git_tutorial_notes

Getting started with github, git and R-studio.

Creating a github account

Go to the github website

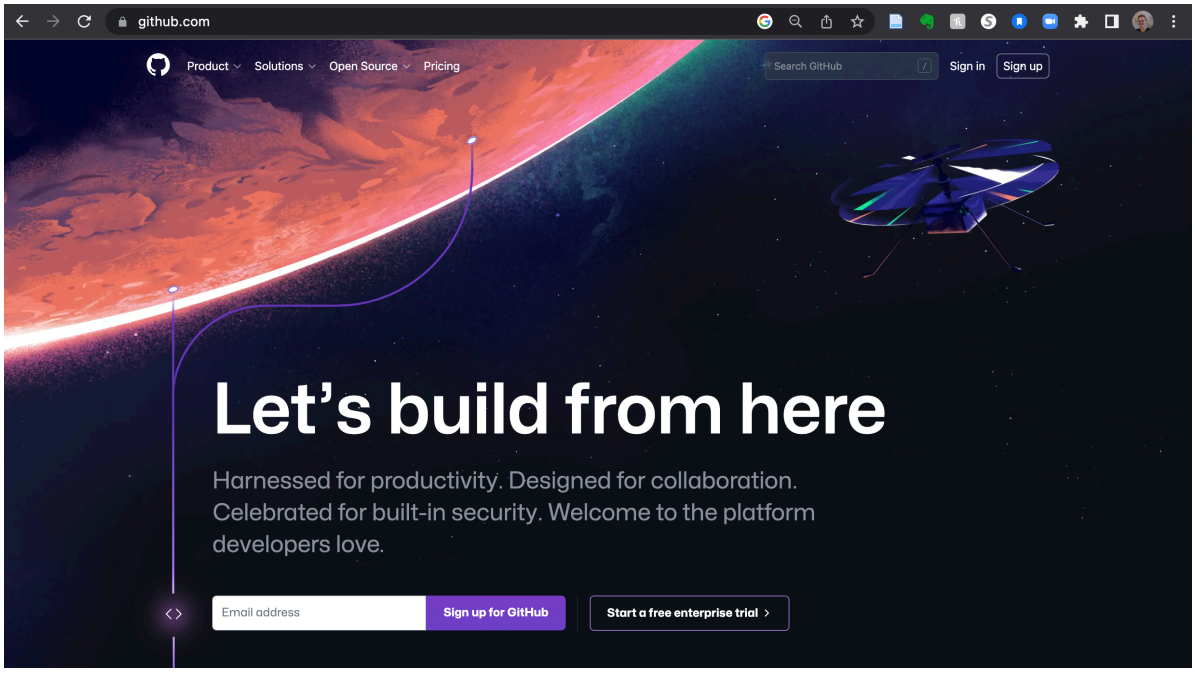


Figure 1: Github website

and signup with your email and password.

Once you have a github site, in the upper left hand corner on your github site create a repository for where you want files from your pending project to go. Name the repository whatever you

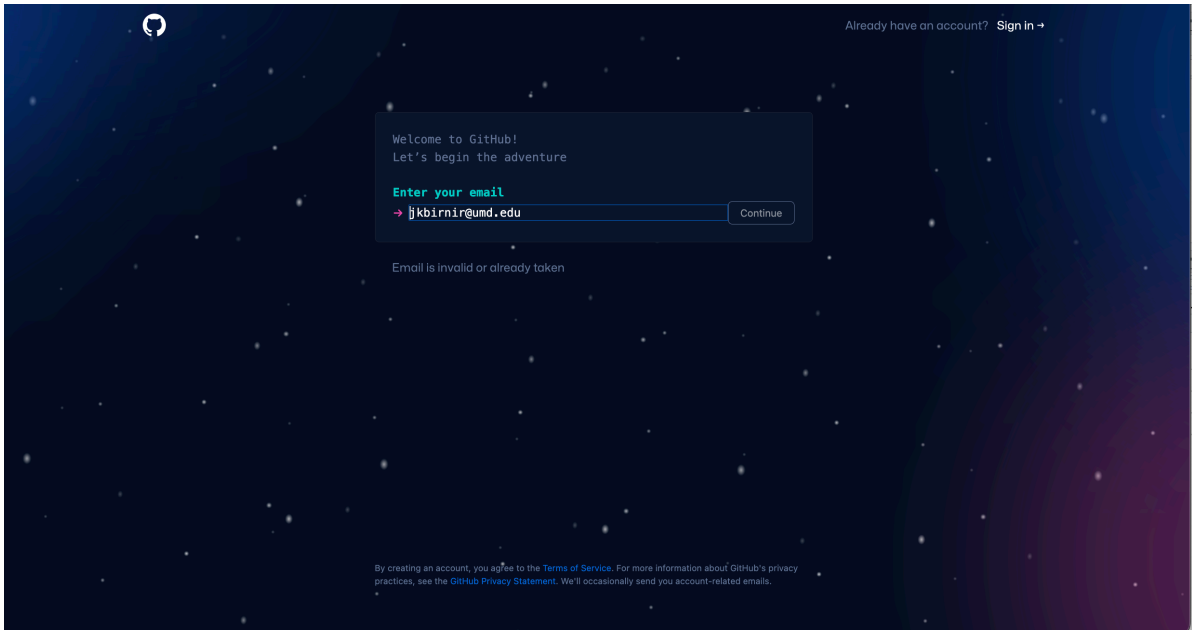


Figure 2: Github signup

will be calling your project. Select to have a readme file where you can post notes about the project. Select to make it private while you are working on int (this can be changed later).

Once you hit create you new repository should look something like this:

Getting started with git on your computer

Check if you have git installed on your computer. For this you have to use terminal to check. In your terminal write

git –version

and you should get back your version number.

If you do not have git on your computer you may have to install it. For instructions on how to check and install git on your computer see this very helpful website or better yet this manual

Setting up your project in R studio

Now that you have a github account and a local git on your computer you are ready to start working with git through R studio.

Open R-studio and start a new project. Choose a project with version control:

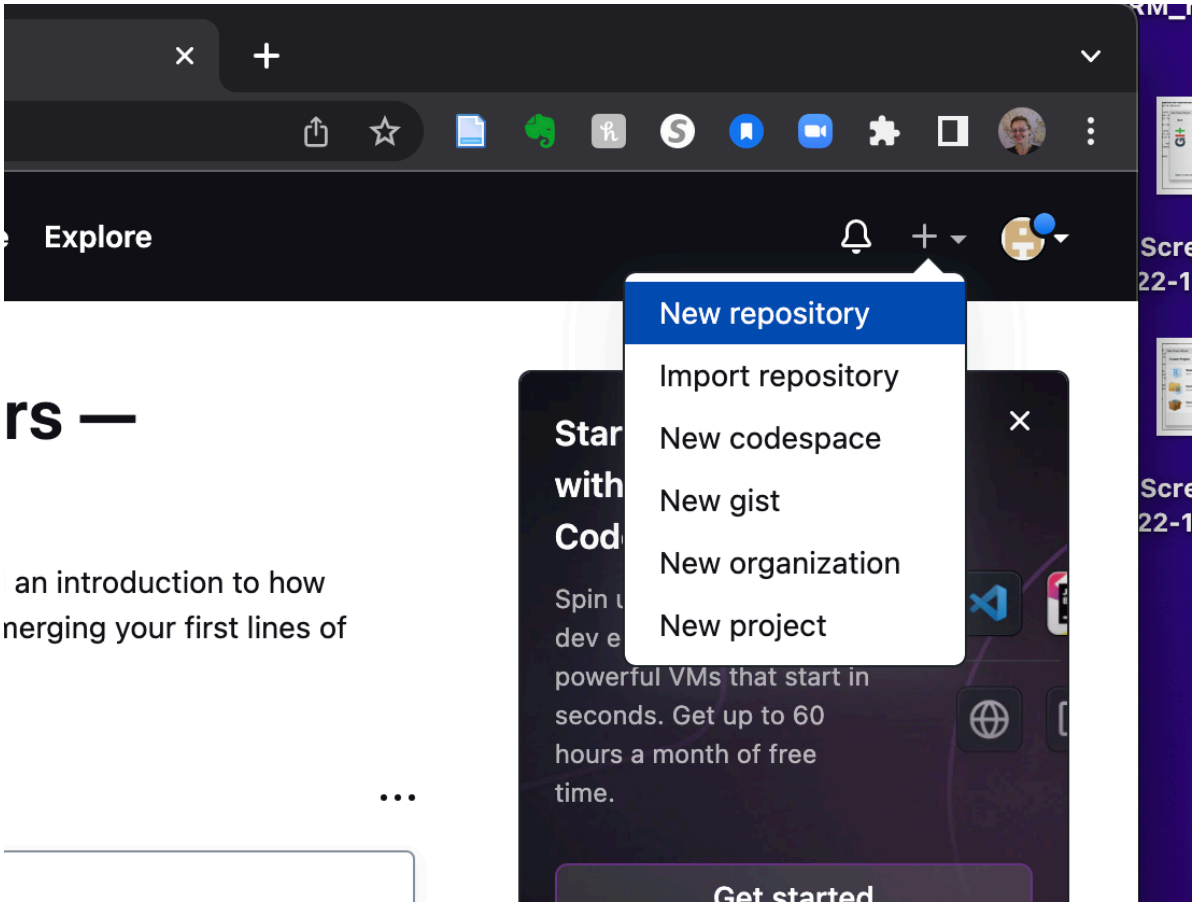


Figure 3: Github new repository

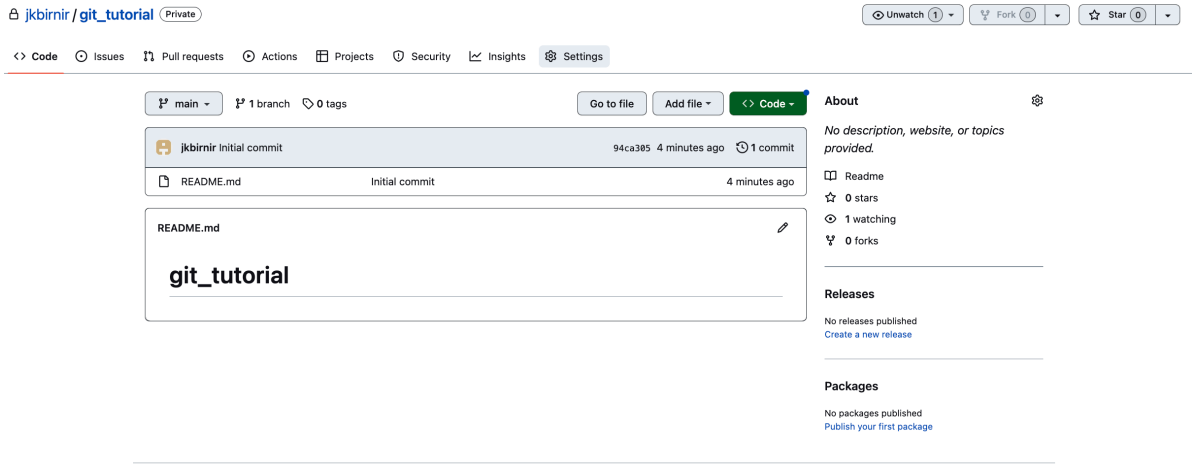


Figure 4: terminal git version

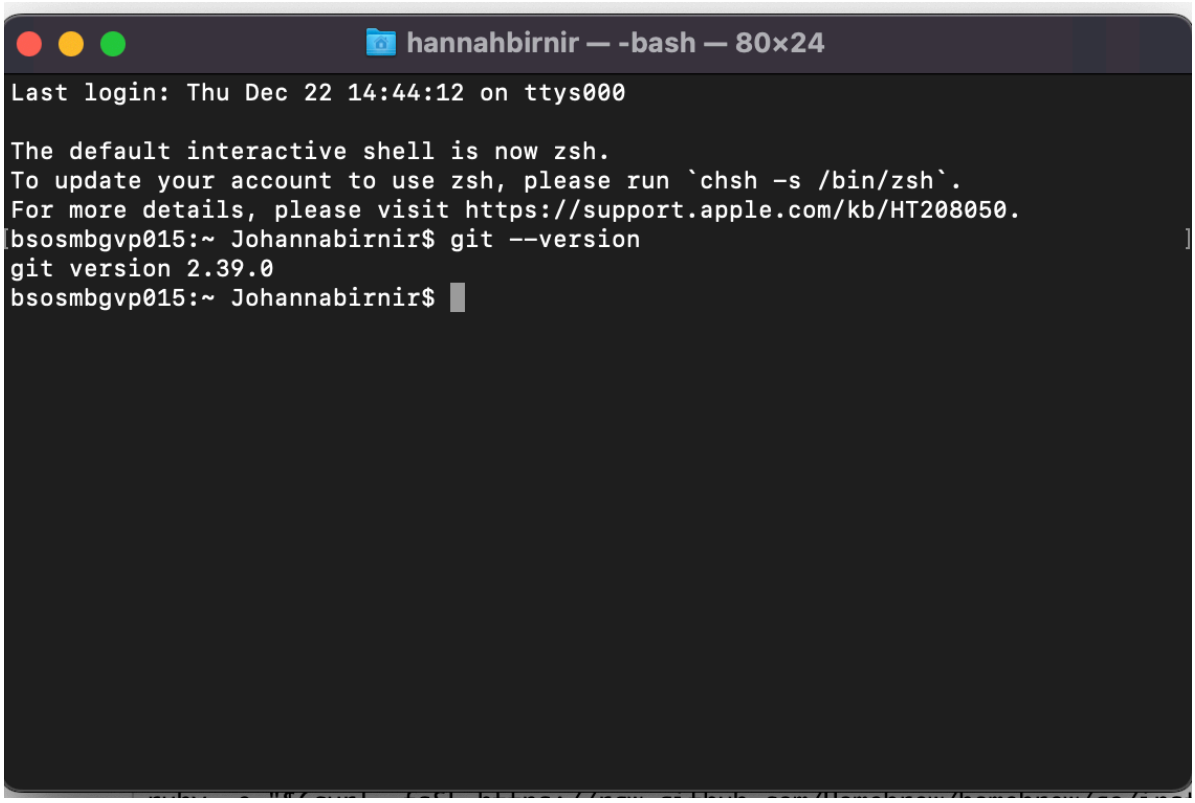


Figure 5: terminal git version

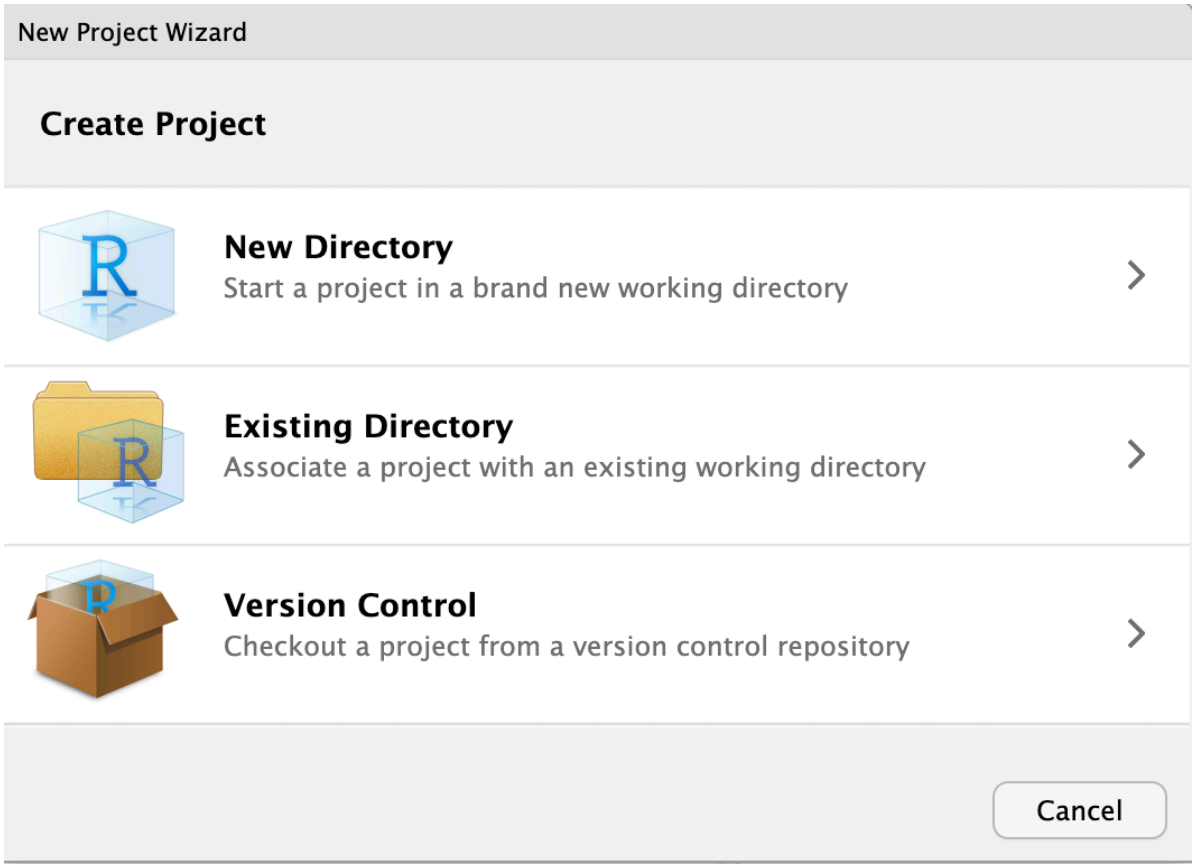


Figure 6: Creating a version control project

Select the option to clone a repository from git. What R studio will then do is to clone the repository you created on github locally on your computer.

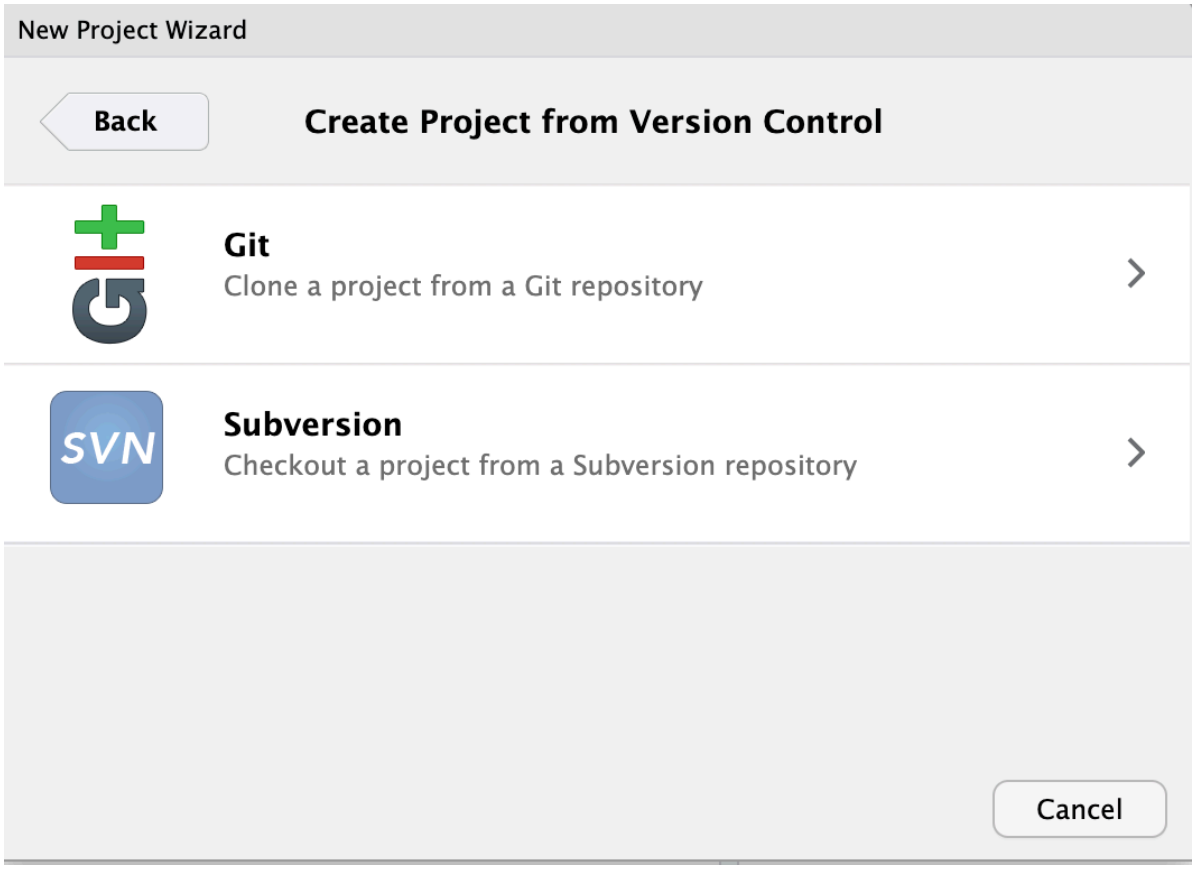


Figure 7: Cloning the git repository

So that R-studio knows which github local repository to clone you have to specify an external url that matches your username and the name of the new repository that you just created on github (Repository URL).

You also have to specify where your local git files are going to be located see (Create project as subdirectory of:).

Hit the create button and R studio will create a project site that should look something like this. Notice how the files replicate what is in

In case you run into trouble at this stage - and are not able to connect your files make sure that your local credentials (signup email matches the email you used to signup with github). To check this you can use:

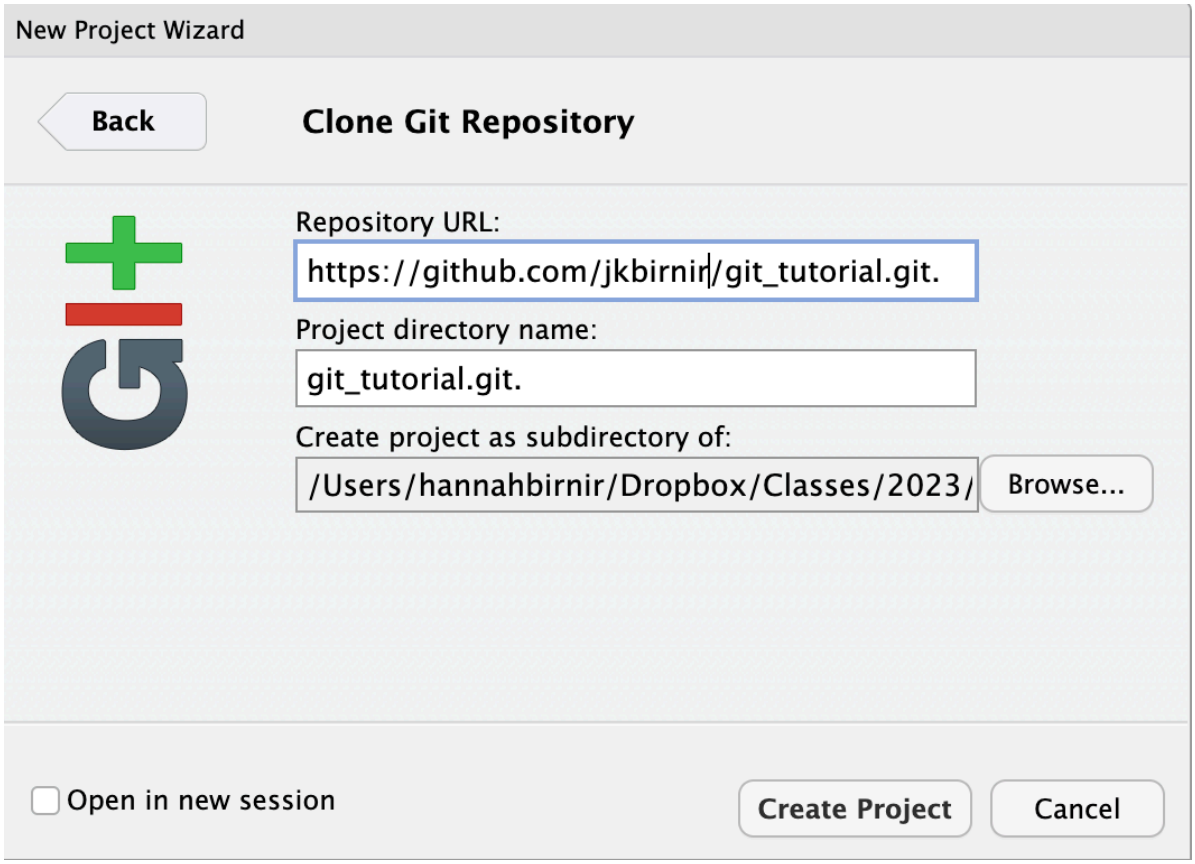


Figure 8: Specifying location of external and local repositories

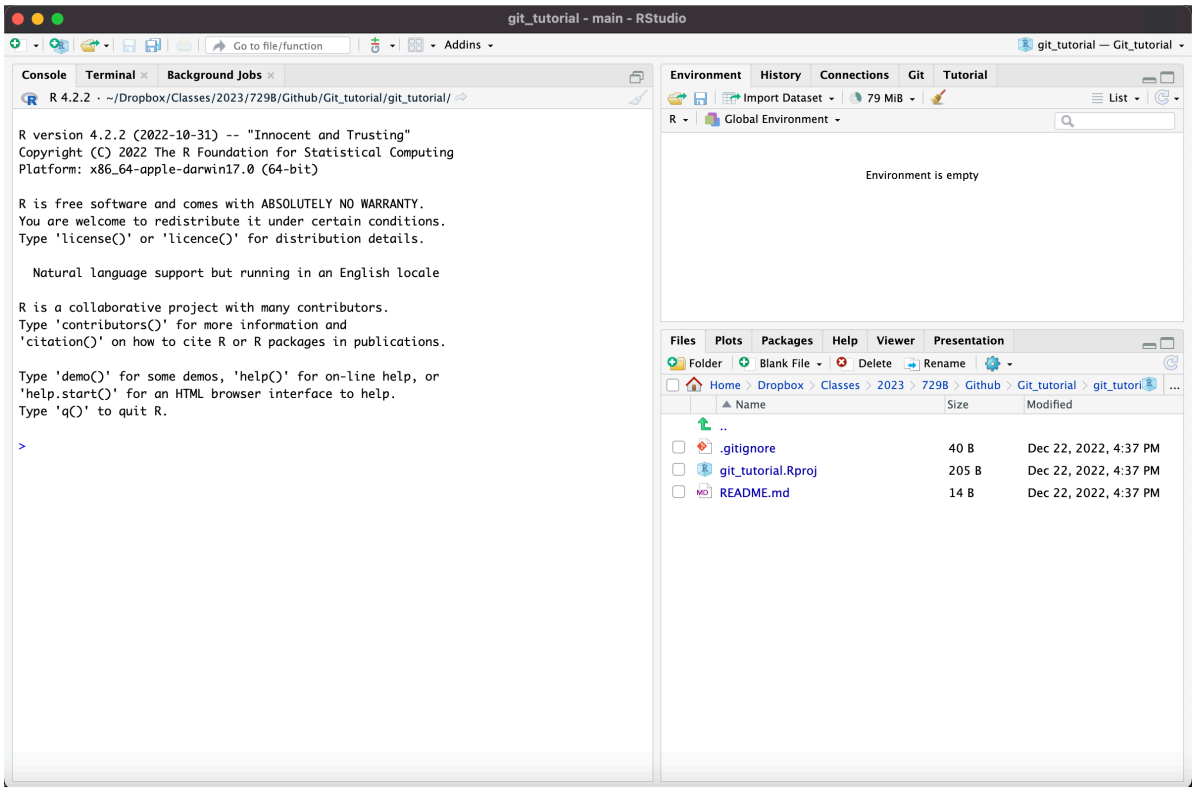


Figure 9: local files replicating external repository files

```
library(usethis)
edit_git_config()
```

```
* Edit '/Users/hannahbirnir/.gitconfig'
```

Remember that you have to install the "usethis" package if it is not already on your computer.

If your git and hub have no problems communicating you can set about modifying your local files at will, adding files and changing them.

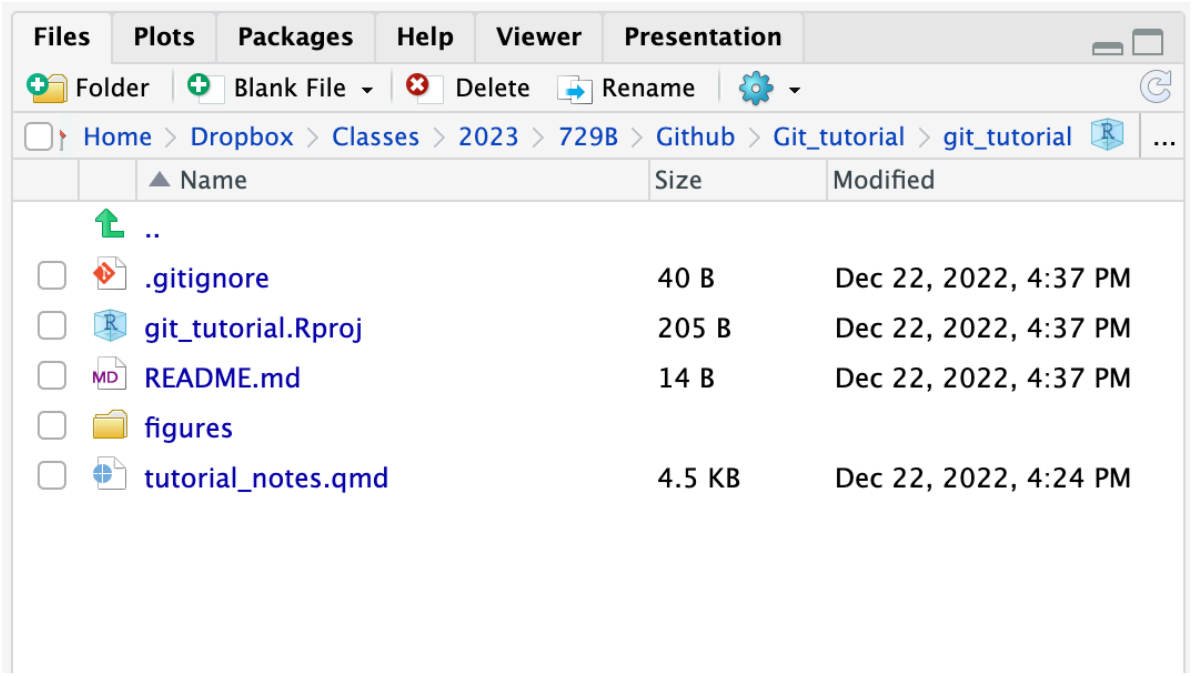


Figure 10: Modifying local files

Each time you add a new file in your local directory or change it in some way it will appear in your git tab like so:

Notice that because I have not modified the README file that was imported from github this file does not appear in my git tab.

Pushing files

The final step is to commit the changes I have made to my local files to the github repository. For this purpose I have to commit the files I want to update (I only commit the files I wish to

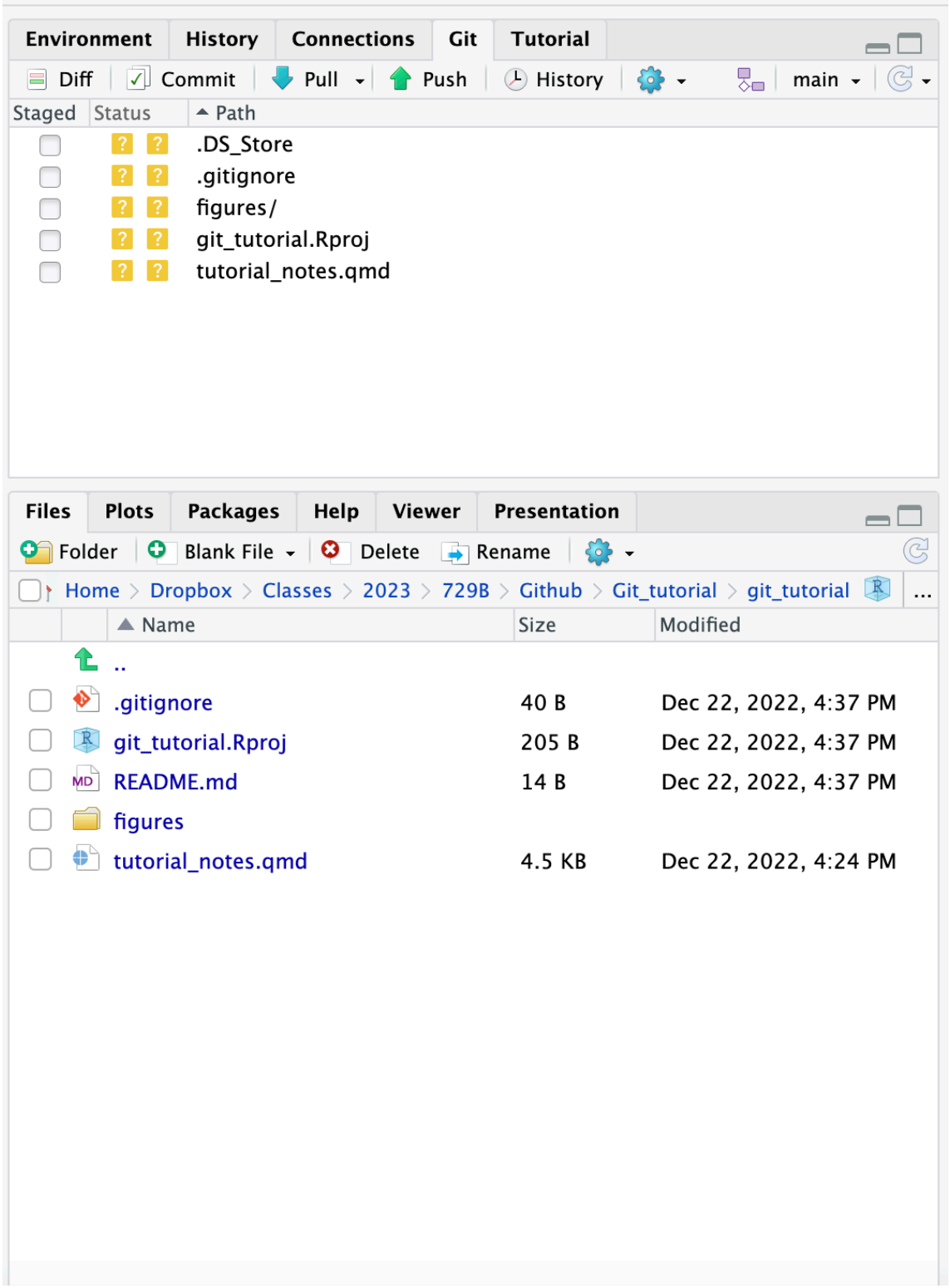


Figure 11: Local files in Git tab

update) and then I need to push them to the external repository.

To do this I first select the files that I want to commit. Next I hit the commit button.

When I hit the commit button another window pops up where I can write myself notes about the changes I am committing.

Once I hit commit R-studio knows which local files I want to change in my external depository. Note that I can work locally and commit many files and then work on other files and commit them later. **Commit only commits changes to my files locally.** The last step then is to push all my locally committed files to my external repository.

From within my project I simply push the push button in my now empty git tab and my external repository is updated.

Github then tracks all of the changes in each file and new files added in each commit while also updating the main file.

This is very nice if you are working on a project - like a website that you might want to change frequently. You can simply open the project - make a change to any one part of the project and commit those changes to github. Imagine, for example, a website where you post new data and other information as it becomes available.

Exercise

Install git on your computer

Create a github account

Create a version control project in R studio and upload it to Github

Working together in git.

One of the most useful things about github is the ability to work collaboratively.

Associating your RStudio with a collaborative project

First, in order to work collaboratively, you may need to associate your RStudio with a project in GitHub that you did not create. If you created the project, do the following to add collaborators:

Go on the Github website to Settings > Manage Access > Invite a collaborator.

Your teammate should accept the invite in their email.

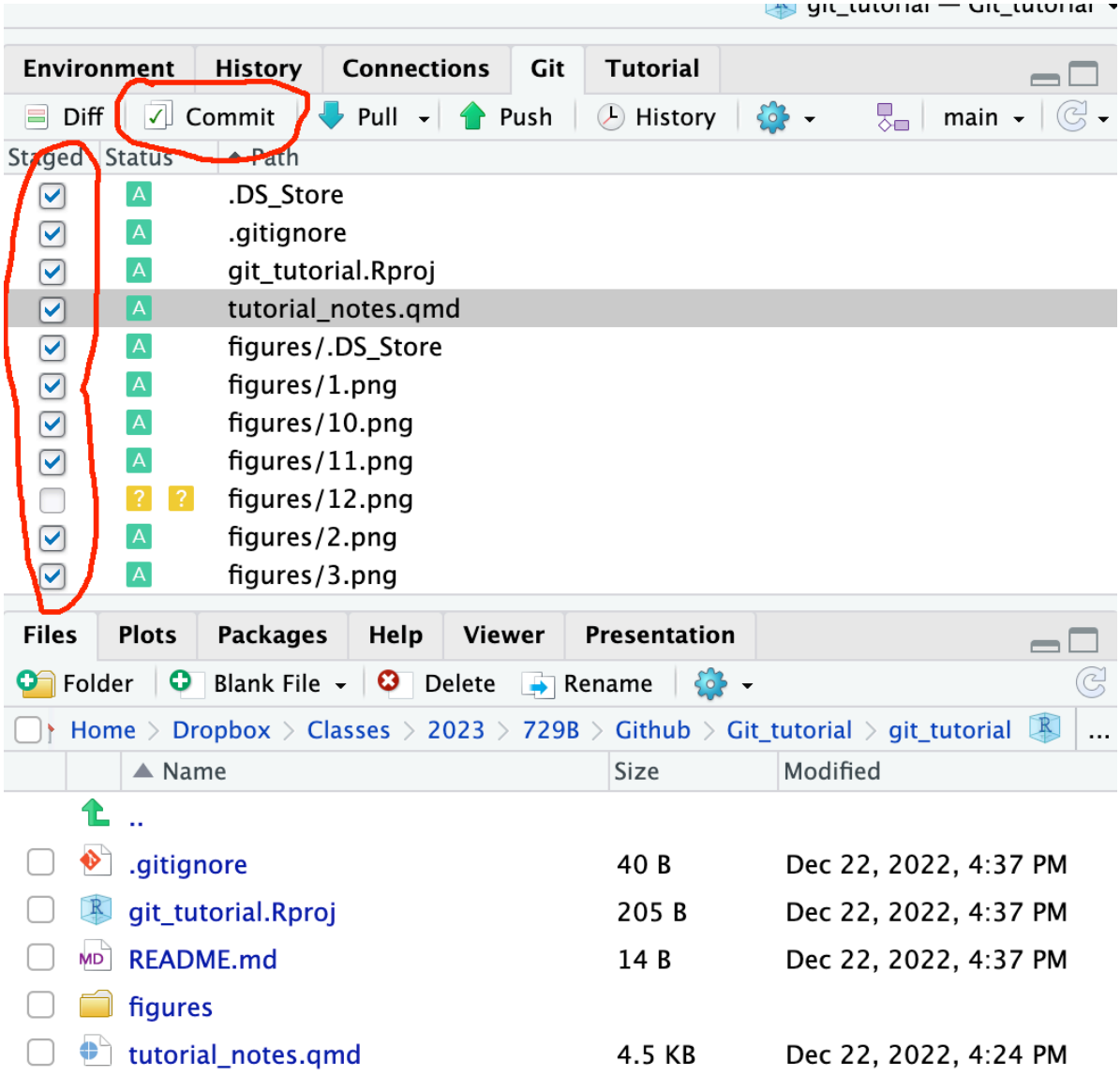


Figure 12: Local files in Git tab

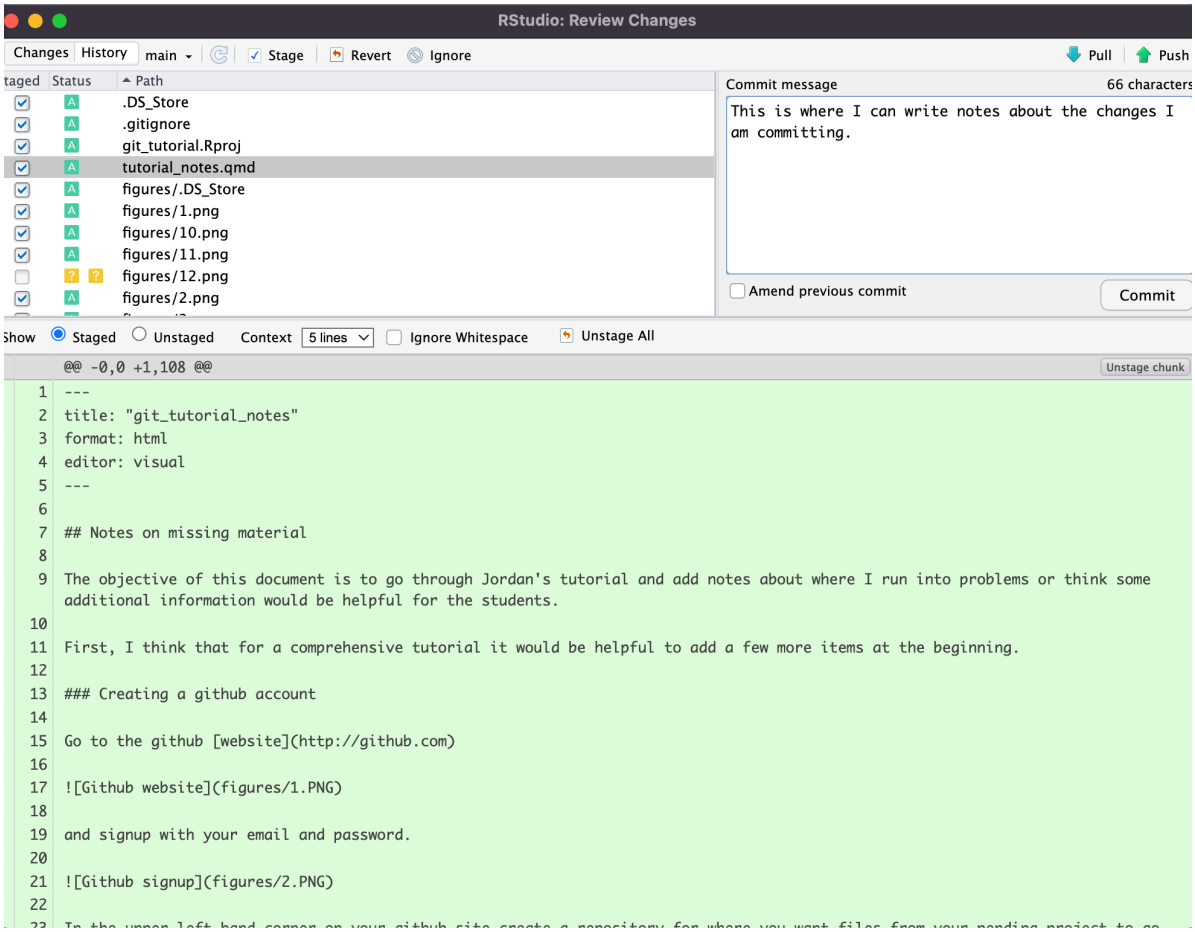


Figure 13: Commit notes

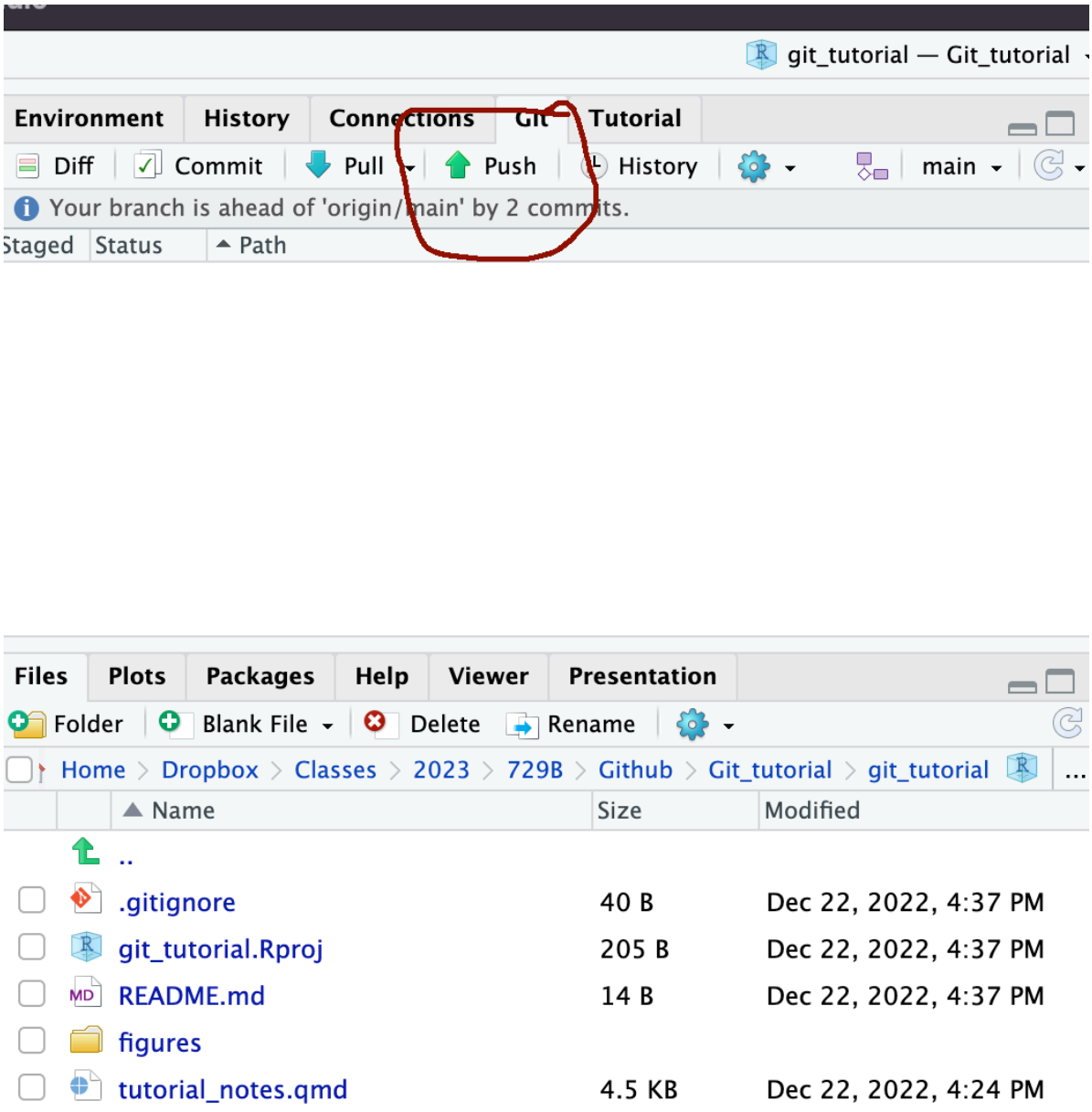


Figure 14: Pushing changes to github

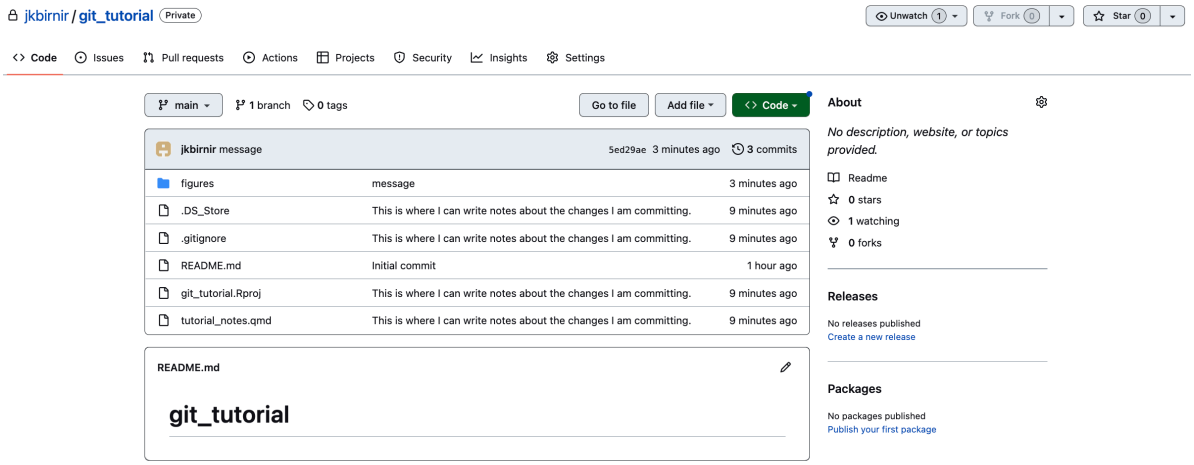


Figure 15: Updated external repository

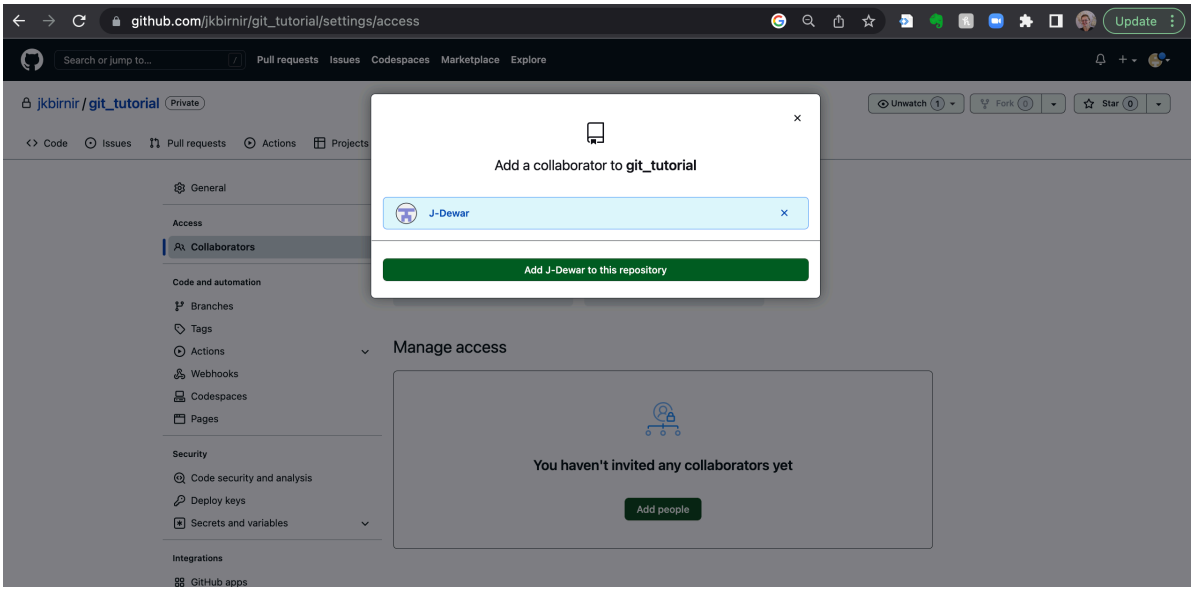


Figure 16: Inviting Github Collaborations

Once this is complete, you can use the steps above to associate your RStuido with the GitHub project.

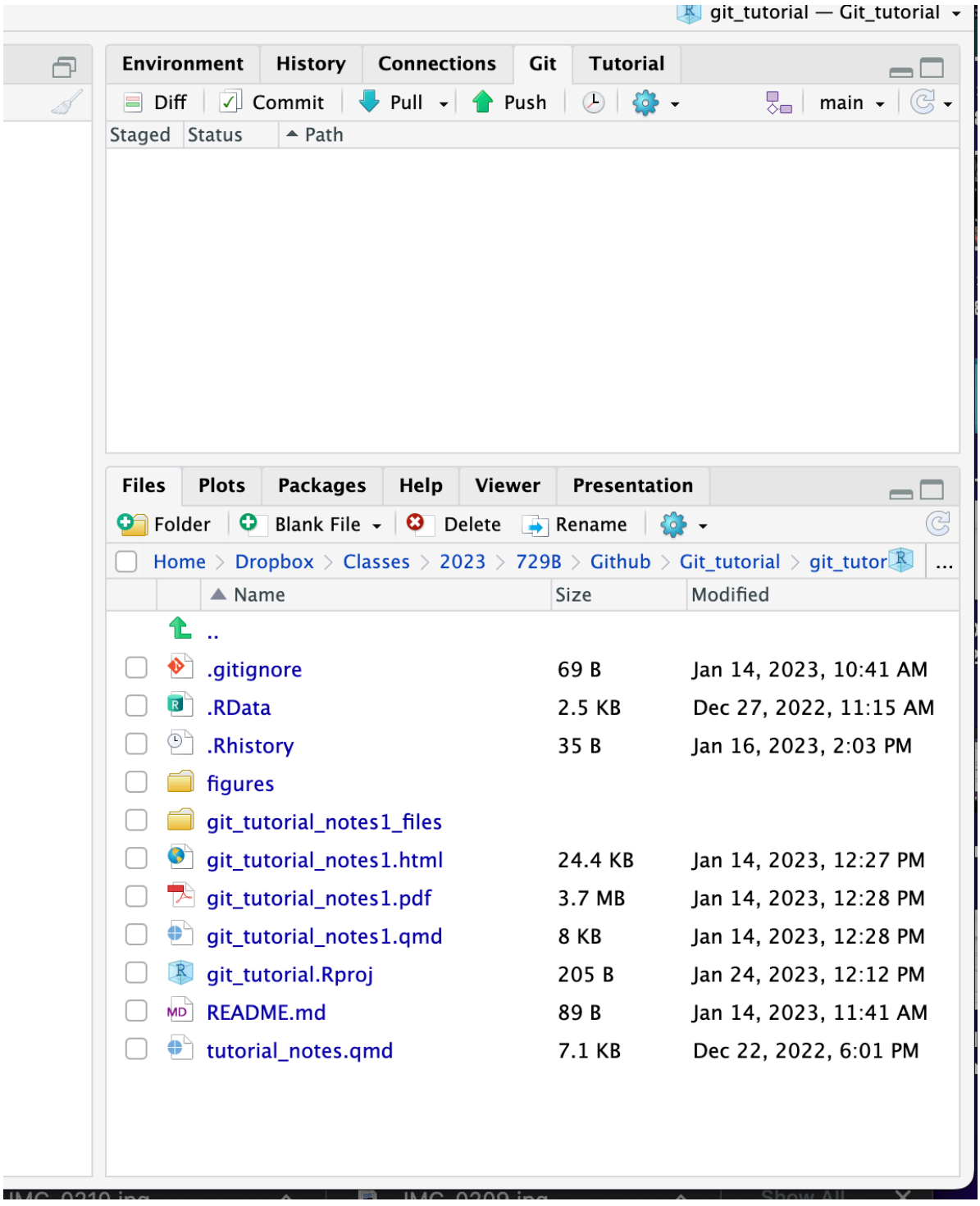
Pulling Changes

One important aspect of collaboration in Github is the ability to pull changes. This allows you to update your code to align with changes pushed by collaborators.

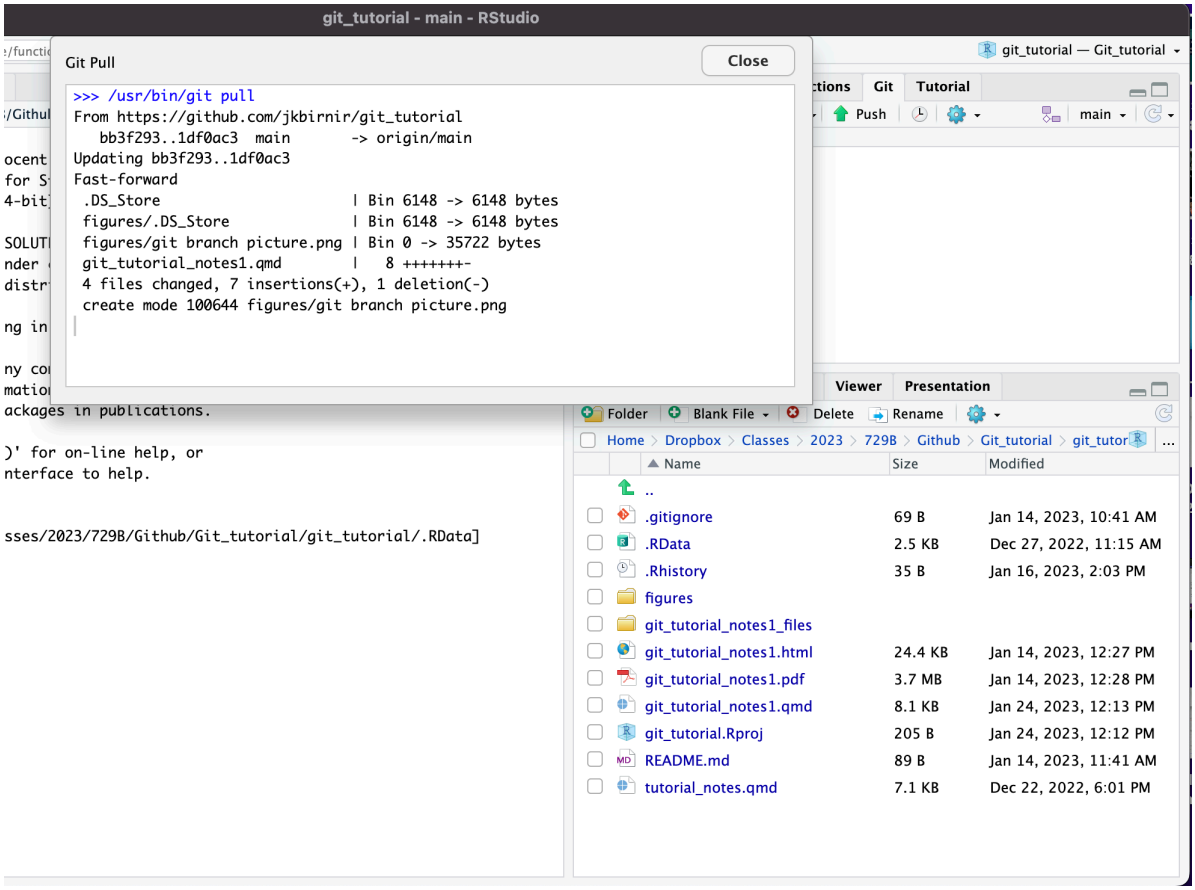
Using the down arrow button, RStudio goes to the GitHub repo, grabs the most recent code and brings it into your local editor. (Pulling regularly is extremely important if you're collaborating, though if you're the only one working on an RStudio project and associated GitHub repo, you know your local code matches what's on GitHub so it's less important.)

To pull, click the blue down arrow on your Git tab to see if you have changes to pull. If collaborating, you might run into merge conflicts.

When you pull your project updates to show the changes your collaborator has made to the project. Look at the dates.

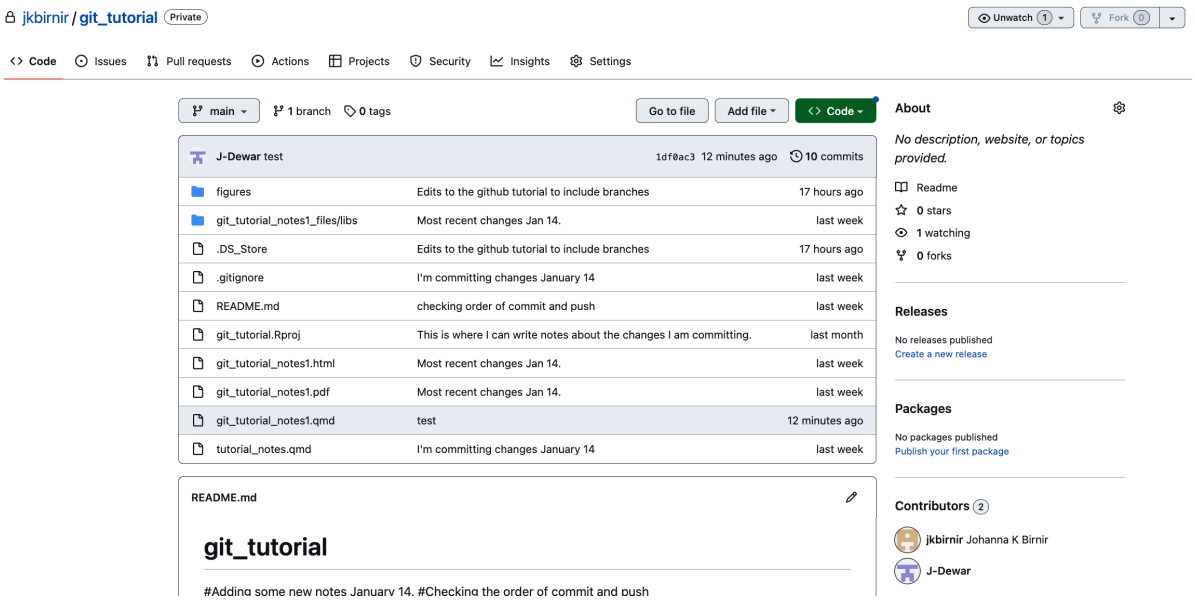

git_tutorial — Git_tutorial
Environment
History
Connections
Git
Tutorial
Diff
Commit
Pull
Push
main
Staged
Status
Path
Files
Plots
Packages
Help
Viewer
Presentation
Folder
Blank File
Delete
Rename
Home
Dropbox
Classes
2023
729B
Github
Git_tutorial
git_tutor
Name
Size
Modified
..
.gitignore 69 B Jan 14, 2023, 10:41 AM
.RData 2.5 KB Dec 27, 2022, 11:15 AM
.Rhistory 35 B Jan 16, 2023, 2:03 PM
figures
git_tutorial_notes1_files
git_tutorial_notes1.html 24.4 KB Jan 14, 2023, 12:27 PM
git_tutorial_notes1.pdf 3.7 MB Jan 14, 2023, 12:28 PM
git_tutorial_notes1.qmd 8 KB Jan 14, 2023, 12:28 PM
git_tutorial.Rproj 205 B Jan 24, 2023, 12:12 PM
README.md 89 B Jan 14, 2023, 11:41 AM
tutorial_notes.qmd 7.1 KB Dec 22, 2022, 6:01 PM

and

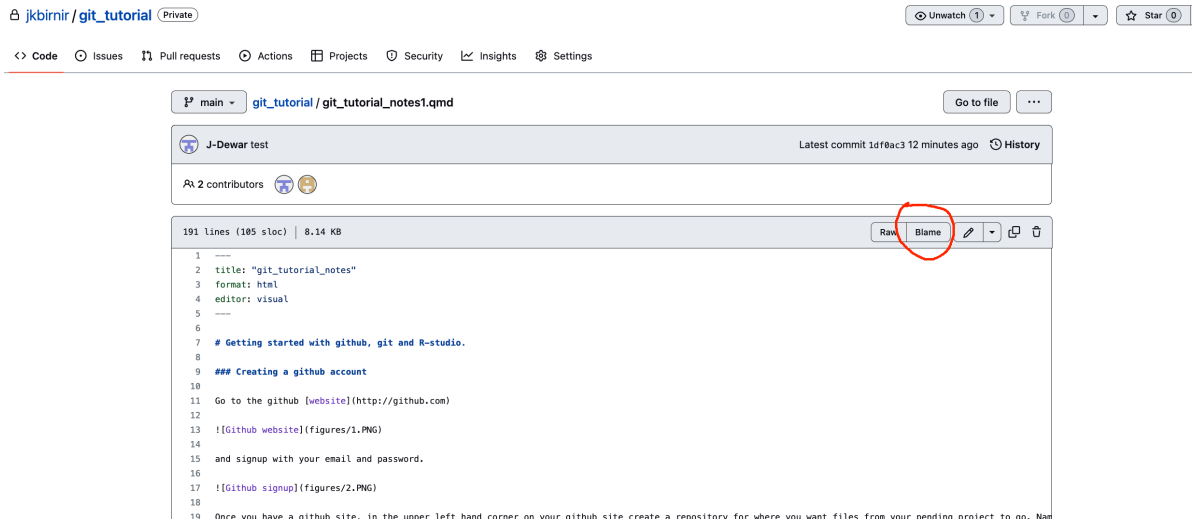


You can also track the changes on github if you want more details.

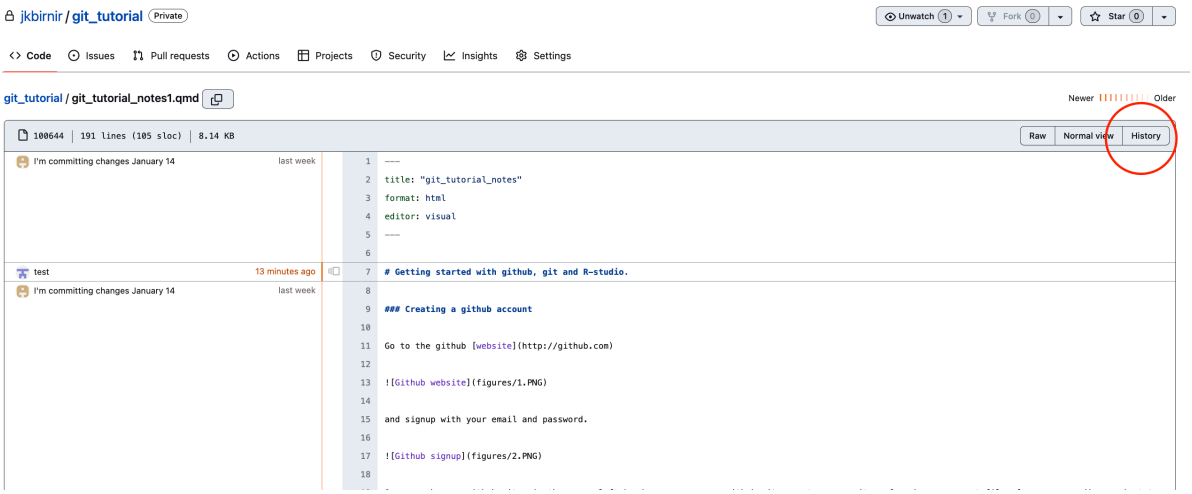
Go to your github project:



There you see who are the collaborators and when each item was updated. For even more information click on any of the files (here the qmd file)

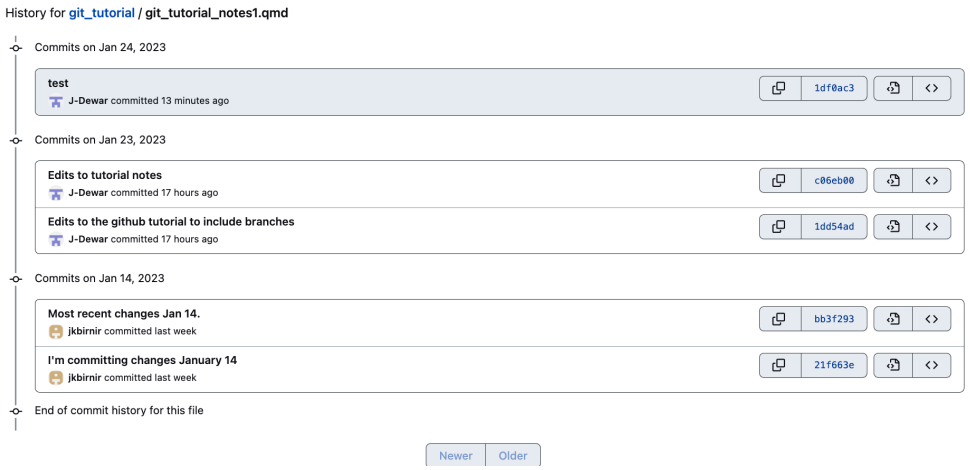


Here you see the history of the development of the project and if you want to see who made what changes when push the blame button.



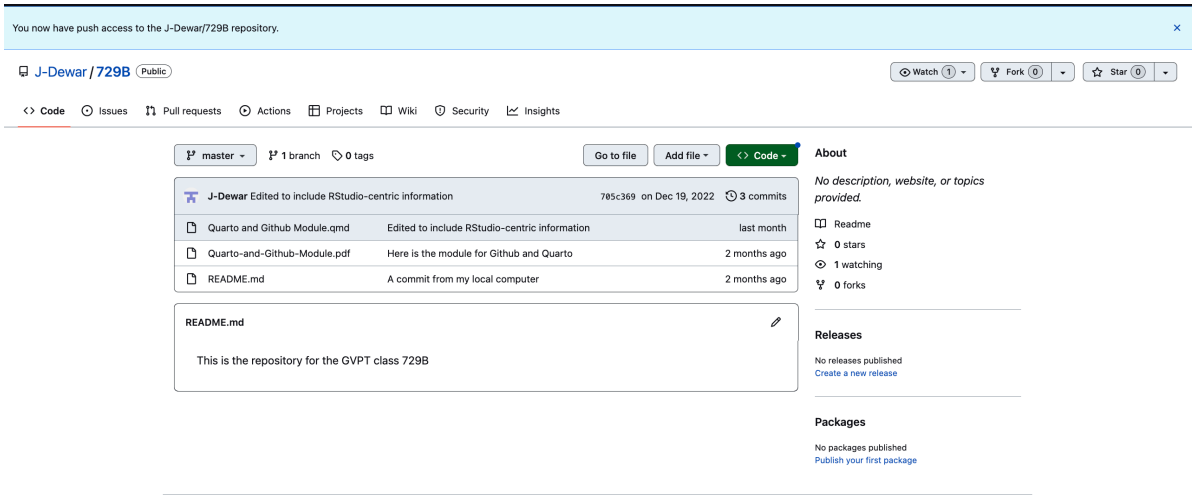
The blame allows you to blame whoever - mostly yourself ;) is responsible for making changes to your project.

If you want more of an overview - then push the history button and you get a summary of changes:

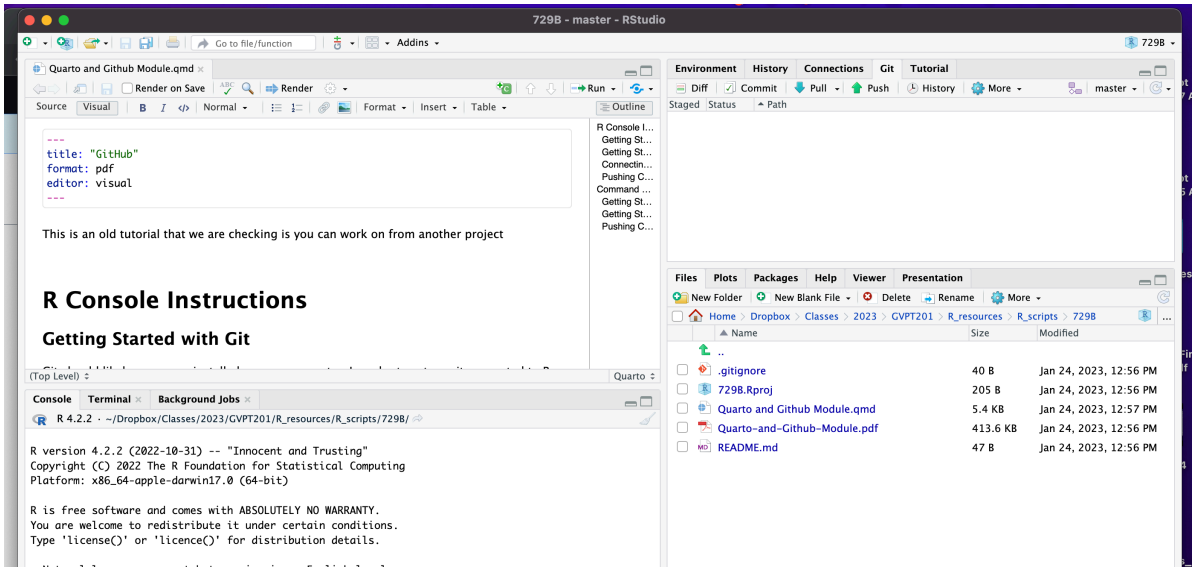


In sum this is the workflow when you and your collaborator are both working on the main project and either one of you can make changes to the project.

When someone invites you to a project and to work on the main as here:



You can open the project locally as you would any other version control project (see earlier steps in the tutorial)



Once you are done changing your files locally - you then go through the same steps of committing and pushing. Which results in this message telling you the process was successful.