

A tutorial on creating a GitHub repository and linking it with R project

Yinghui Wei, 25 June 2023

Purposes

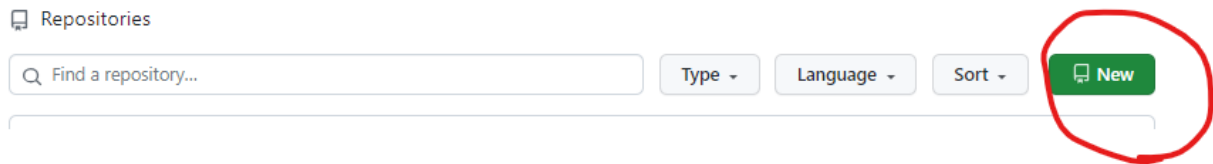
This tutorial guides you through

1. Create a repository on GitHub
2. Create a R project linked to your GitHub repository
3. Guidance on folder organisation and data security

Tutorial

1. Create a GitHub repository

Step 1. Click on “Yinghui-Wei-team” and create on “new” to create a new repository.



Make the repository as **“private”** to start with (see next page).

Create a new repository

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


Required fields are marked with an asterisk ().*

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

 Yinghui-Wei-team ▾

Repository name *


/ presentation_results


✔ presentation_results is available.

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

Description (optional)

This repository gives example code to produce publication-quality tables and figures

☐  **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:

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

Add .gitignore


.gitignore template: R ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

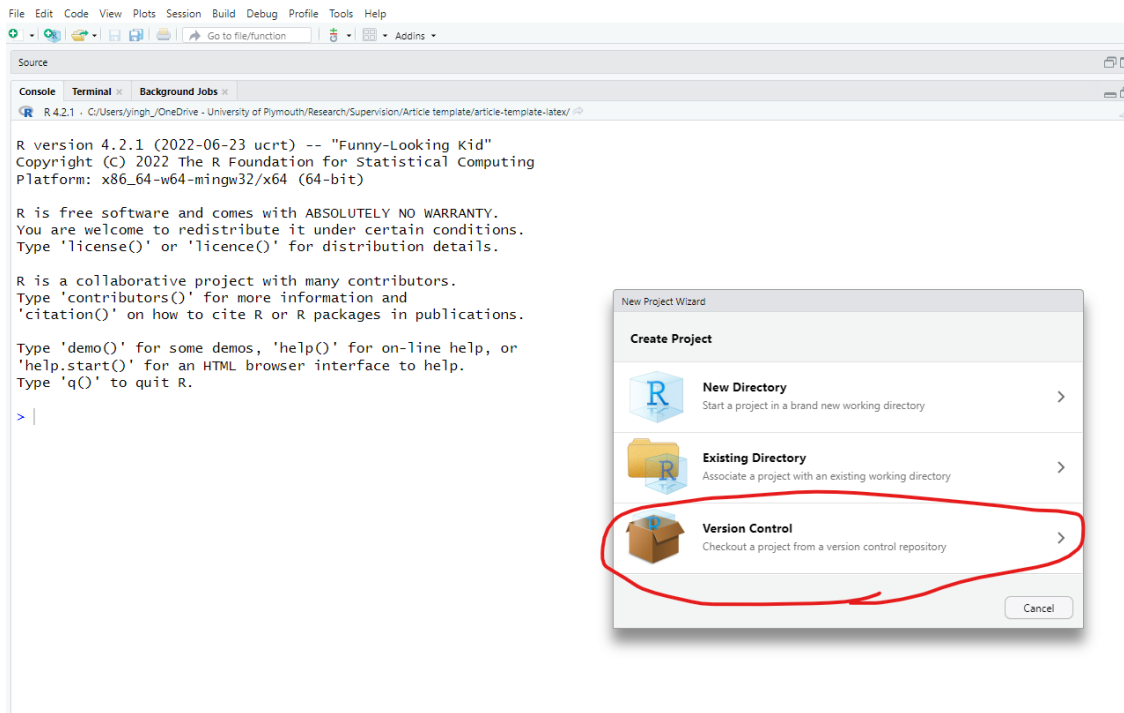
License: MIT License ▾

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

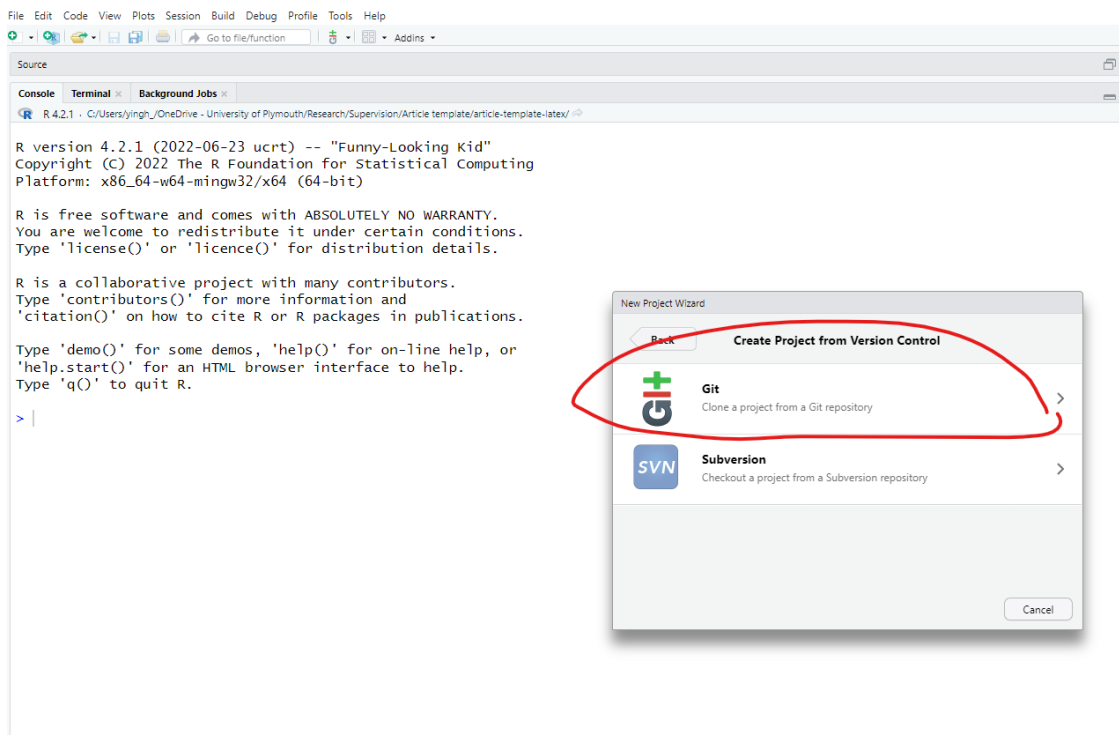
 You are creating a private repository in the Yinghui-Wei-team organization.

Create repository

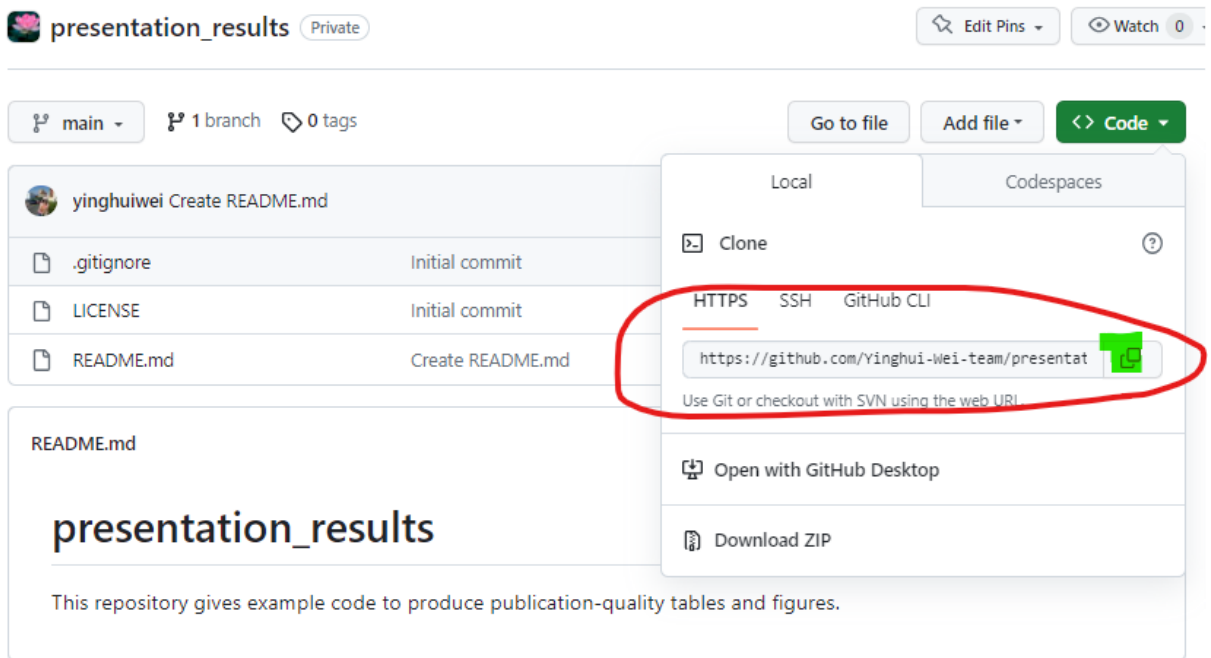
Step 2. Open R Studio. Click on “File” -> “New Project” -> select “Version Control”



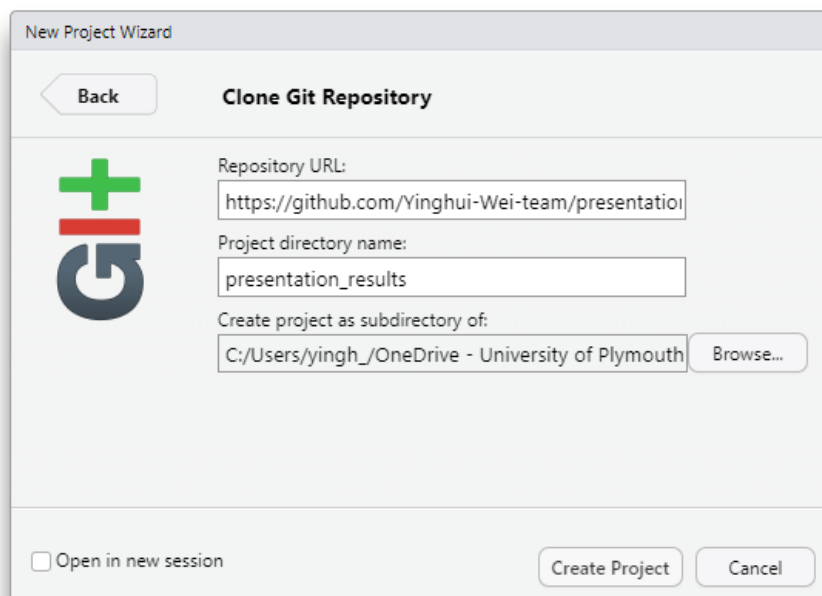
Step 3. Select “Git”



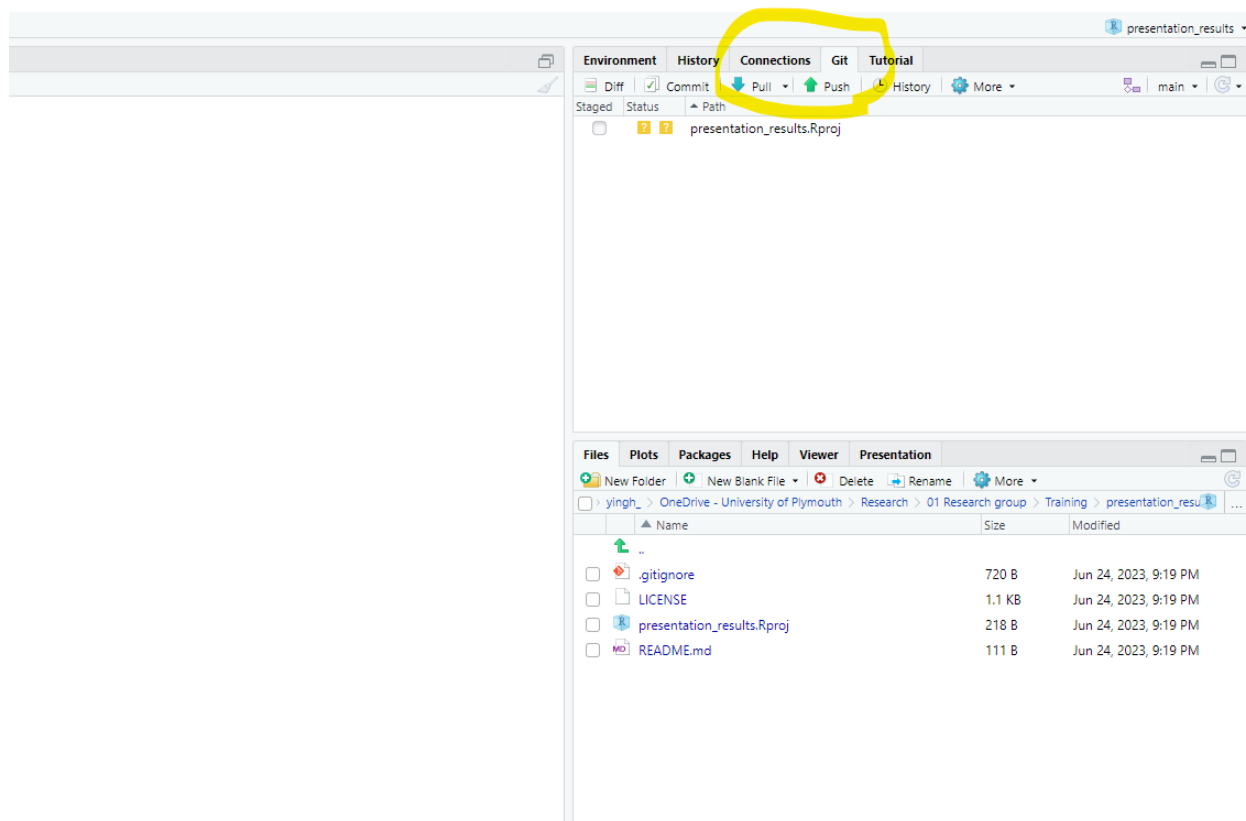
Step 4. Copy GitHub repository URL by clicking “Code”, click the copy button (highlighted in green below)”



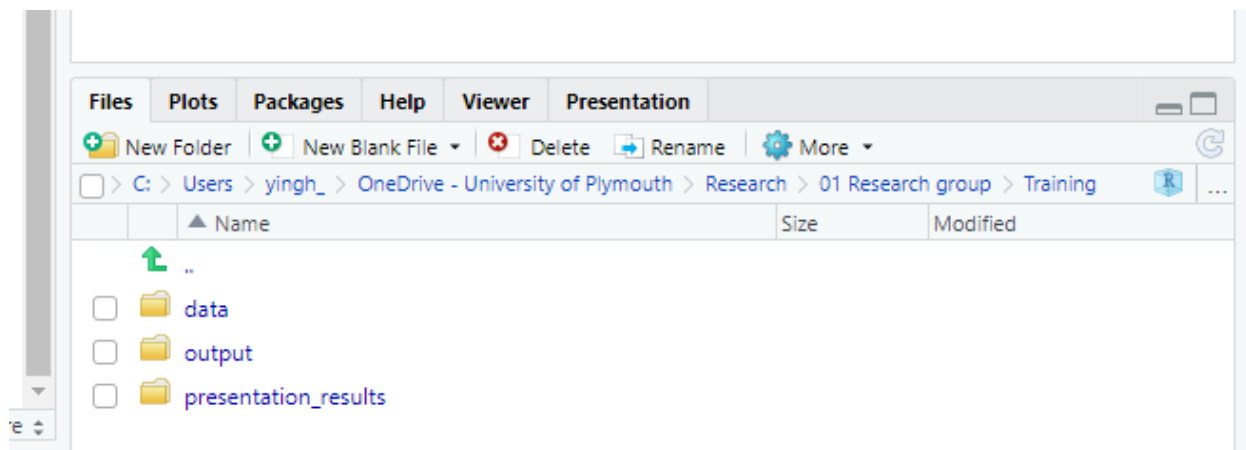
Step 5. Paste the github repository URL in your “New Project Wizard”. Your project directory name – if has more than one word, use underscore “_” to connect words. Make sure you create your project in a relevant subdirectory.



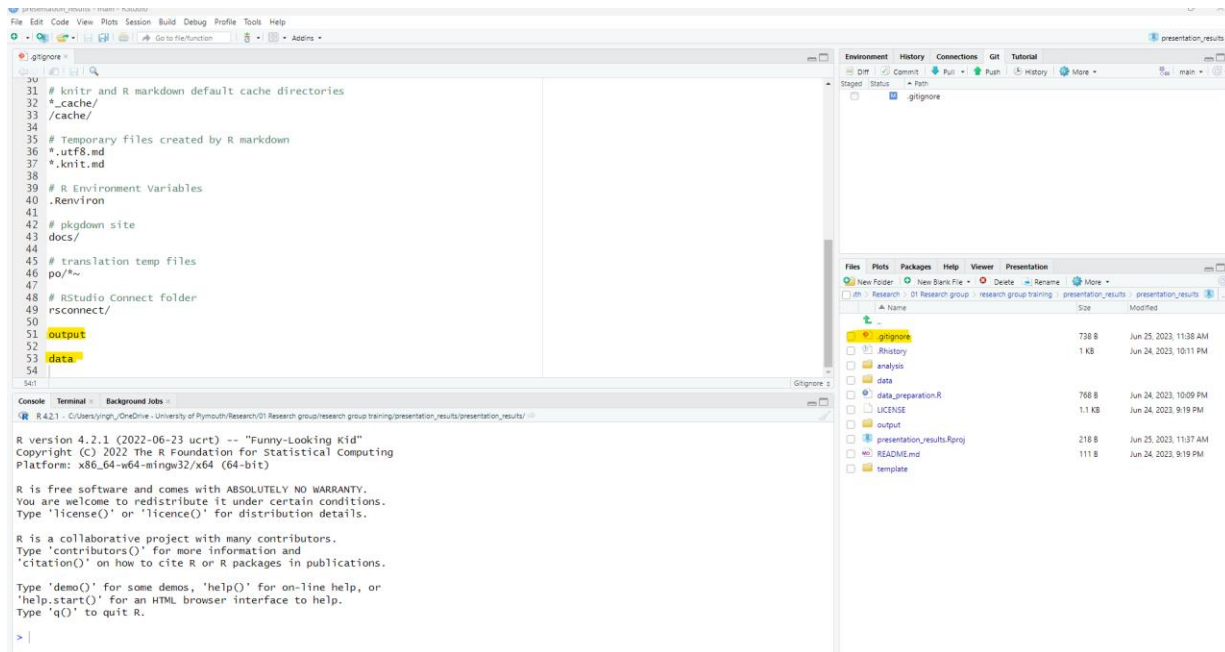
Step 6. You should see a tab “Git” on the top right window. “Commit” is the button where you store the changes you make in your local folder, and ready to be “push” to the github repo. “Pull” means you pull down from github repo to your local folder; “Push” means you push from your local folder to github repo. When you make changes, make sure you “commit” and “push” – so that your local drive and github repo are up-to-date. This is important especially when you are changing a computer (e.g. home / work). Before making any change to your local folder, always make sure you “Pull” from the github repo – otherwise, it can create a conflict between your local folder and the github repository.



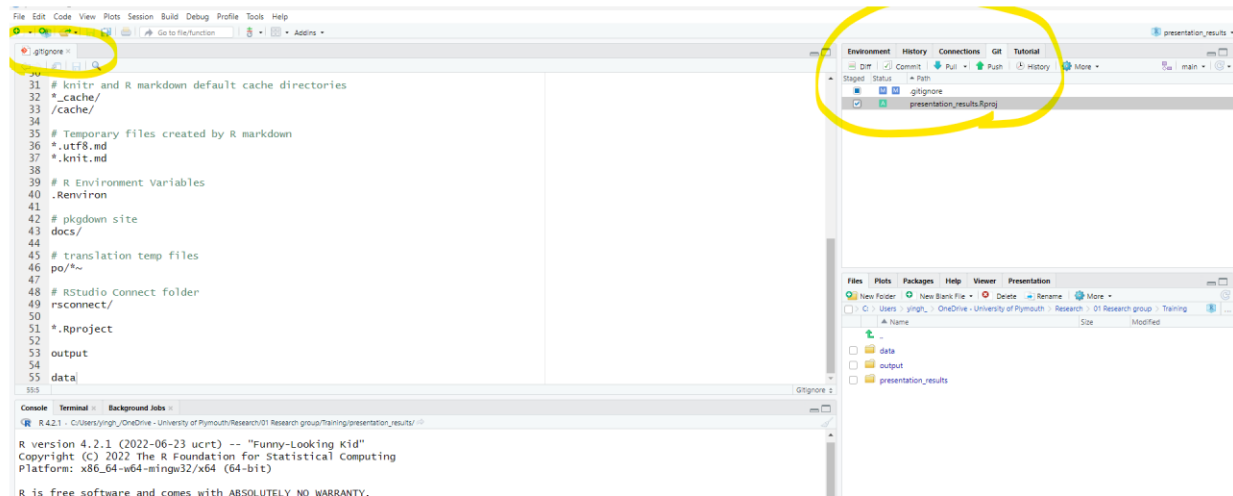
Step 7. Go one-step back to the parent directory, and create a folder “output”, and another folder “data” – this is to ensure that both of these folders are not committed to GitHub repo.



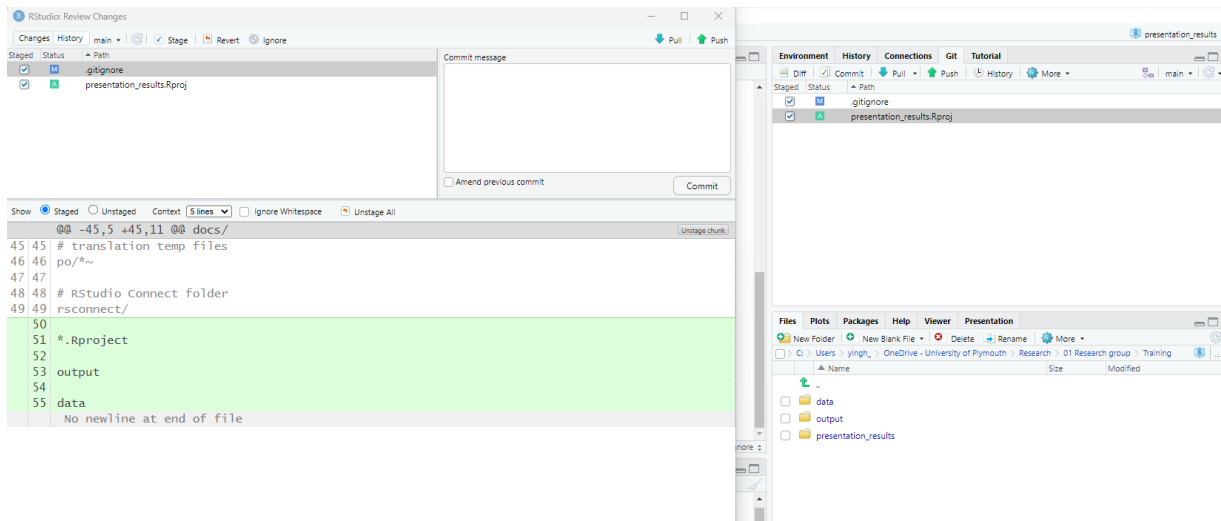
Step 8. gitignore: open this file (you can find it in the bottom right window), and type in output, and data. So any folder or files named after these will not be committed to GitHub repos. If you incidentally created “output” and “data” folders in the current directory, they will not be committed to github if you have specified “output” and “data” in gitignore – Be extremely careful, folder names “output” and “data” are case-sensitive, make sure the folder names and what you specified in gitignore are not mismatched – otherwise gitignore will take no effect.



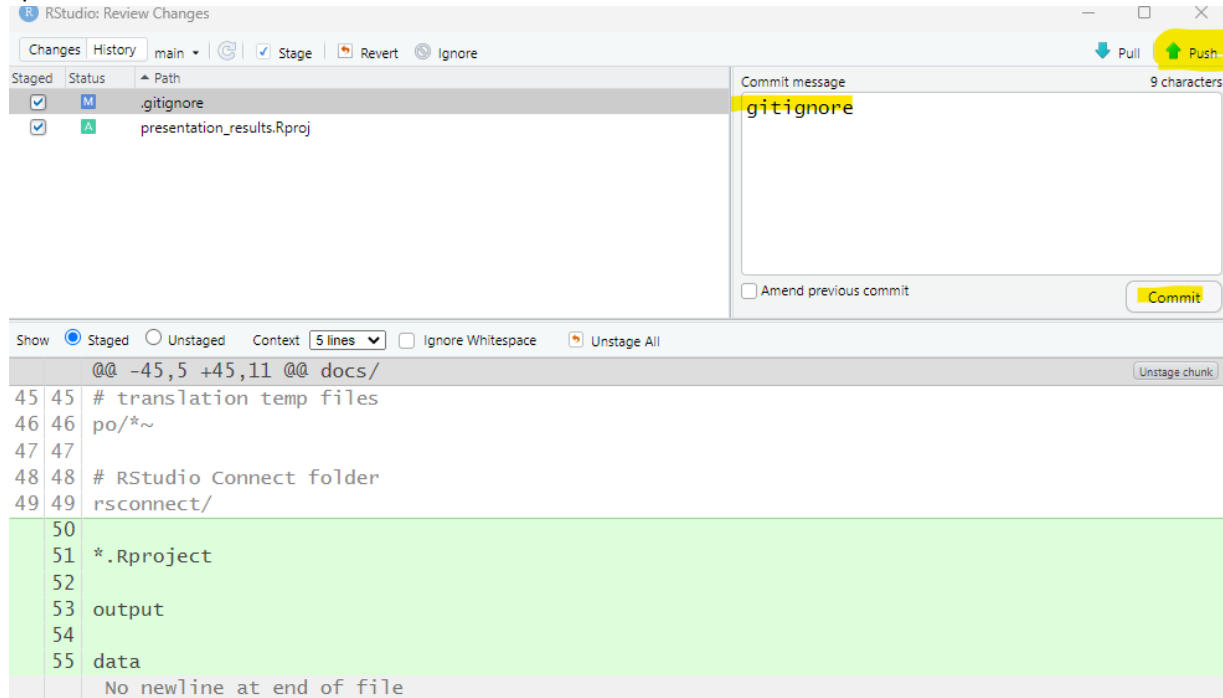
Step 9. Click “save” for your gitignore file – once you click on “save”, the star attached to the name of the script disappears. Always check whether there are items under Git – it is important that they are committed and push to Github – **but make sure you DO NOT push any data and output to GitHub! (see steps 7 and 8 for this).**



Select the files under Git, click “commit”



Type a short message in the commit box, click “commit” and then click “push”



Your tasks

Create a GitHub repository and link it to an R project in your local folder. Your example data set should come from the “survival” package below. Your folder should have all the features outlined in the steps above, and essential points are also highlighted below

- Your R project folder should include *.Rproj (which allows you to access your local folder directory directly)
- You must specify gitignore file correctly – type “output” and “data” in your igitnore file (see step 8, be extremely careful about spelling, directories, and upper/lower cases)
- output and data folders in the parent directory to your R project folder (to avoid being committed to GitHub).
- Make sure your R project folder and GitHub repos are up-to-date with each other.

```
library("survival")
data(diabetic, package="survival")
data <- diabetic
```