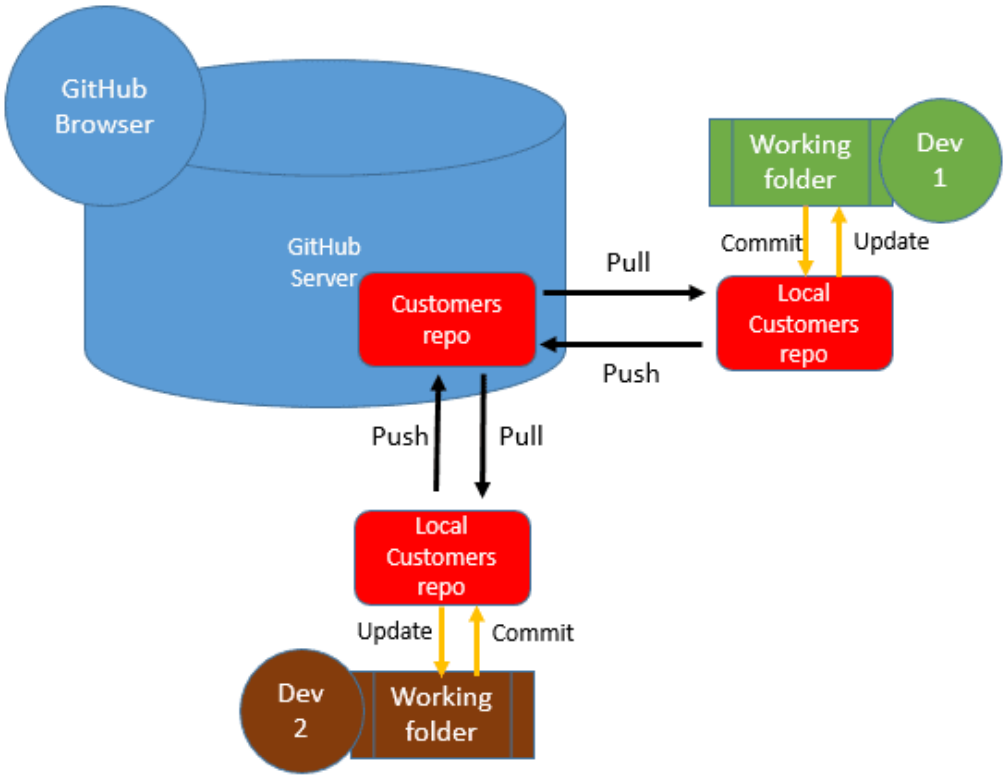
Cleaning up your code and providing it in github - **for papers**

Quick overview of git and github

Git is a version control system.
→ It saves a history of code versions on your local machine.
→ you can add a new version of your code (for example a new script or a change to an older script) via the **add** and **commit** commands.

GitHub is a free webservice, were you can provide your git code history.
→ To use GitHub you need a Github account. Moreover, we have a group account, were we share the code for all published papers and tools.
→ You can **push** code the code on your local machine to the internet. Or **pull** code from the internet version of the project to your local machine.



Important:
1. **push** will only push code that was **commit**ed before.
2. Sometimes you cannot push, because the version of the code was changed in between in Github (E.g. when you change something from the github website). Then you need to **pull** before you **push**.

1. Prepare a clean final version of your code

- Important!!! All figures that you contributed to the paper should be made by the code you provide. Also make sure, that the code really is the final final final version of the code.

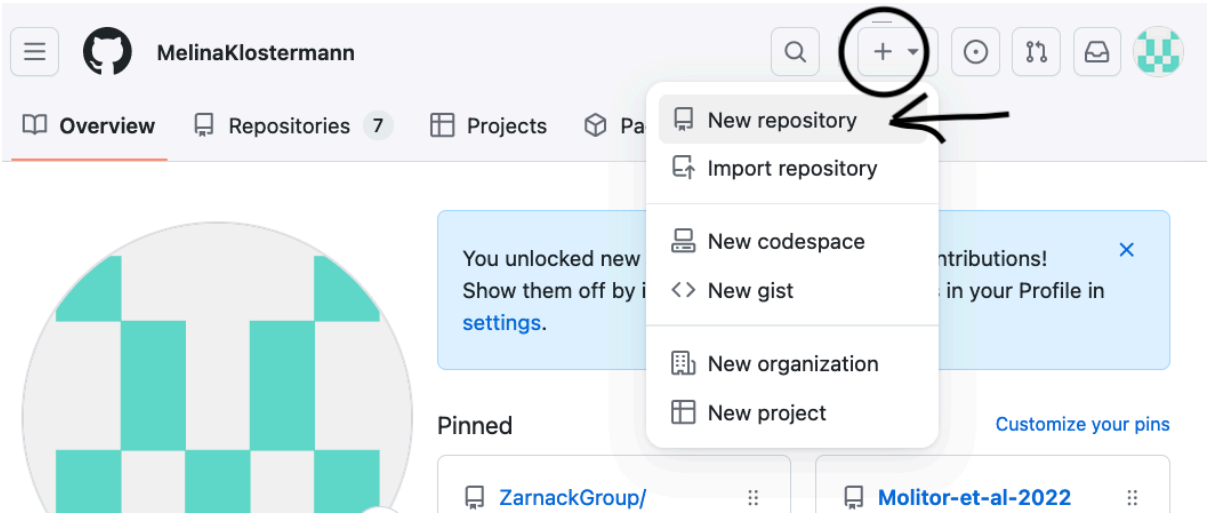
- Make a new empty folder.

- This folder will get all the finial code. Don't put any other code. E.g. code pieces that do not work, figures that were not shown in the publication, analyses that you decided not to show in the publication.

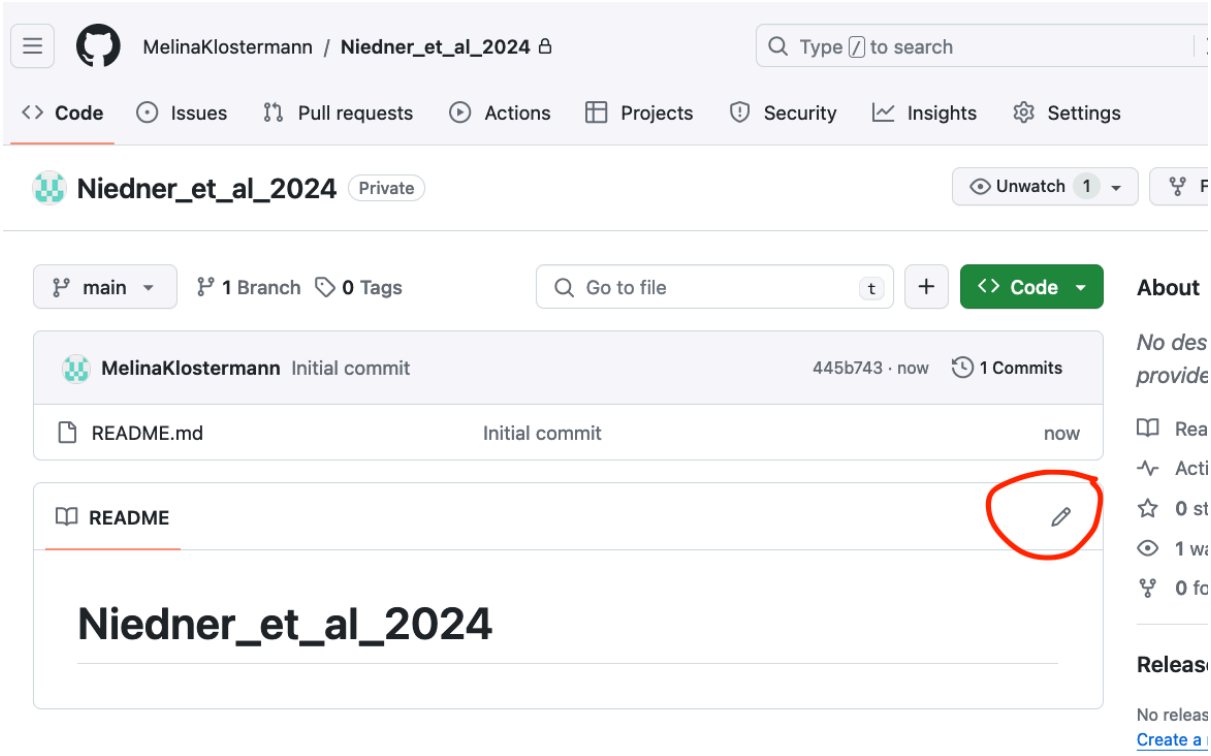
- Before you put anything in it. Make an Rproject inside this folder

1.0 R project with github link

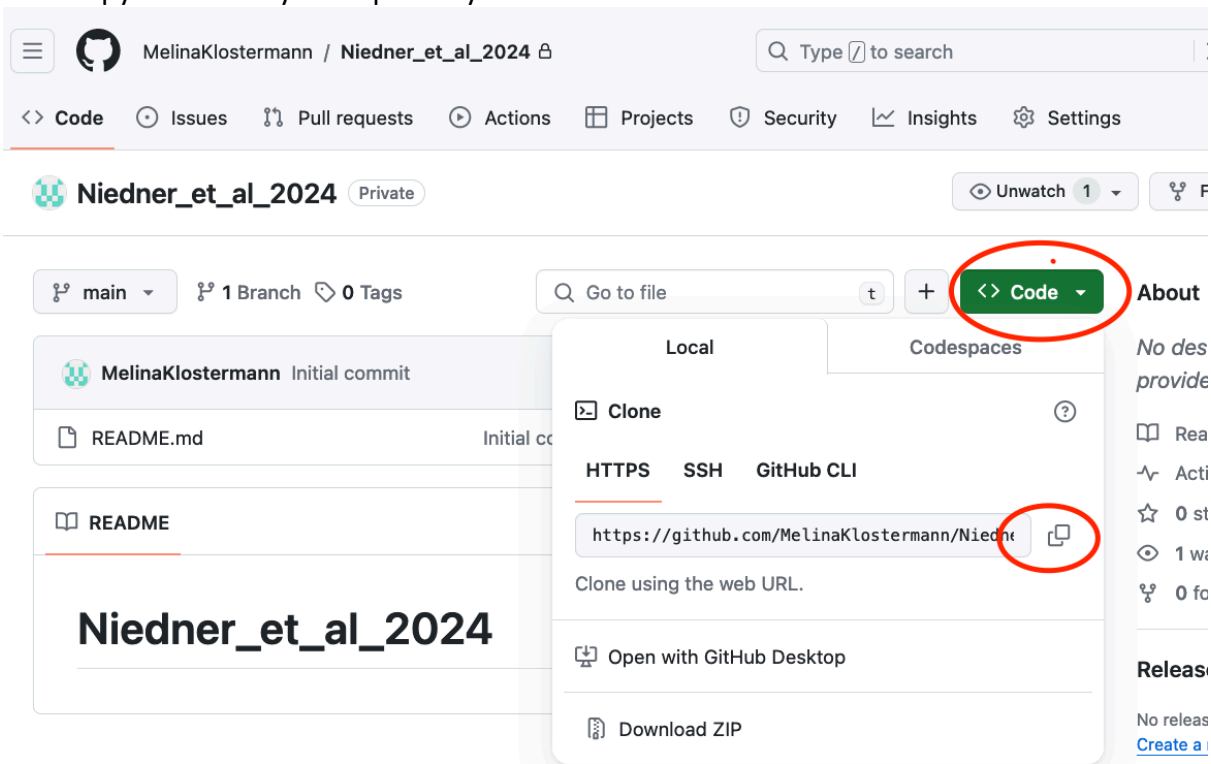
- If you haven't already, create a github account for yourself at https://github.com/
- Then create a new repository.
- The name of the repository should be "**NameOfTheFirstAuthor_et_al_year"** to make the connection to the manuscript easy.
- When you first create the repository choose **private.** Only make it **public** as soon as it is send out to a journal. (Also when it is send out, make a fork to the Zarnack group github (or tell Melina to do if you do not have the necessary github rights to do it). Look below for how to do a fork)
- Also choose "Add a README file"



- You now have your github repository. Add the abstract of the manuscript in the README file



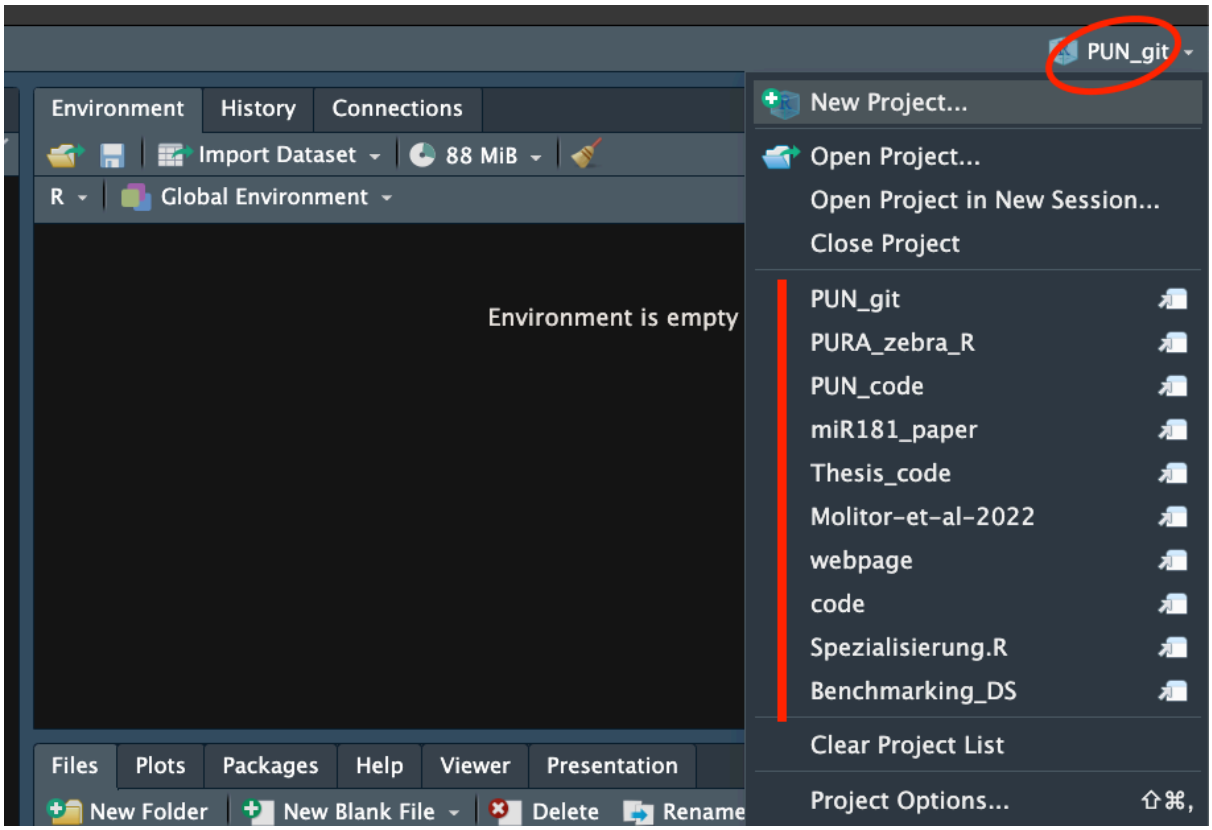
- Now copy the link to your repository



-

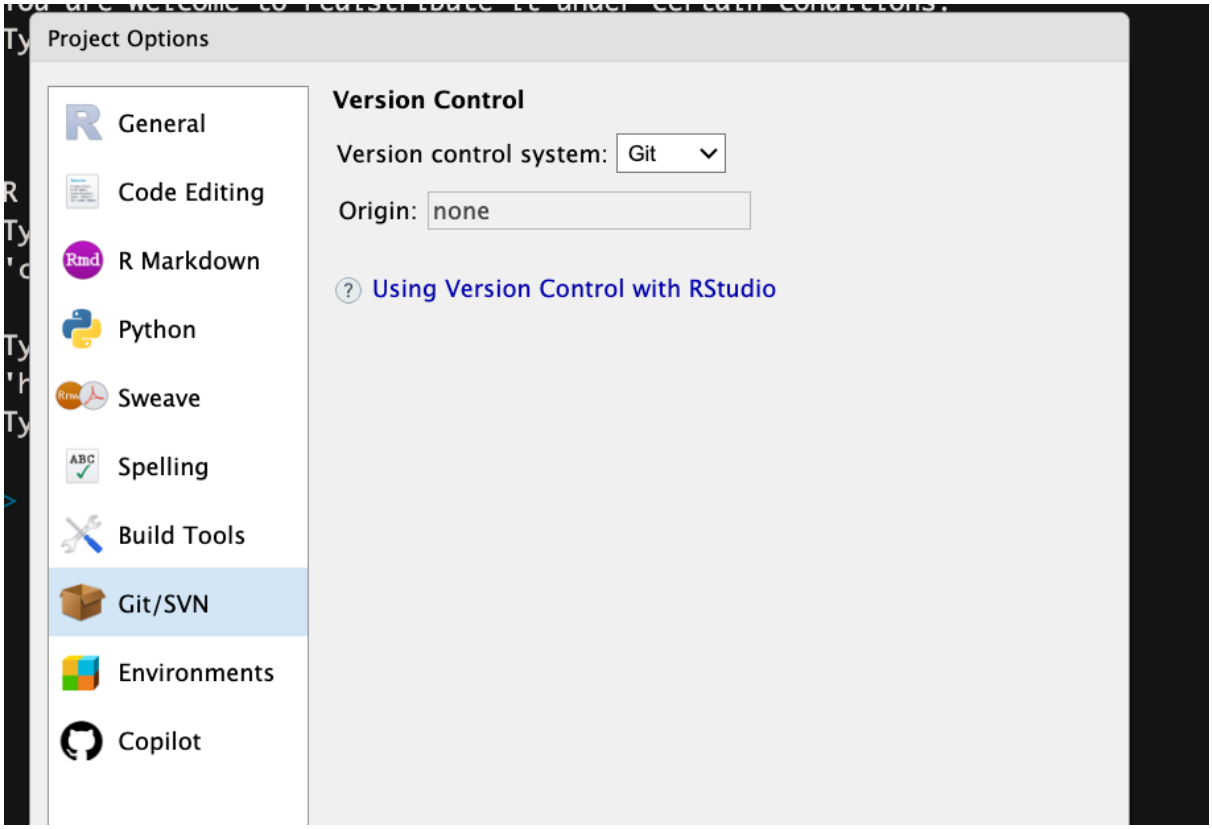
- Open RStudio and choose File > New project >version control > git
- The provide the link to your github repository and the path to the empty folder
- Tada you now have a R project.

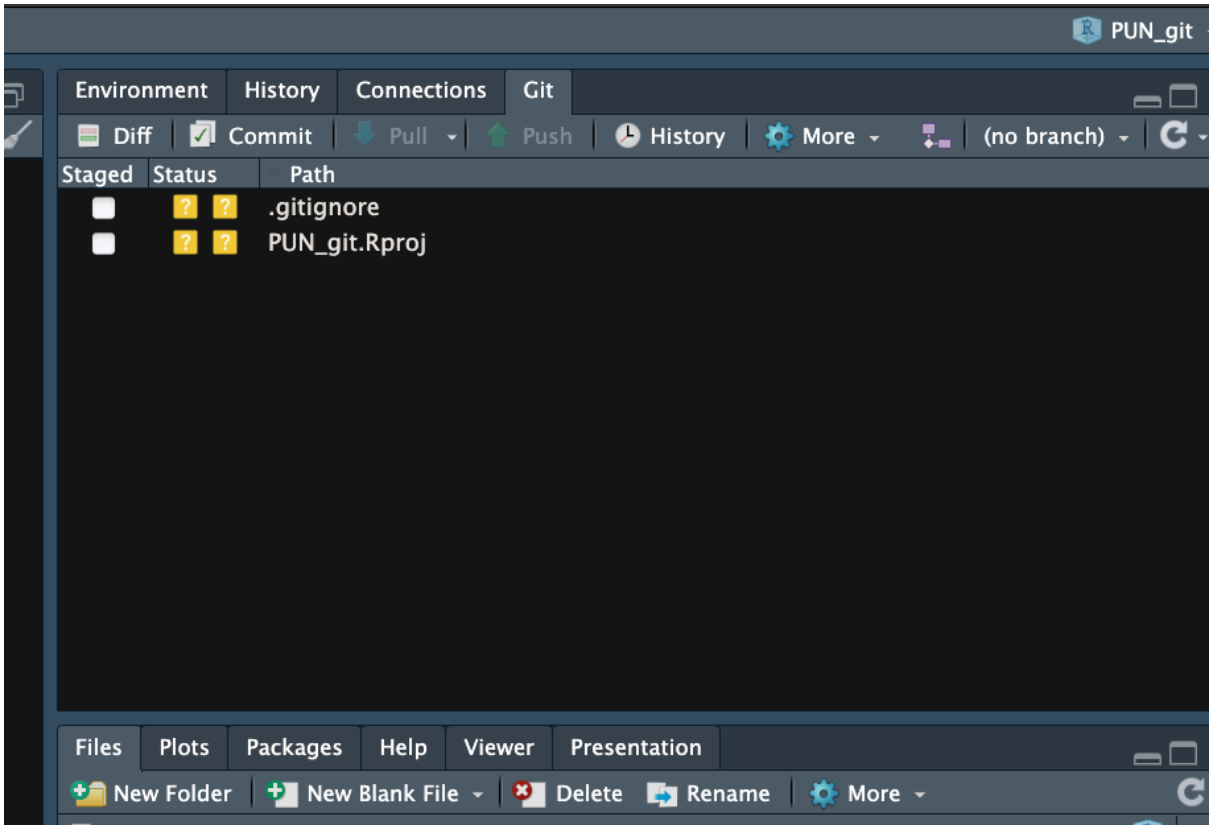
- You can see in which project you are on the top right of the Rstudio window. And you can also change projects there.



-

- Now choose tools > version control > project setup > git/svn and set git as version control

- 

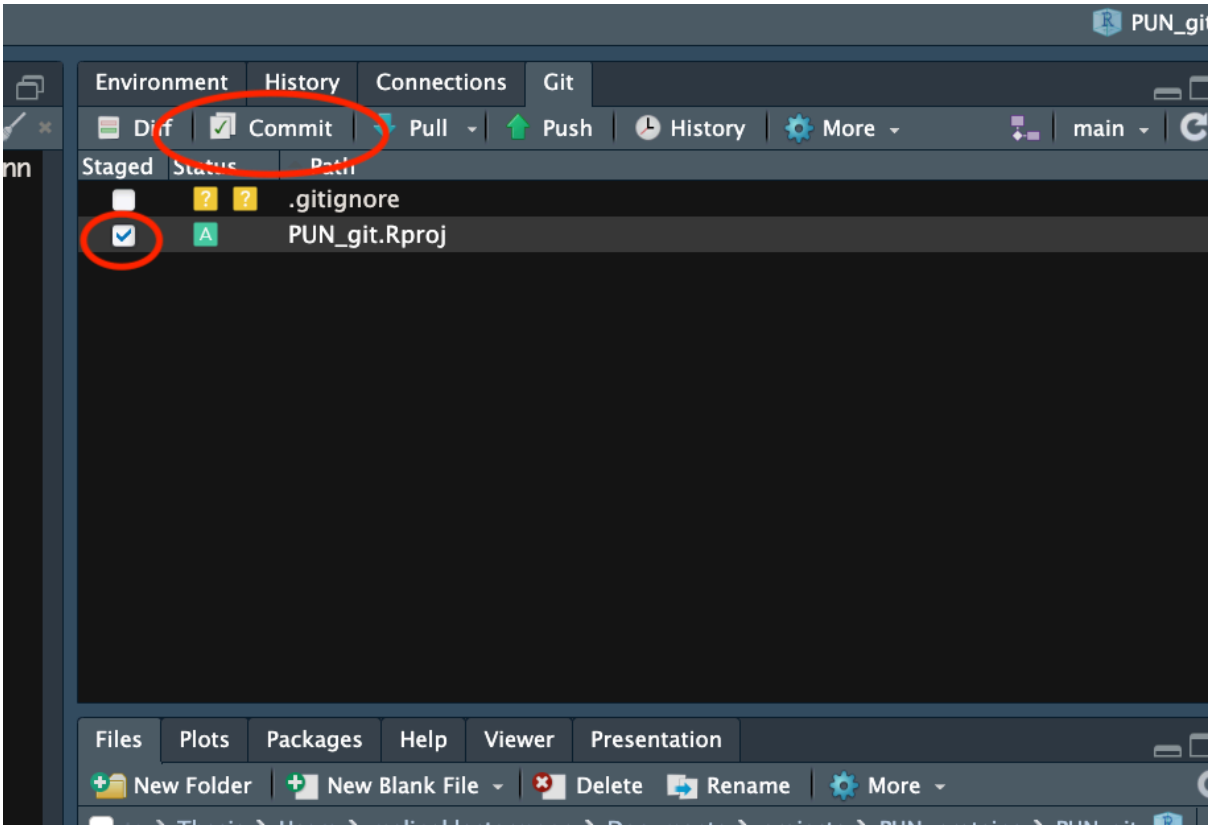
- Restart RStudio
- You should now see the Git Tab in the upper right panel
- 

- To connect to github go un more > new terminal

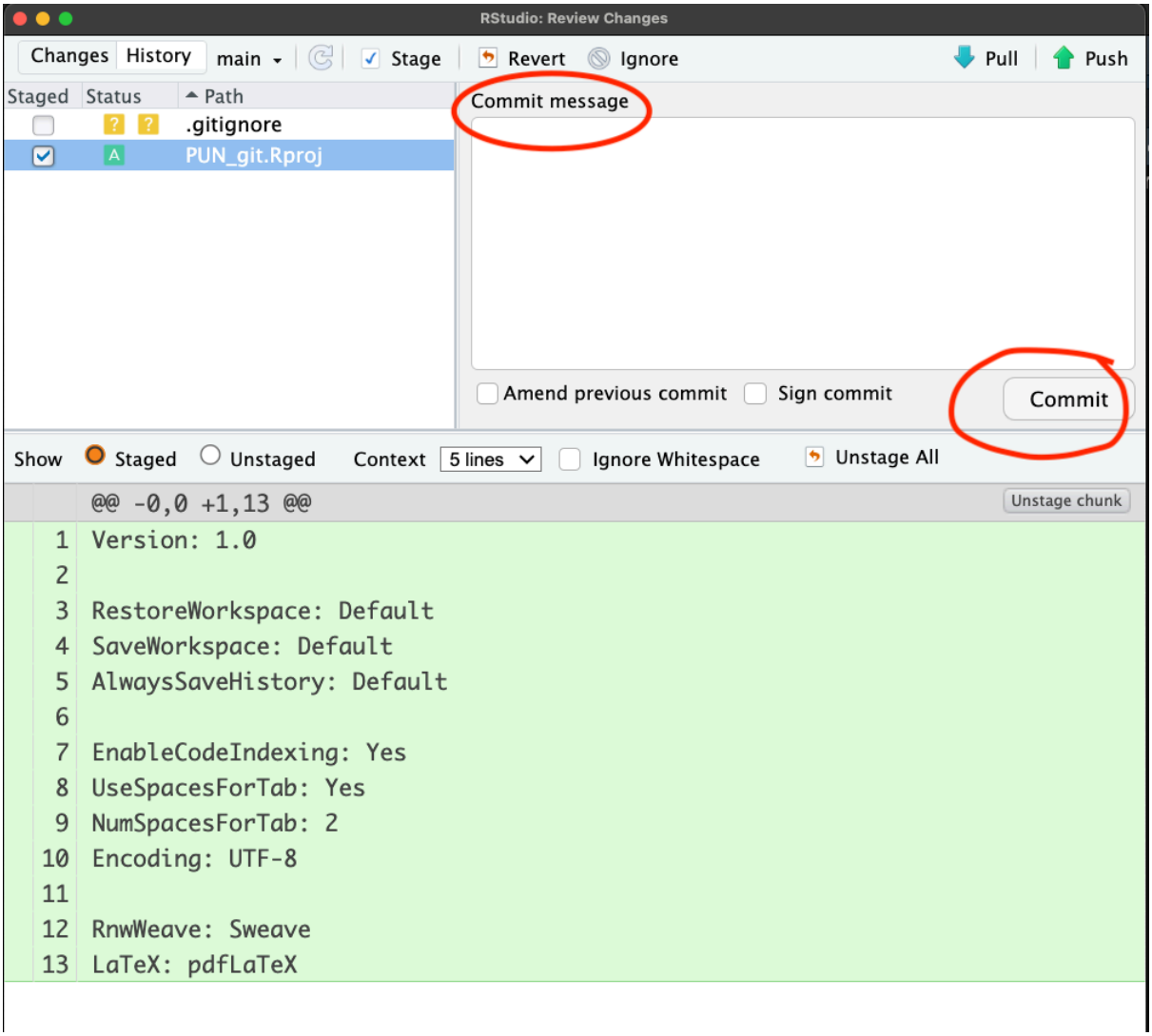
- In the terminal type

  git remote add origin <github link>
  git pull origin main
  git push -u origin main

- Make scripts with you code in the new project. See below for some rules on clean code ☺
- When you have a script finished. Click on the script in the Git window and then click commit



- The **commit** window will open, you have to type some kind of commit message, then click commit. You now saved a new version of the script in your history.



- After you commited you can now push the script to github by clicking **push**
- Check in the webbrowser if the file is there ☺
- Do this for all code of the publication.

Here are some comments on how to make sure your code is clean:

- When you have multiple scripts give the scripts meaning full names and number them in the order they need to be executed. Put the Number in the beginning of the script name.

- Also add a one sentence description of each script in the README file

1.1 R code

- Take a new empty Rmarkdown / RQuarto file(s).
-

In the header include:

- the full title of your thesis/project
- yourself as the author
- the date
- a short abstract of what is done in this script (2-3 sentences)

```yaml
---
title: "My very cool thesis"
output: html_document
author:
- name: "me myself and I"
- name: "Supervision: MySupervisor"
date: "2022-04-20 - 2022-11-11"
abstract: "Blablabla"
---
```

Libraries

- The first code chunk after the setup chunk should be a chunk called libraries. Here you specify all libraries used in your project.
- The very last code chunk of the Rmarkdown should perform sessionInfo() to state the versions of each library that was used.

````
# Session Information

```{r, session_info, include=TRUE, echo=TRUE, results='markup'}
sessionInfo()
```
````

Input files

- The next chunk should be loading all needed input files.
- If you did preprocessing in bash or galaxy, load preprocessed files here and look in the sections below, about bash and galaxy

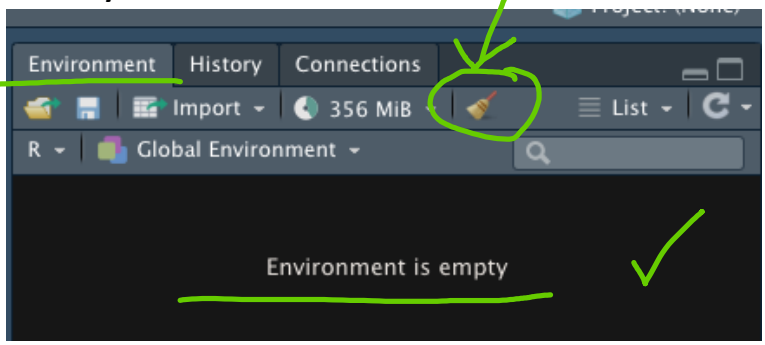
Your code

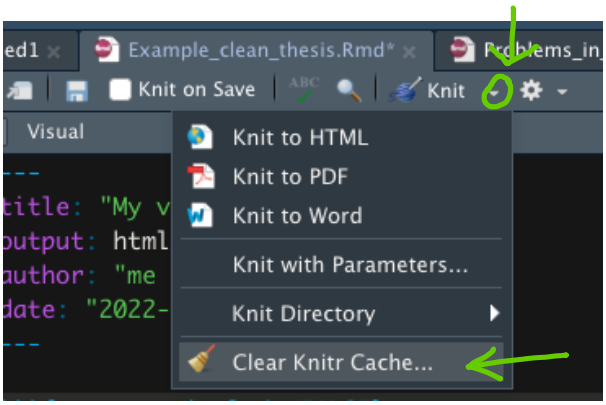
- Now add all the code necessary to produce all your results (as shown in the results part of your thesis) including all plots and tables starting from the input files that you received or downloaded initially.

- Look at these coding style rules and follow them: https://style.tidyverse.org/files.html
- Comment you code in a way that other people can understand what you did.
- Structure your code in reasonable code chunks. Each chunk does one specific analysis.
- Remove all unnecessary code.
- Remove all out commented code.

Your markdown

- You should knit your rmarkdown/quarto to html or pdf in the end.
- **Before you knit clear the knit cache and the environment!!!**



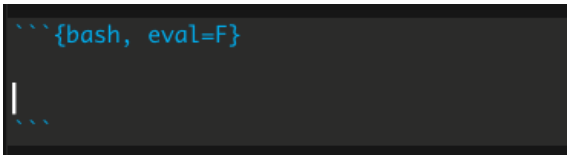


- Use headers (# First big topic) and subheaders (## small part of first topic) to structure your markdown.
- Write quick explanations of why you are doing what you are doing in front of the chunks where you do it (e.g. PCA plots are made to look at variability between samples)

1.2 bash scripts

If you used bash scripts in your thesis prior to further analysis in R include all your .sh files in the rmarkdwon as not executed bash chunk.

- Although the bash chunks are not executed keep them in the order of execution.
- Explain the input and output files either in comments in the bash script or between the chunks in the R markdown.



- Mention the versions of all tools you use in comments.

1.3 galaxy steps

If you preprocessed your input files with galaxy. Copy the description of all steps performed in galaxy form the methods part of your thesis and put it in the beginning of your markdown script.

1.4 other languages

For python ask Dominik ☺
For other languages come up with something nice!

3. Code revision

(We cannot do code revision for all papers, only look at this if Kathi wants someone to revise your code, for the manuscript.)

In addition to the github code, you should provide several input and output files on the ebersberger cluster.

3.1 What you should provide:

Make a folder that contains

1. A pdf or html version of your knitted final Rmarkdowns/Quartos

2. All raw data that you received from collaborators (also if you do not show anything of them in the thesis/your project in the end). Write a file called raw_files_used.log and add it to the folder. In this file for each sample specify
    a. how you call it in your Rmarkdown
    b. what the original name of the file was (e.g. SRA236807585)
    c. What celline and condition the sample is from
    d. What experiment type the sample is from (e.g. RNAseq)

3. If you downloaded the data from a public source don't put it in the folder, write a file called raw_files_used.log and add it to the folder. In this file for each sample specify
    a. how you call it in your Rmarkdown
    b. what the original name of the file was (e.g. SRA236807585)
    c. What celline and condition the sample is from
    d. the link where you downloaded the file

4. If you did preprocessing in bash or galaxy, include the description of the steps in the Rmarkdown file as described above and provide the output files of the preprocessing, so the files that where read into R . For example:
    a. the htseq count table
    b. majiq dpsi table + splicegraph + modulaise output
    c. pureclip bed files

5. Provide output files from your R analysis as .rds files. Examples are:
    a. The (unfiltered) results table of a DeSeq analysis.
    b. The defined binding sites granges object.
    c. A table with binariezed LSVs.
    d. Generally the output of script1 which is later the input of script2

    → Ask your reviser which output files are needed.

3.2 Where to put it?

- Give the folder the same name as your github repository
- zip the complete folder.
- If you do not have access to the ebersberger cluster and send the zip file to your supervisor.
- Else log into the ebersberger cluster and upload the folder in the xxx directory.