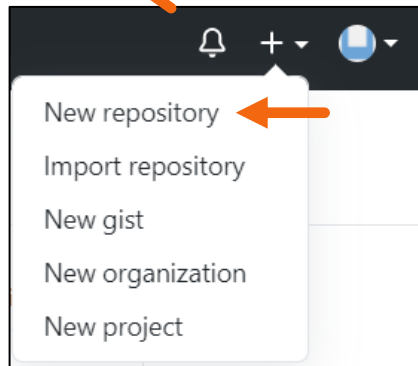
# Create a GitHub repository

**Note:** The homepage might look different for each user depending on recent activity or account overview. The important navigator is found on the top right corner...



# Create a GitHub repository

Select the “+” symbol that we saw earlier and click on “New repository”.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*  / Repository name \*

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

Description (optional)

☒ 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:  
Skip this step if you're importing an existing repository.

☐ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

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

[Create repository](#)




### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---


Owner \*      Repository name \*


 nattalides /

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

Description (optional)

---

☒  **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:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

This is the repository name. This is also the URL address that others can use to go straight into that project.

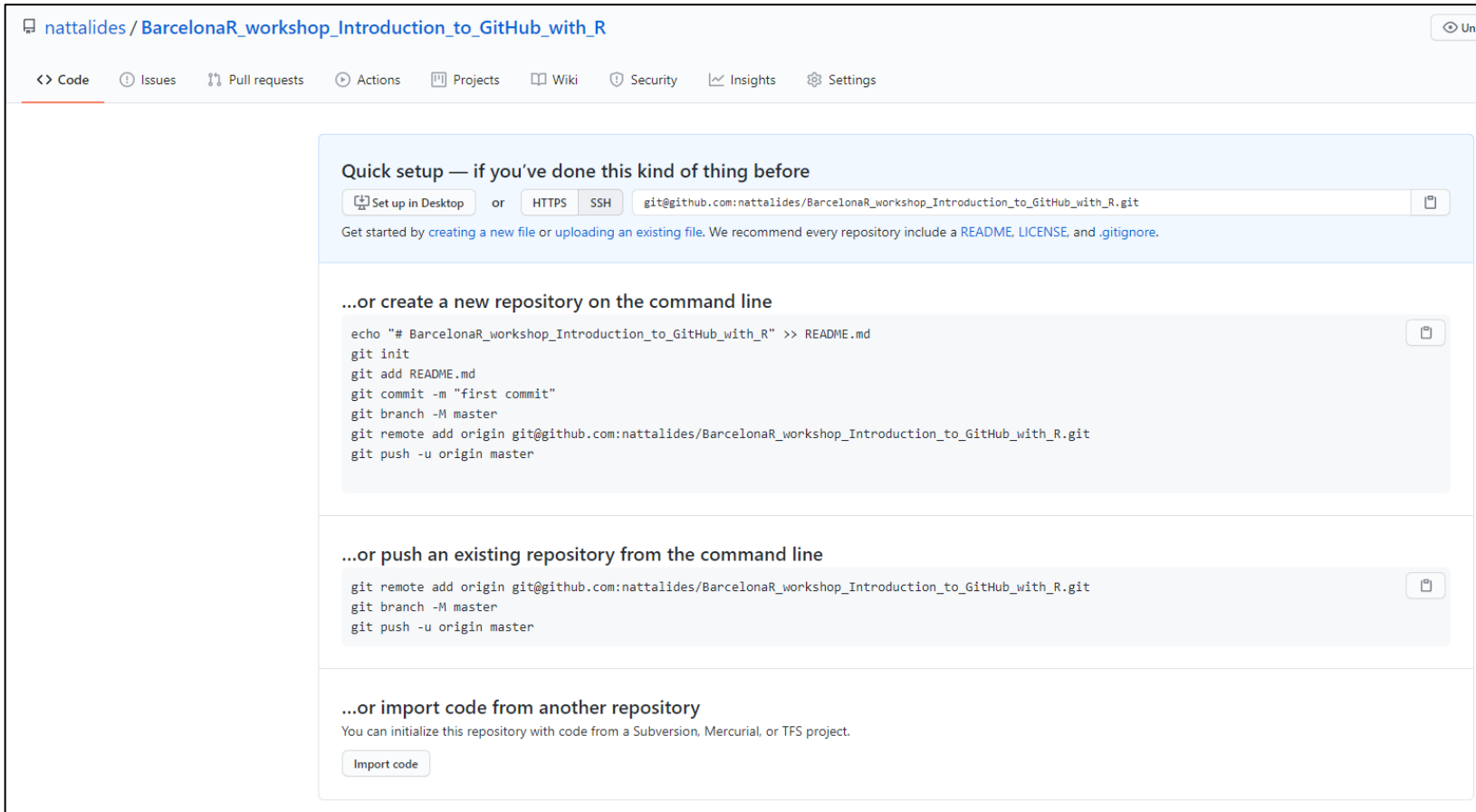
e.g. BarcelonaR\_workshop\_Introduction\_to\_GitHub\_with\_R

Select the repository type, in this workshop we will work with a public repository

It is good practice to add a README file as this is the first thing a new user will get to see/read about your project. In this workshop we leave these options unticked but it is worth coming back to explore these options!



# Congratulations! You have just created a GitHub repository!



The screenshot shows the GitHub interface for a newly created repository. The repository name is 'nattalides / BarcelonaR\_workshop\_Introduction\_to\_GitHub\_with\_R'. The 'Code' tab is selected, showing options for cloning the repository. The 'Quick setup' section provides instructions for cloning via Desktop, HTTPS, or SSH. The '...or create a new repository on the command line' section shows a series of git commands to initialize a new repository, create a README, and push to the master branch. The '...or push an existing repository from the command line' section shows commands to add a remote and push. The '...or import code from another repository' section provides a link to import code from Subversion, Mercurial, or TFS.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# BarcelonaR_workshop_Introduction_to_GitHub_with_R" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git
git branch -M master
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code



# Create a GitHub repository

**Quick setup — if you've done this kind of thing before**

📄 Set up in Desktop

 or 

HTTPS

SSH

git@github.com:nattalides/BarcelonaR\_workshop\_Introduction\_to\_GitHub\_with\_R.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# BarcelonaR_workshop_Introduction_to_GitHub_with_R" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git
git push -u origin master
```

**...or push an existing repository from the command line**

```
git remote add origin git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git
git branch -M master
git push -u origin master
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

This is important...  
you will see later on  
why

# Connect repository with R project

In order to link up our GitHub account with RStudio (desktop) we need to establish a secure connection. This is done by creating and using an **SSH key** and it only needs to be done once.

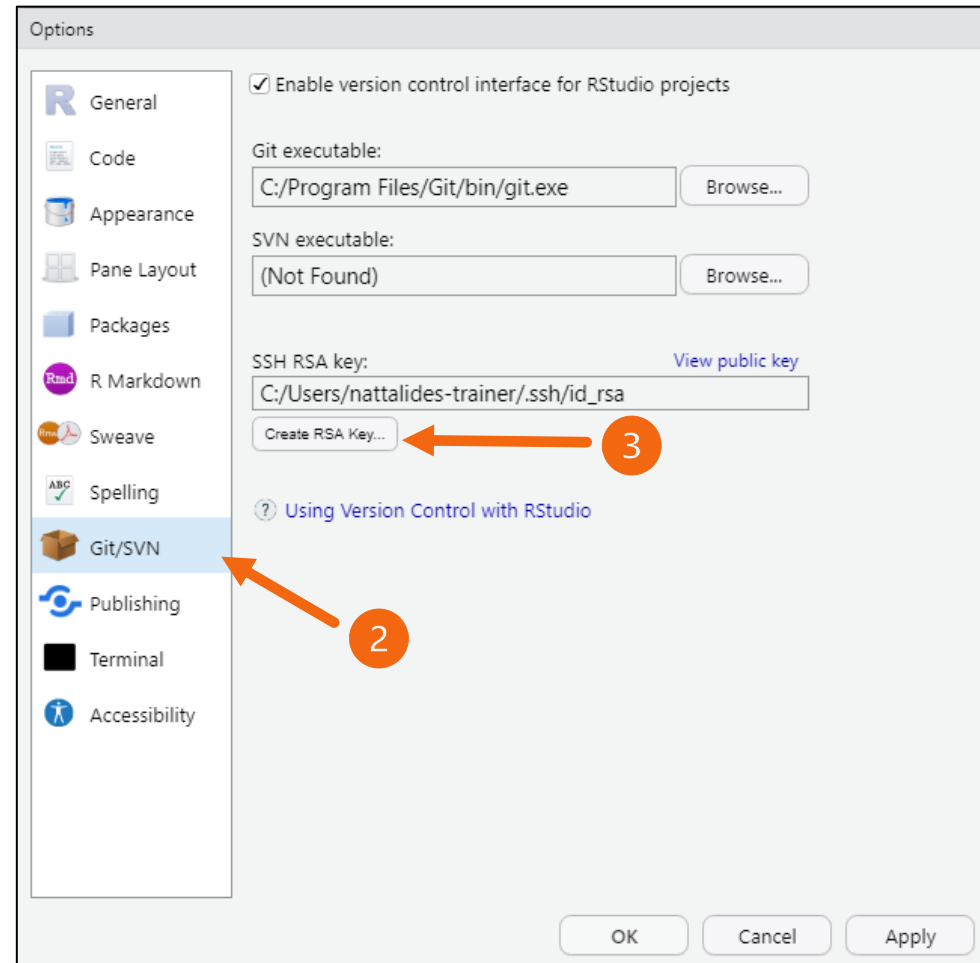
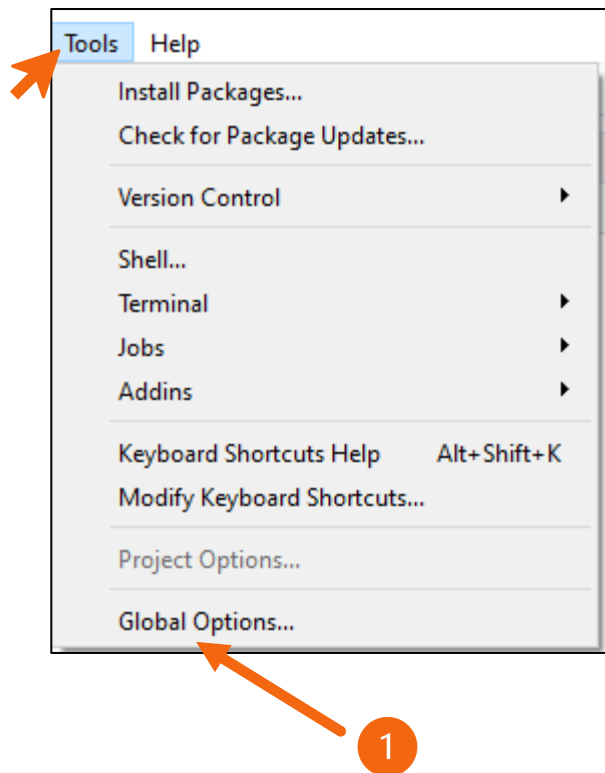
SSH stands for Secure Shell and is a “cryptographic network protocol for operating network services securely over an unsecured network”.

- ▶ **Create an SSH key from your RStudio desktop**
- ▶ **Associate the SSH key with your GitHub account**
- ▶ **Clone a GitHub repository to an R project**

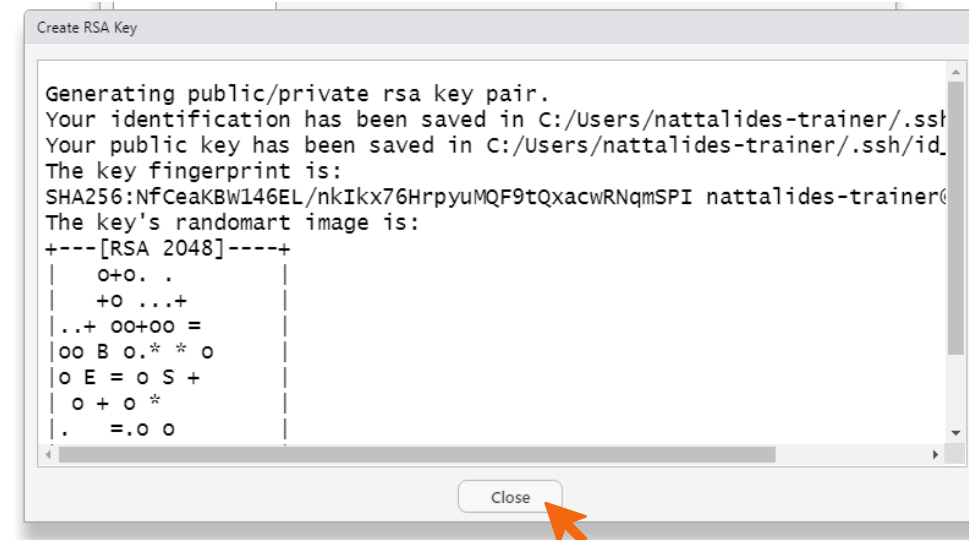
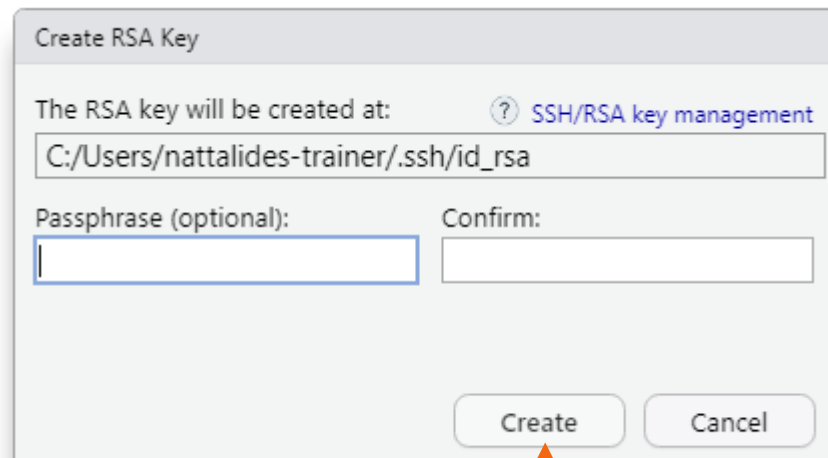


For security **do not** share with others your SSH key

# Create an SSH key from your RStudio desktop

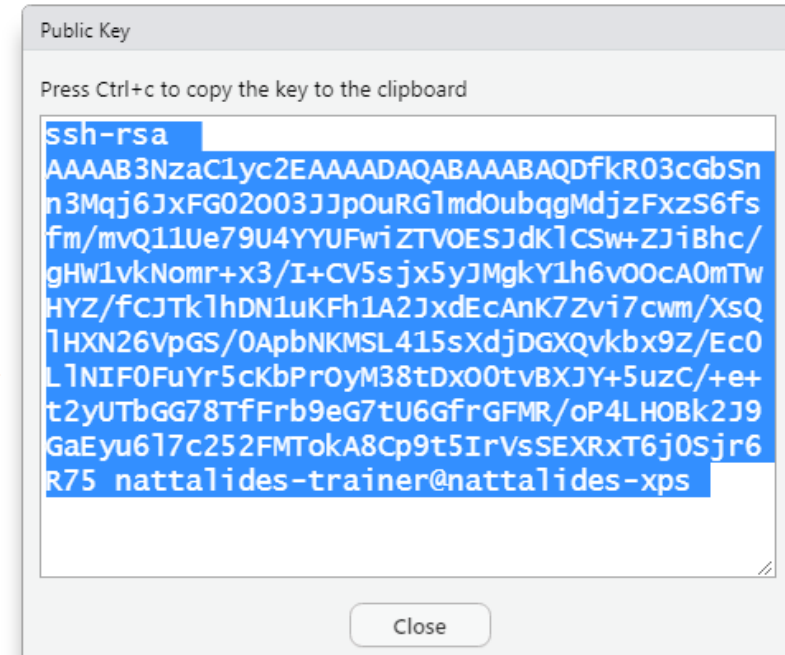
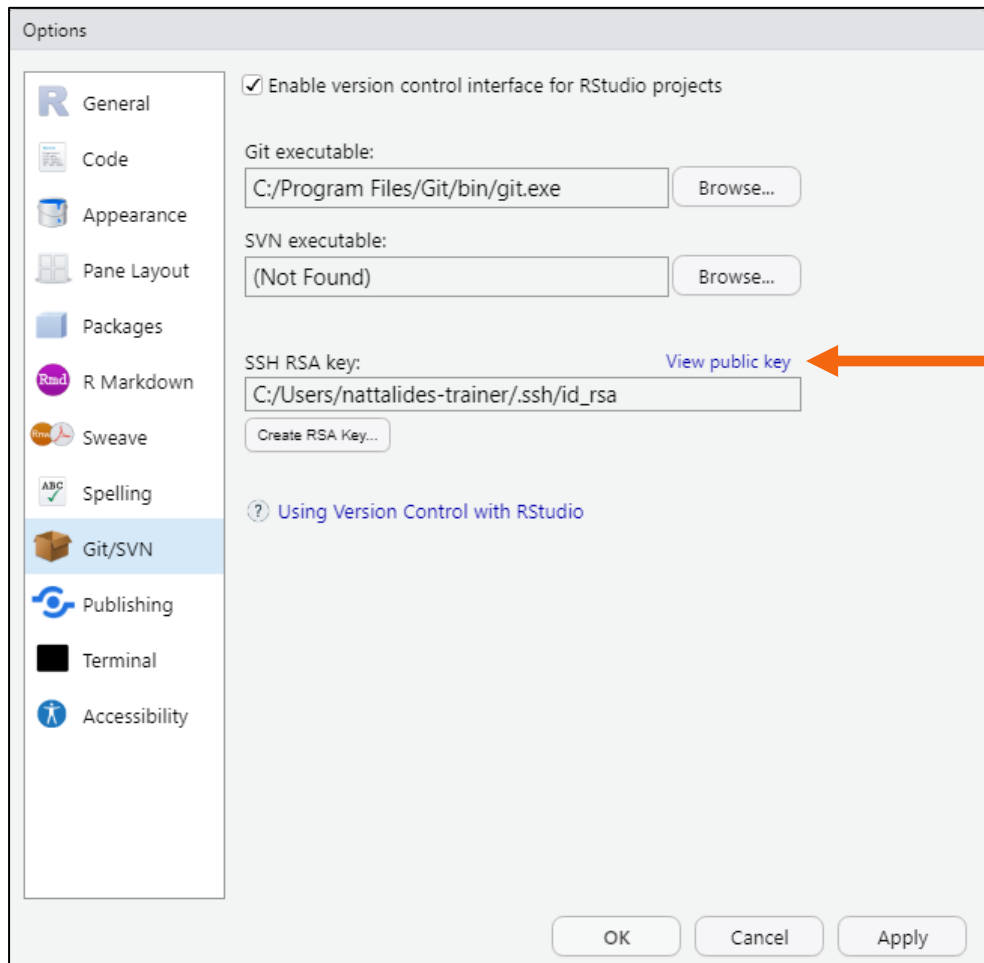


# Create an SSH key from your RStudio desktop



A passphrase is like a password that you will be prompted to supply. For simplicity we will leave this blank.

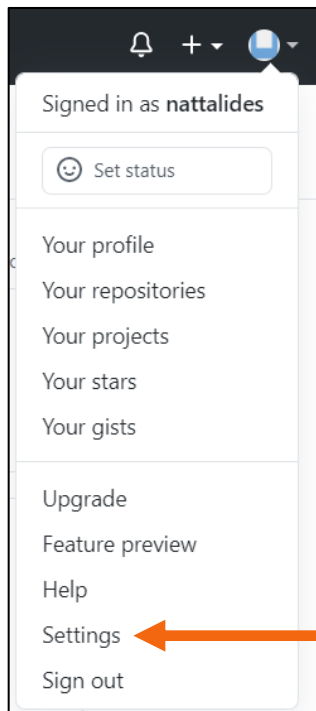
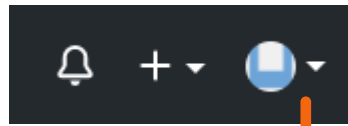
# Create an SSH key from your RStudio desktop



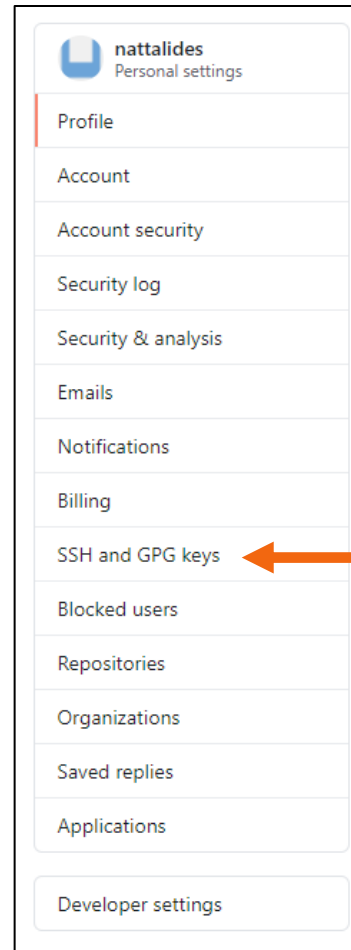
6

Press Ctrl + c to copy the key to the clipboard

# Associate the SSH key with your GitHub account



7



8

New SSH key

9

# Associate the SSH key with your GitHub account

SSH keys / Add new

Title

Key

Begins with 'ssh-rsa', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', or 'ecdsa-sha2-nistp521'

Add SSH key

Typically SSH keys are linked to a specific machine, so good practice is to give this key a title to reflect that, for example, "Nicolas-Laptop"

Paste here the SSH key that you copied earlier from RStudio

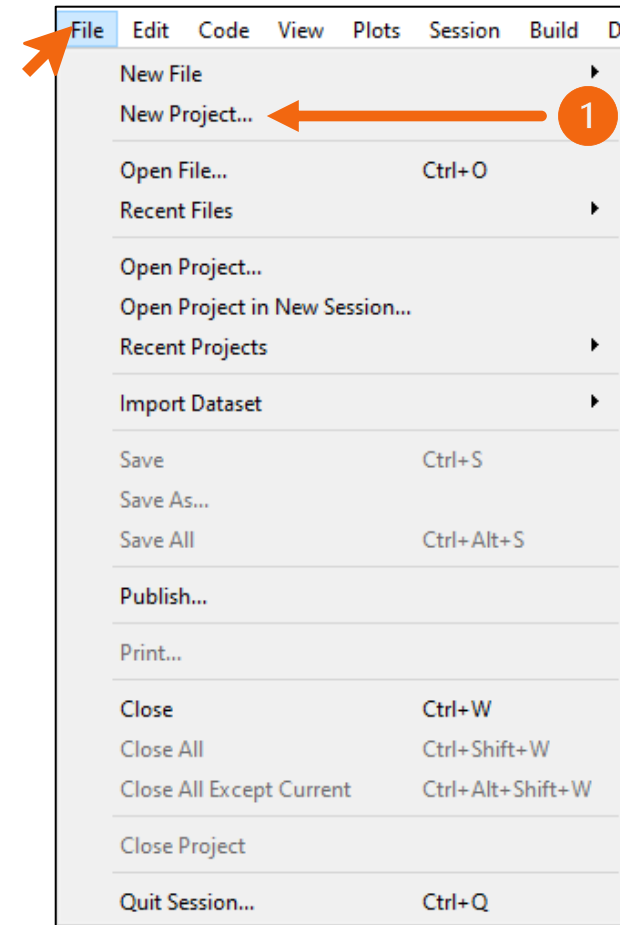
10



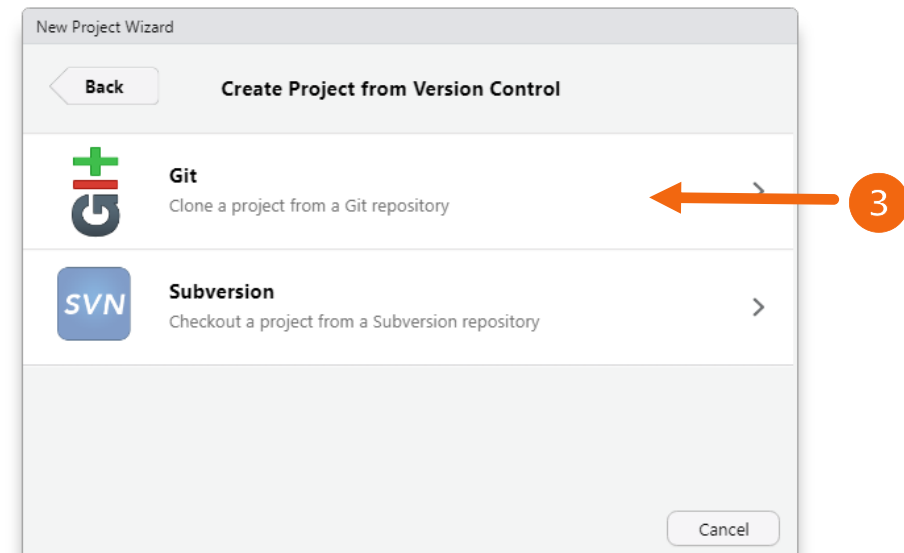
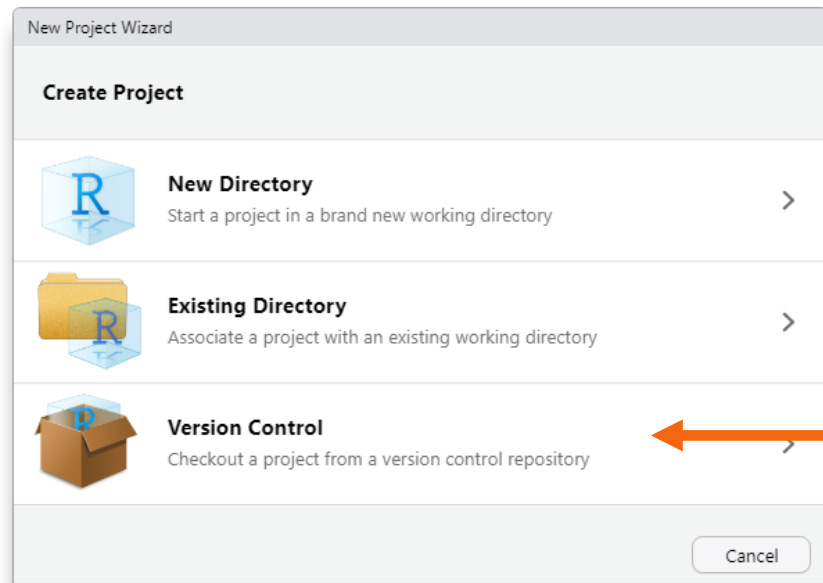
# Clone a GitHub repository to an R project

It is a good idea to learn how to write code within R projects. This is because by using R projects you can easily divide your work in a structured way. Each project has its own setup such as a working directory, workspace and history.

This is even better and more efficient when it is **version controlled!**

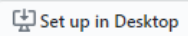



# Clone a GitHub repository to an R project



# Clone a GitHub repository to an R project


Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** `git@github.com:nattalides/BarcelonaR_workshop_Introduction_to_GitHub_with_R.git` 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

New Project Wizard

**Back** **Clone Git Repository**



Repository URL:

Project directory name:

Create project as subdirectory of:  
 **Browse...**


☐ Open in new session **Create Project** **Cancel**

4

Copy the URL path to clipboard and paste to Repository URL

New Project Wizard

**Back** **Clone Git Repository**



Repository URL:

Project directory name:

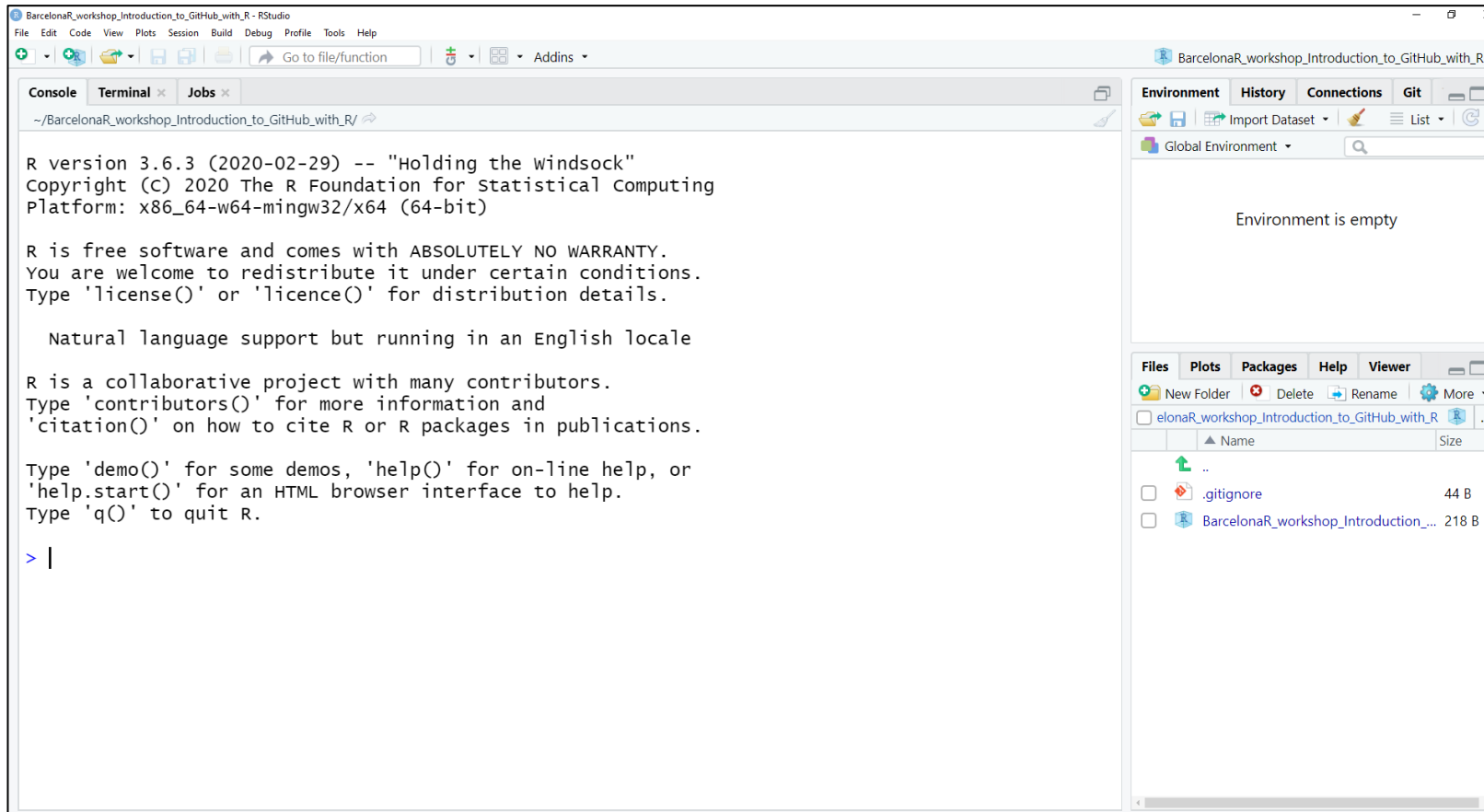
Create project as subdirectory of:  
 **Browse...**

☐ Open in new session **Create Project** **Cancel**

5

You can edit the project directory name and the folder where it will live on your machine

Congratulations! You have just connected a GitHub repository with an R project!

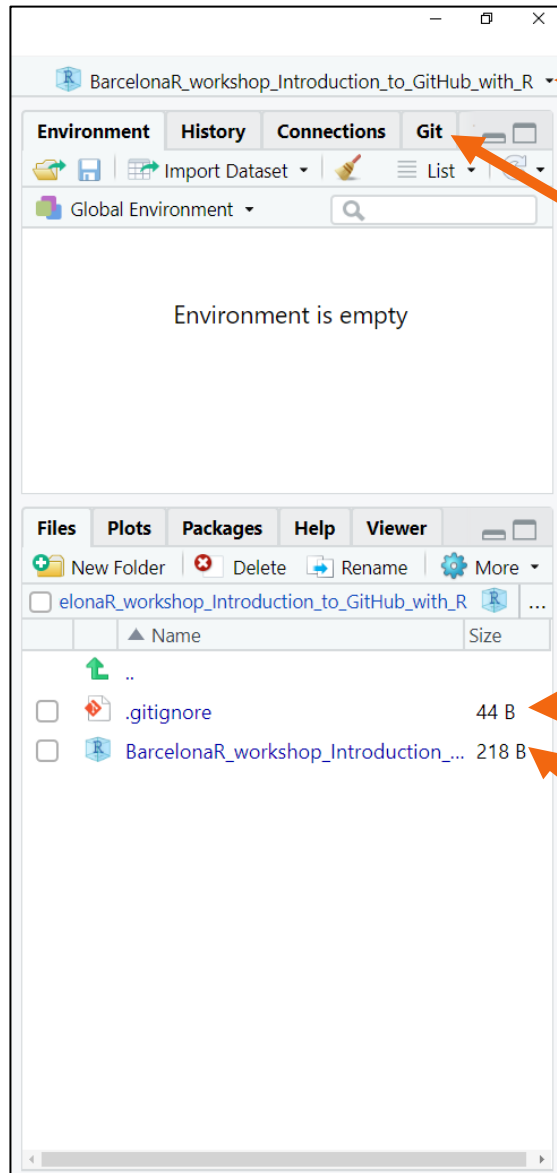


The screenshot shows the RStudio interface with the following components:

- Console:** Displays the R version (3.6.3), copyright (© 2020 The R Foundation), platform (x86\_64-w64-mingw32/x64), and welcome message. It also shows the current directory: `~/BarcelonaR_workshop_Introduction_to_GitHub_with_R/`.
- Environment:** Shows "Global Environment" and "Environment is empty".
- Files:** Lists the following files:

Name	Size
..	
.gitignore	44 B
BarcelonaR_workshop_Introduction...	218 B





This tells us that we are now inside the R project:  
"BarcelonaR\_workshop\_Introduction\_to\_GitHub\_with\_R"

This is the "new" tab that should appear that tells you that your R project has added Git functionality, i.e. it is version controlled. This tab will be very useful as we develop and version control our code!

This is a very useful (hidden) file that communicates to our GitHub repository what files/folders should be **ignored**... these files/folders **will not** be version controlled! More on this later.

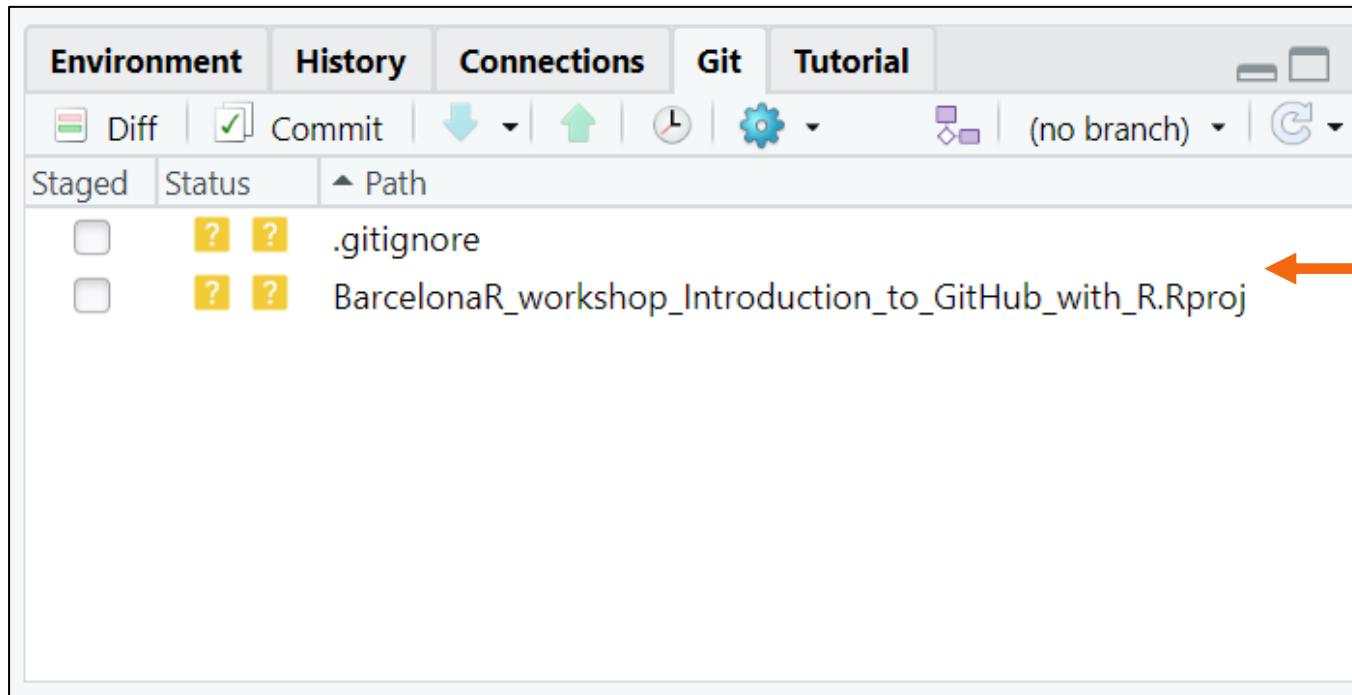
This is the .Rproj file associated with this project

# Basic version control processes

Now that our project is version controlled let's cover some of the basic processes to version control a file, keeping things as simple as possible. A file in our repository can be one of four statuses:

1. **Untracked:** The file is not being tracked, i.e. it is not version controlled
2. **Unmodified:** The file is tracked but has not changed
3. **Modified:** The file is tracked and has been changed
4. **Staged:** The file is tracked and the changes are ready to be version controlled

# Basic version control processes



These files are currently untracked

# Basic version control processes

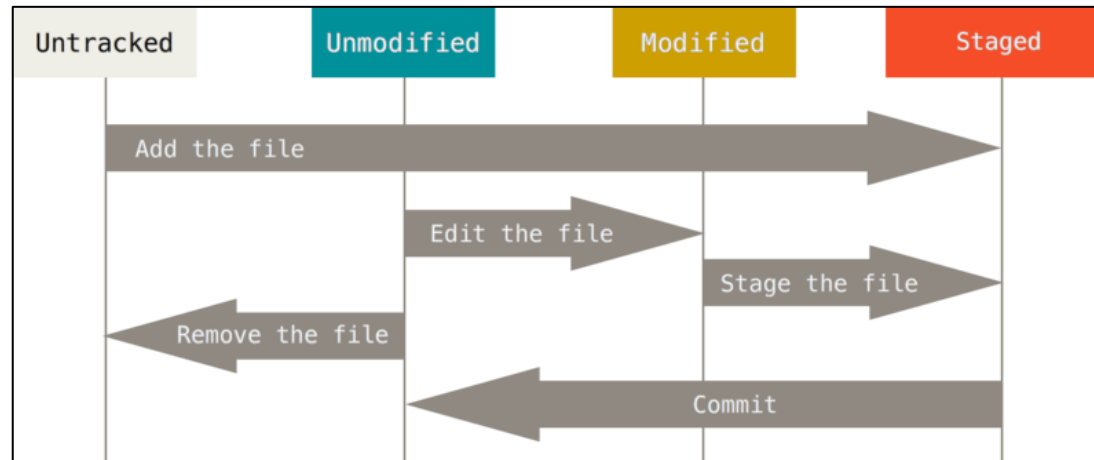


Image from: <https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

**Untracked File:** We need to add the file to the “staging area”

**Modified File:** We need to stage (add) the file to the “staging area”

```
git add [file]
```

This is the command to add the file to the “staging area”.



# Basic version control processes

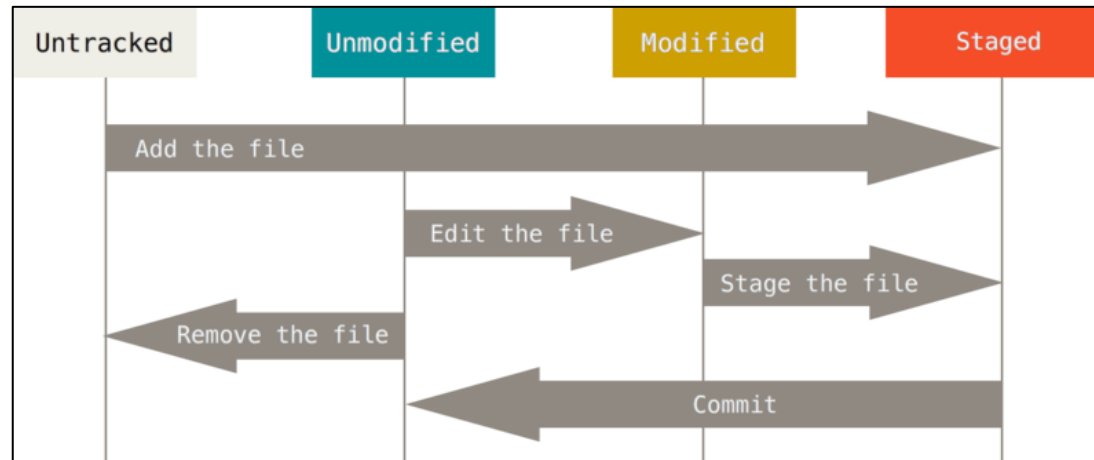


Image from: <https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

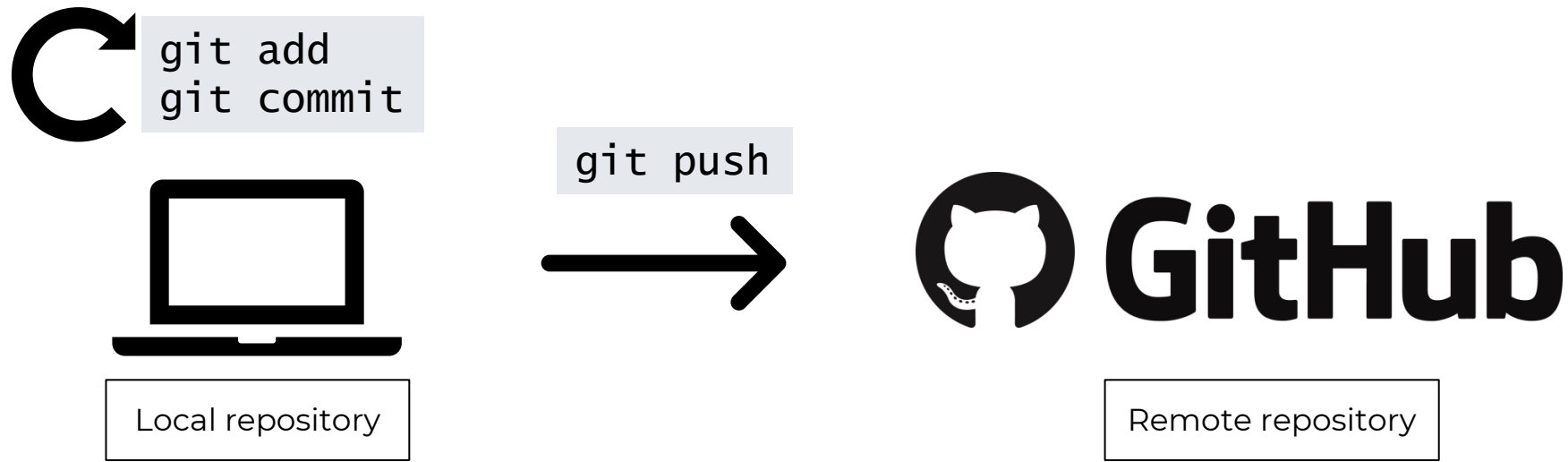
**Staged File:** We need to commit the file and all of its changes with some useful documentation or comment about the changes.

```
git commit -m "your_comment_goes_here" [file]
```

This is the command to commit the file from the “staging area”.

## Basic version control processes

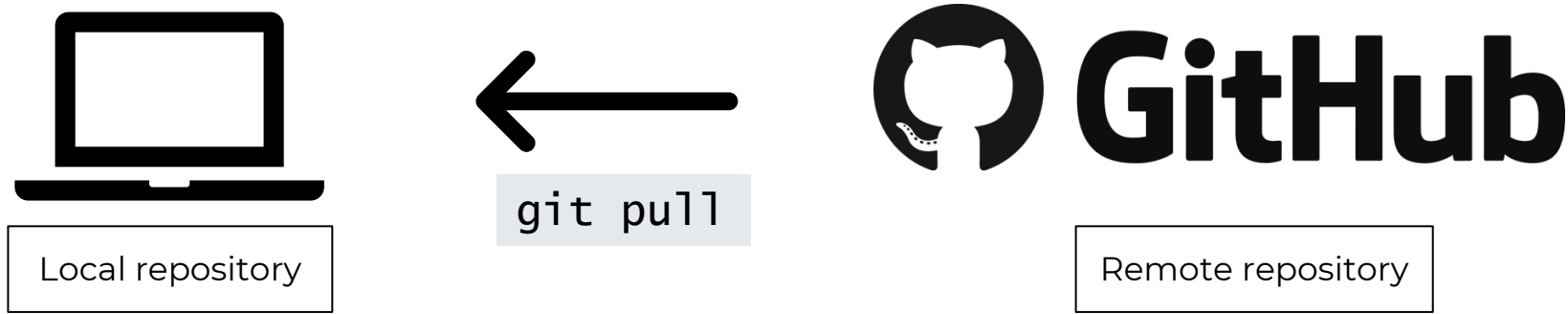
A file has been added to the staging area and the changes have been committed. This is “stored” locally (local repository) so the next step is to **push** these changes to our remote (online) repository in GitHub. Now the file has been version controlled!



## Basic version control processes

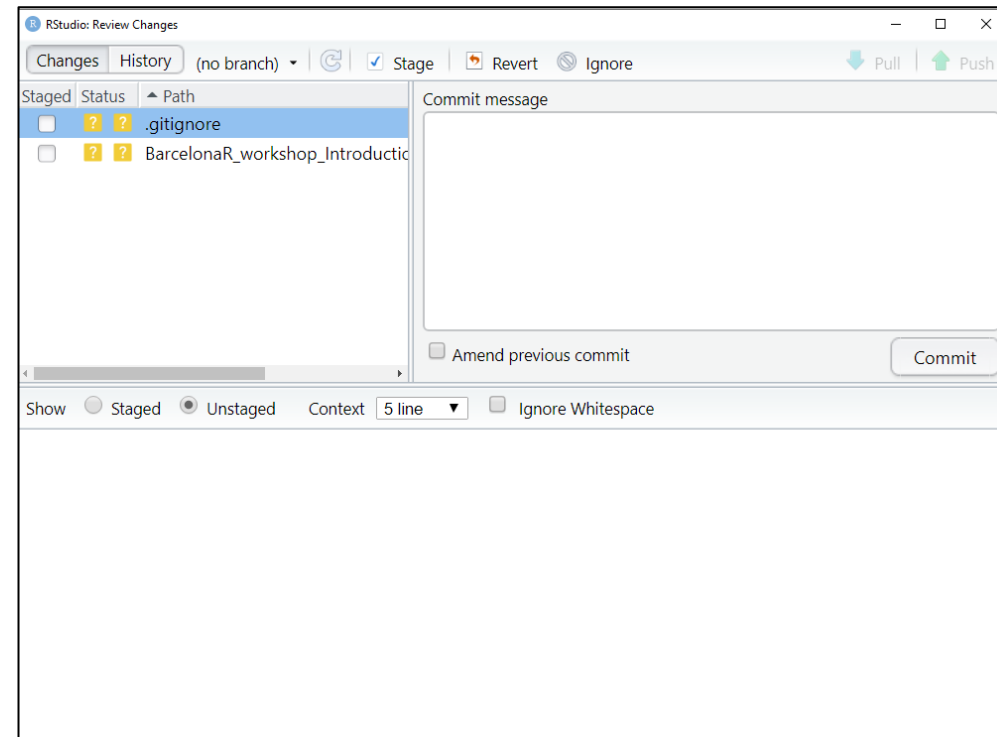
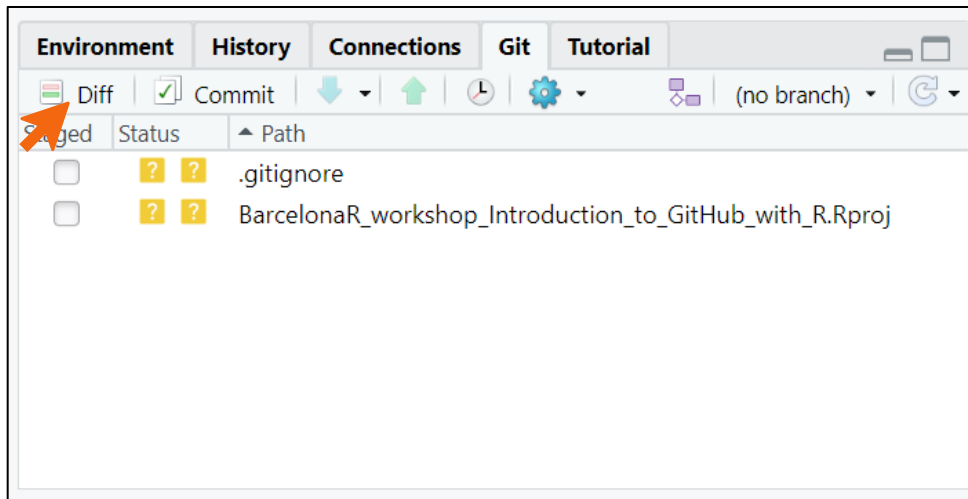
Let's say a collaborator has a copy of the up-to-date remote repository on his/her machine and decides to add a new R script to our project. They follow the steps of *adding*, *committing* and *pushing* the file to the remote repository.

- ▶ Our local repository is now “behind” the current version of the project so we need to **pull** the changes (i.e. the new file) into our local repository.

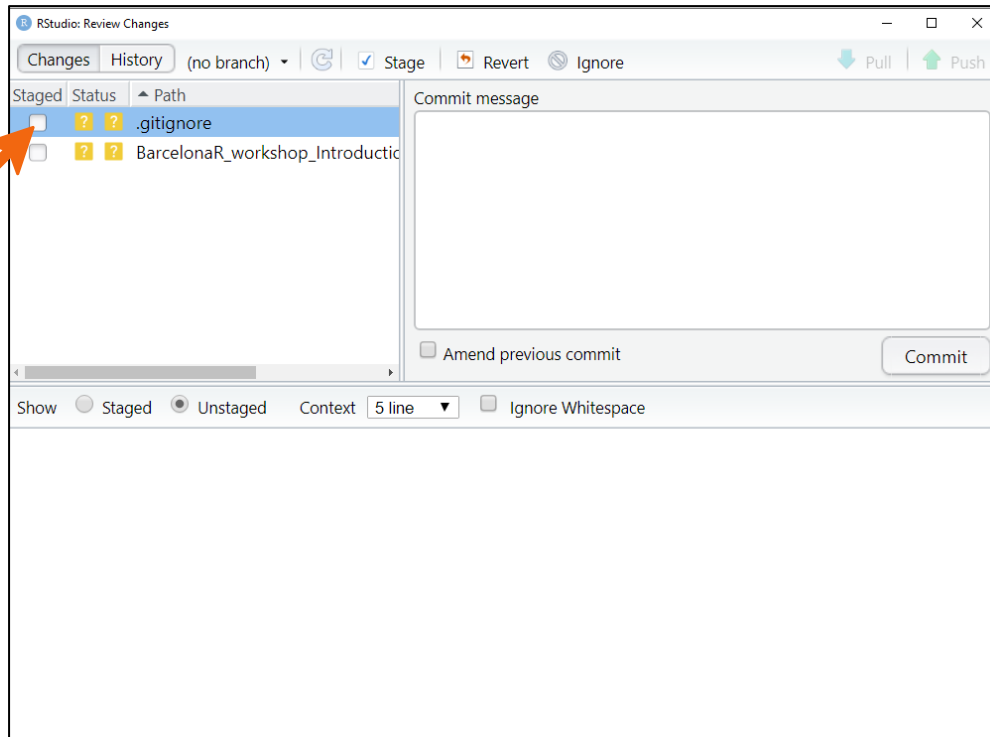


# Basic version control processes

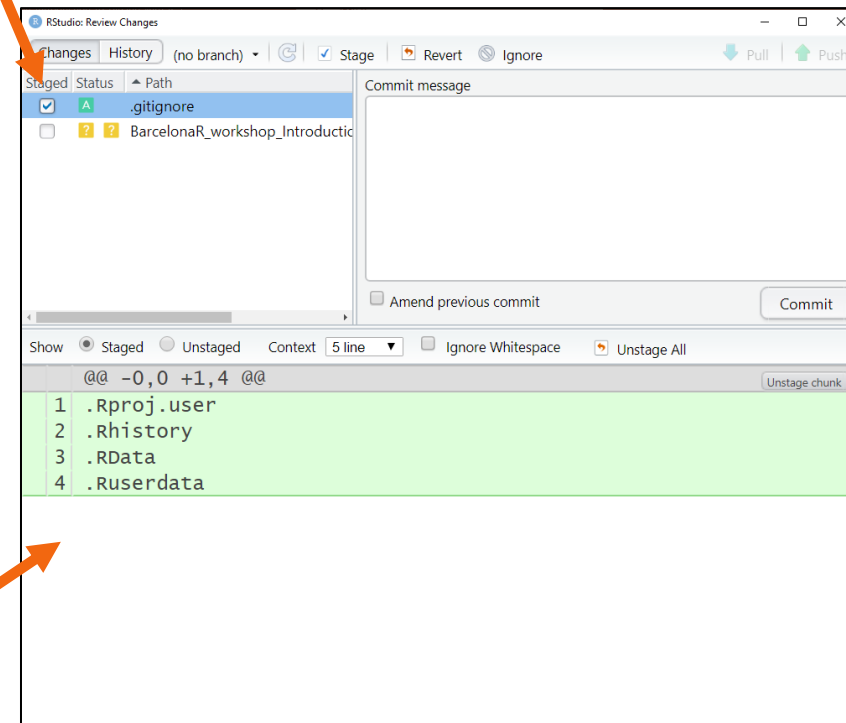
We can easily do all these version control actions in RStudio using the interactive git tab window...



# Basic version control processes

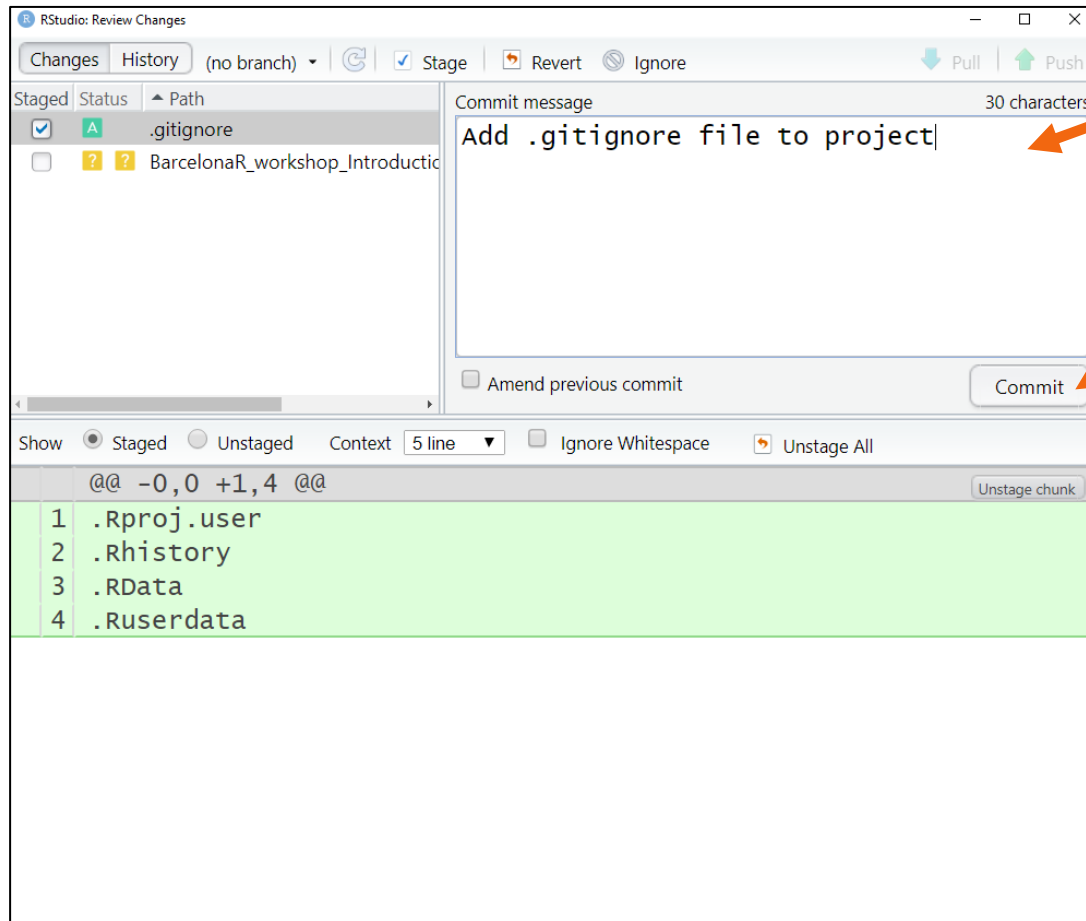


git add



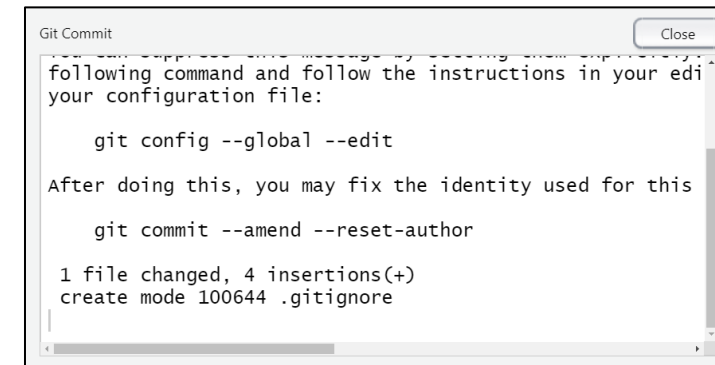
This tells us what are the changes  
(green highlight is added changes,  
red highlight is deleted changes)

# Basic version control processes



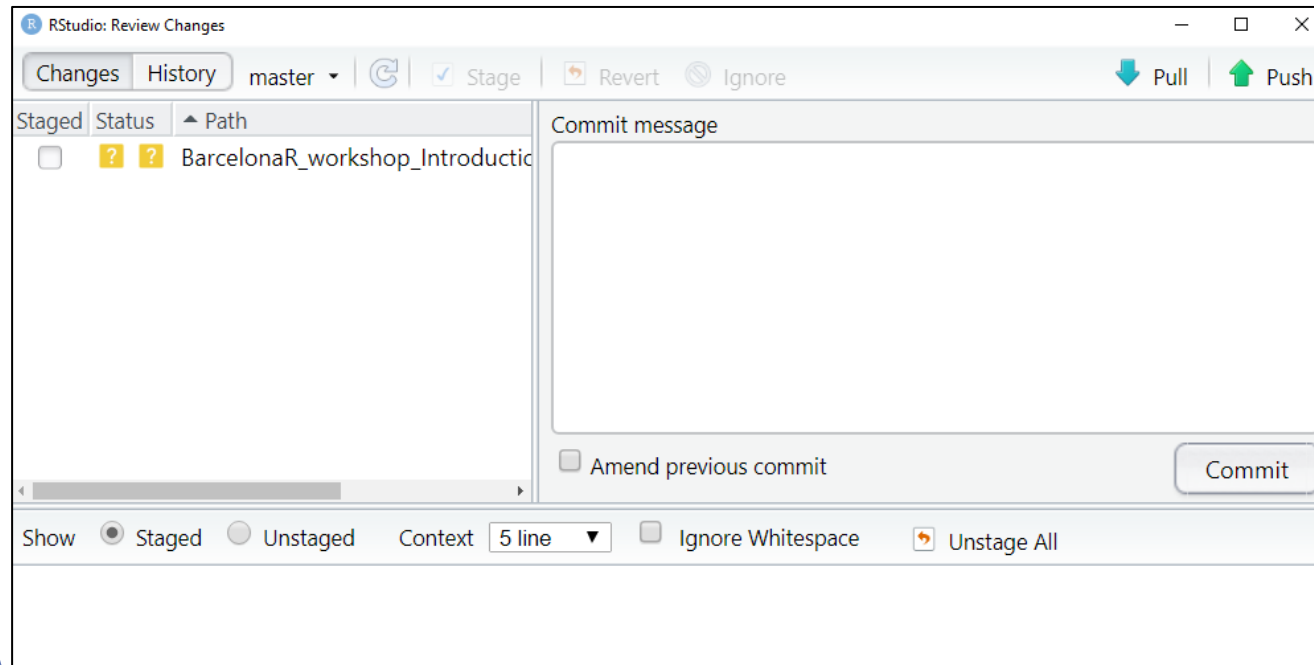
Insert the commit message  
for the file to be staged

`git commit`



Some information about your commit

# Basic version control processes



git push

We can now push the staged changes from our local to our remote repository

```
Git Push
Close

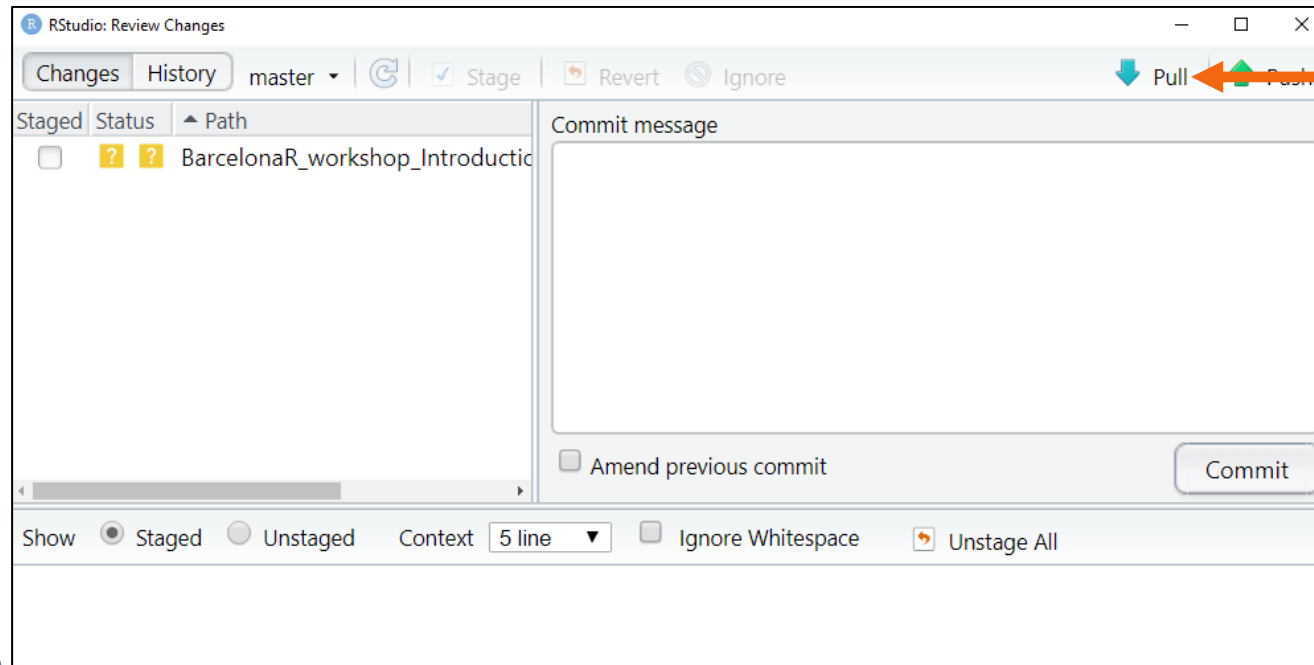
>>> C:/Program Files/Git/bin/git.exe push origin HEAD:refs/heads/master
Warning: Permanently added the RSA host key for IP address 192.168.1.1.
To github.com:nattalides/BarcelonaR_workshop_Introduction_h_R.git
 * [new branch]      HEAD -> master
```

Some information about your push.



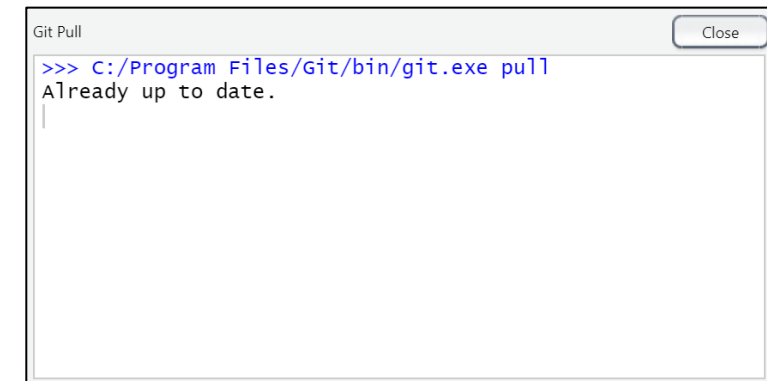
A **master branch** has been created

# Basic version control processes



```
git pull
```

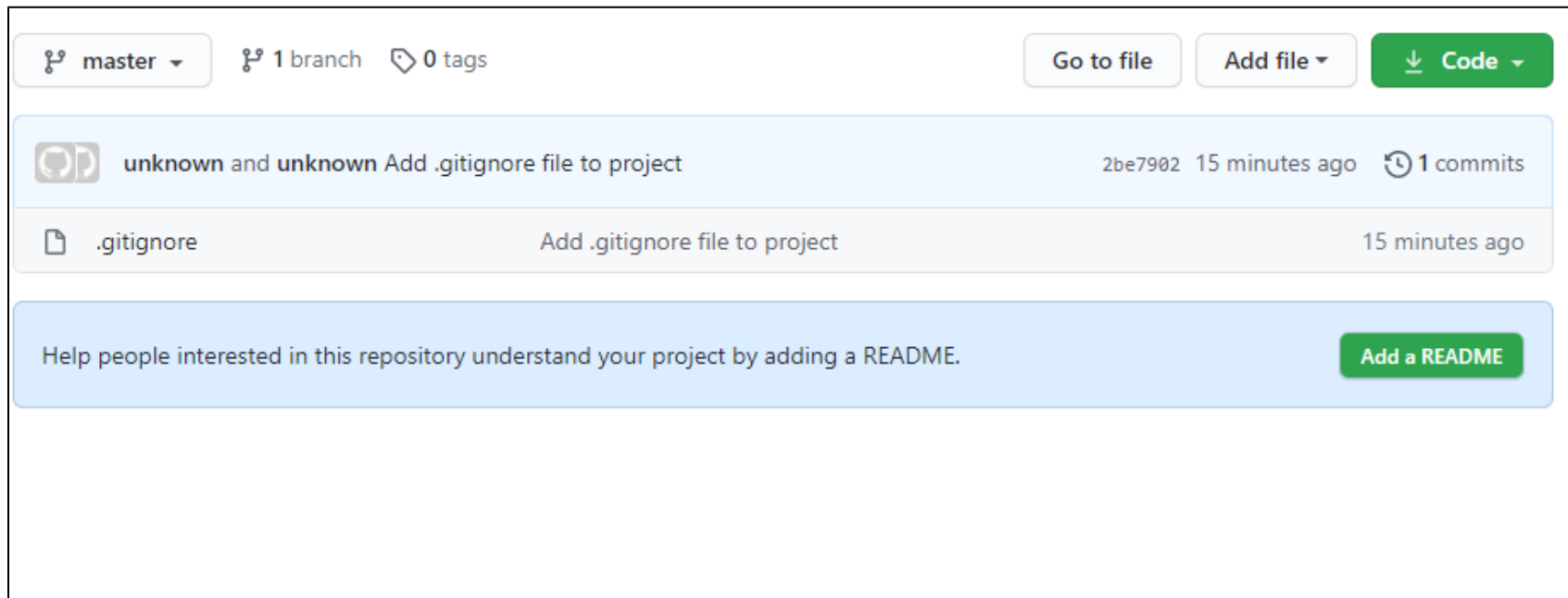
We can now pull any new changes from our remote to our local repository



Some information about your pull.



Congratulations! You have just added, committed and pushed a file into a GitHub repository within an R project in RStudio desktop!



## Some further notes

During this workshop we have seen the basic version control actions. Using Git (and GitHub) can be a little frustrating at first but after a while you will get used to it and there is a definite reward to all of this hard work!

- ▶ There is a lot of useful resources online if you get stuck!
- ▶ Working in branches (we have not explored this) can improve the workflow and avoid headaches.
- ▶ Any sensitive (e.g. passwords) or large files (e.g. data) should not be version controlled... use the .gitignore file to “tell” Git to ignore these.
- ▶ A very good resource on Git and GitHub can be found at:  
<https://happygitwithr.com/>

Thank you to our sponsors and partners!



## QUIZ TIME



<https://ahaslides.com/GITHUB>