# Version control with Git and GitHub in R

Sean Hellingman©

Data Visualization and Manipulation through Scripting (ADSC1010)

*shellingman@tru.ca*

Fall 2024

THOMPSON RIVERS UNIVERSITY

## Topics

## Introduction

- We can use a version control system to keep track of our R code.
- Version control systems can also be used to collaborate with others.
- We can use RStudio to interface with Git.
  - The user interface in RStudio is easier to work with, but we will lose some of the functionality.
- We will also cover using GitHub (web-based hosting service).
  - GitHub is not designed to host very large files.

## Version Control

- A **Version Control System** (VCS) keeps a record of all the changes made to a particular project.
  - It then allows you to revert to previous versions of files if you need to.
- *If you badly mess things up, or accidentally lose important files you can revert to a previous stage of your project and fix your problems.*
- Originally designed for collaborative software development.
- There are many different version control systems available.
- We will focus on Git because it is easy to integrate into RStudio (also free and open source).

## Why Version Control

- Version control automatically takes care of keeping a record of the versions of a file and allows you to revert back to previous versions.
  - *If you find yourself making many copies (versions) of a project, this can help you.*
- It allows you to keep track of all of your files in a single location.
- It also allows collaborators (or reviewers) review, contribute, and reuse your work.
- If you include your work on the GitHub website your files can be available from anywhere on any computer (internet required).

## Git

- **Git** is a version control system that allows you to track changes of a set of files.
- These files can be any type of file including those we typically use in data science.
- All the files that make up a project is called a **repository** (repo).

## GitHub I

- **GitHub** is a web-based hosting service for Git repositories which allows you to create a remote copy of your local version-controlled project.
- Can be used for collaboration purposes or to archive your project.
- All the files that make up a project is called a **repository** (repo).
- Usually, at the start of a project we create a **remote repository** on GitHub.
- Then we **clone** (copy) this repository to our **local computer**.
  - Cloning is (usually) only done once and then you can work on your computer creating and saving files.

## GitHub II

- After we have made important changes we can take snapshots (**commits**) of our files.
- We then **push** our commits to the GitHub repository to make backups or make our changes available to collaborators.
- You can **pull** changes from the *repository* (project) back to a local machine.
  - Especially useful if other people are working on the same project (everything is synchronised).

## Install Git

- We can check to see if Git is already installed on our machine:
  - Click the Terminal tab in the Console window in RStudio and type:

```
git --version
```

- If Git is not installed we will need to install it:
  - Windows: *Git Bash*
  - Mac: *Installation Steps*
- *Sometimes Windows can have trouble linking RStudio and Git and we need to link it in RStudio (see textbook* (1)*)*

## Configure Git

- Once we know that Git is installed on our computer we need to configure it.
  - Click the Terminal tab in the Console window in RStudio and type:

```
git config --global user.email 'you@youremail.com'
git config --global user.name 'Your-Name'
```

- *Use your actual name and email.*

- Ideally, use your university email because we will use the same email to set up a GitHub account.

- If you were successful, when you type `git config --global --list` into the Terminal you should see your name and email.

## Configure RStudio

- We can use Git through the command line, but we can also integrate it with RStudio (friendly graphical interface).

- To use RStudio's Git integration:

  Tools ⇒ Global Options ⇒ Git/SVN

  - Make sure *Enable version control interface for RStudio projects* is selected.
  - Make sure the *Git executable:* path is correct (if not use *Browse...* to navigate to where Git is installed).
  - Restart RStudio if you made any changes.

## Register a GitHub Account

- *If you only want to keep track of your file versions on your computer Git is sufficient.*
- If you want to host your project (off-site) or make it available to collaborators you will need a web-based hosting service for your Git repositories.
- You can sign up for GitHub HERE
- Some other options: GitLab, Bitbucket or, Savannah

## GitHub Account Recommendations

- Use your university email address as it will allow you to apply for an educator/researcher account.
- Username:
  - Choose a shorter username (ideally including your actual name).
  - Use all lowercase letters (hyphenate multiple words).
  - **Choose a username that you will feel comfortable sharing with an employer.**
- Select the *Free Plan* option and finish your registration.
- You should now be ready to use Git and GitHub.

## Project Creation

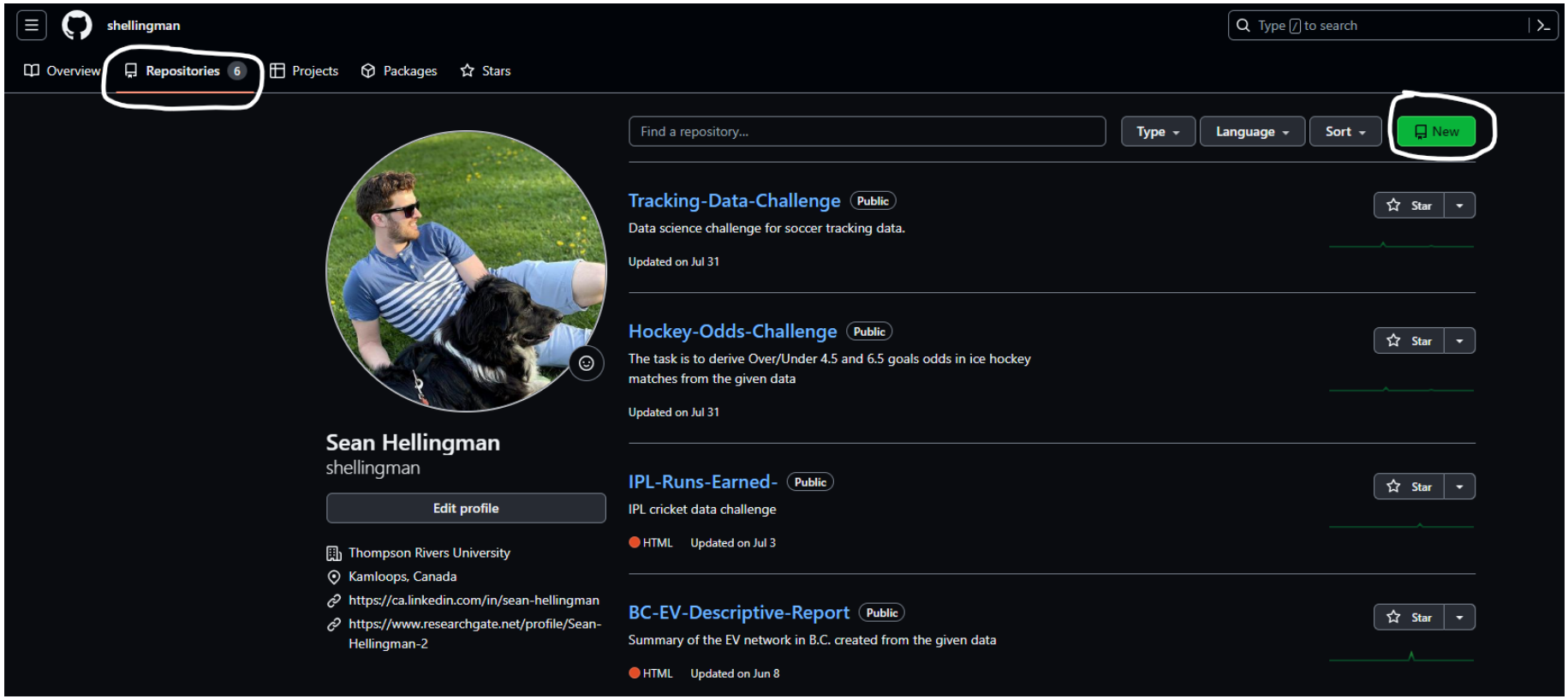
- Now that we have Git and GitHub set up we can create repositories and use version control.
- There are two main ways to do this:
  1. GitHub first
  2. RStudio first

# 1. GitHub First

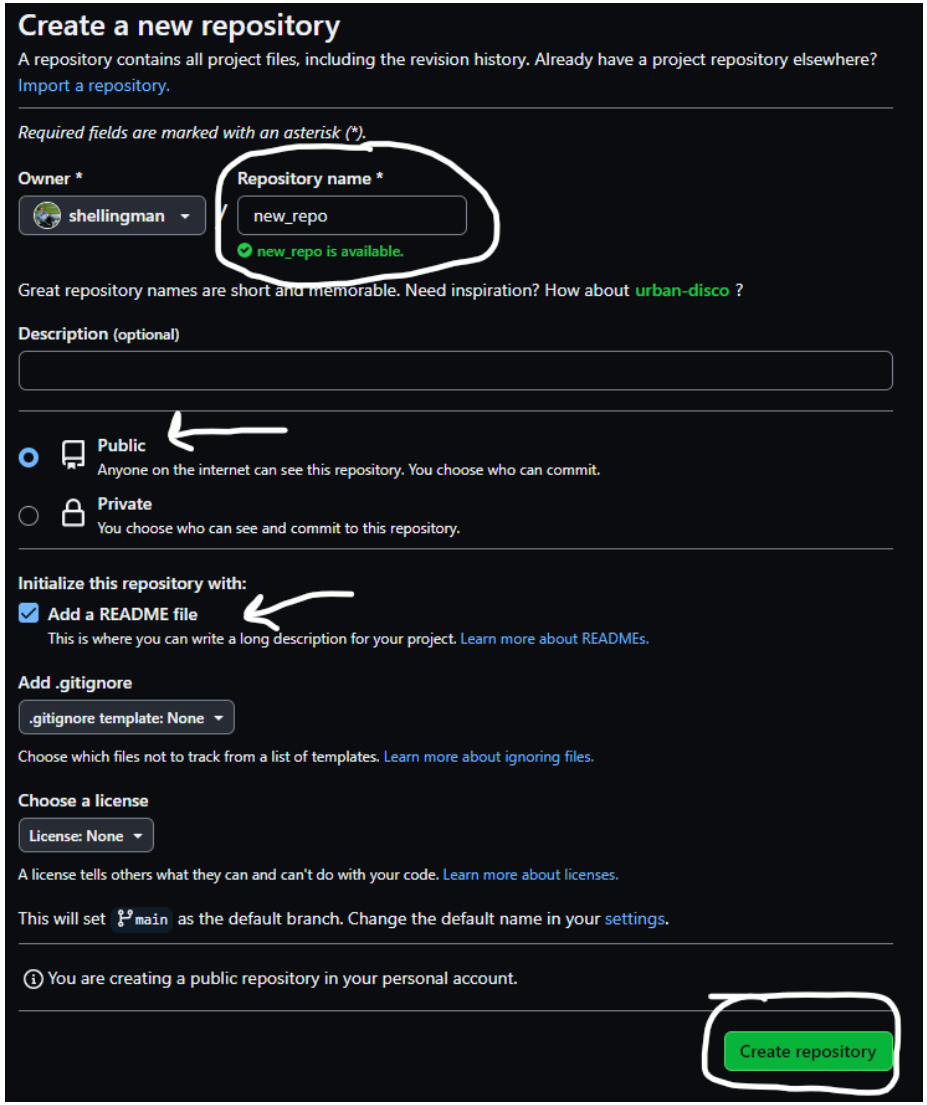
## GitHub First I

- We need to start by creating a new **repository (repo)** on our GitHub page.
- Go you your GitHub page ⇒ Click Repositories ⇒ click the green New button.
- Give your new repo a name ⇒ select Public ⇒ select Initialize this repository with a README ⇒ click Create repository.
- Your new GitHub repository will now be created.
- The README file is in Markdown format.
- Next, click on the green Code button ⇒ copy the HTTPS URL to your clipboard.

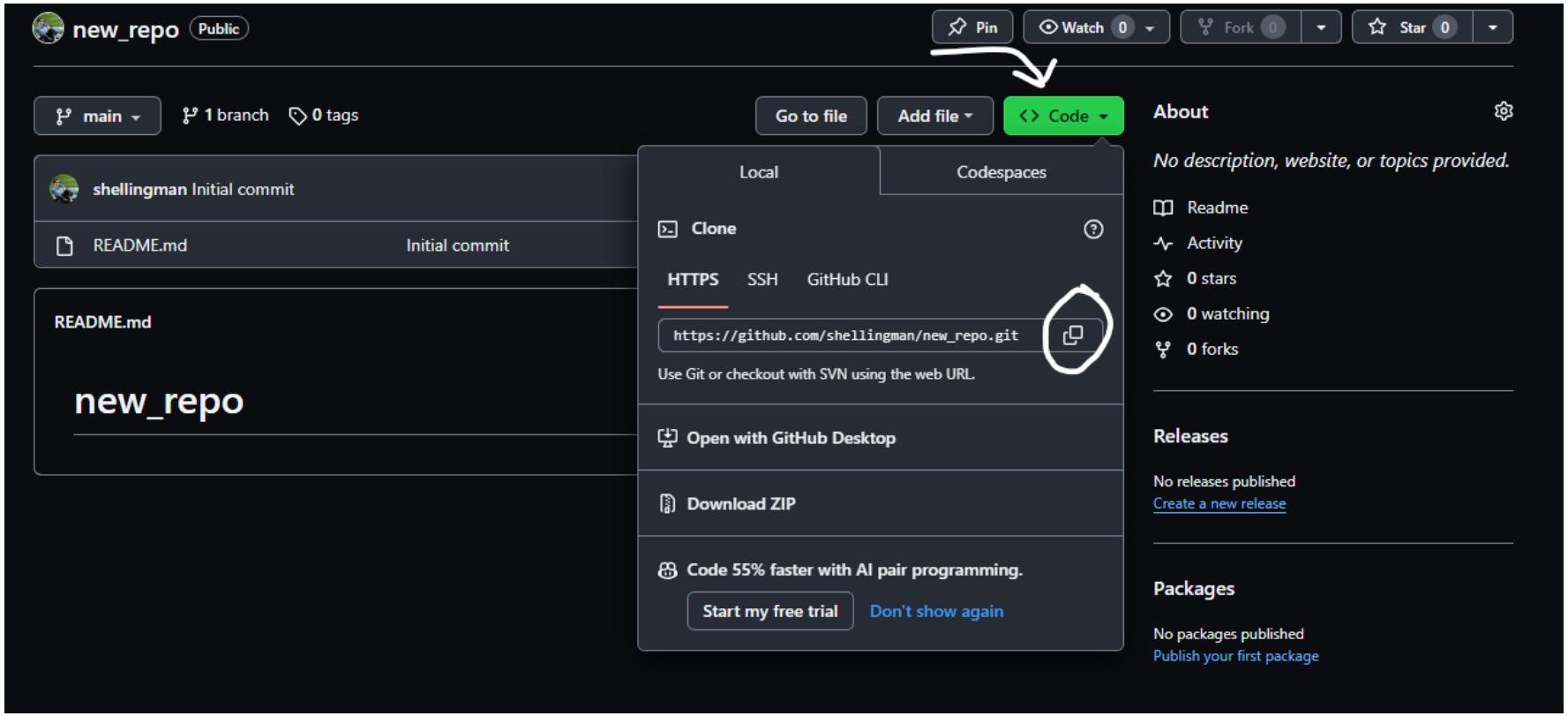
# Create a New Repo



# New Repo Settings



# Copy Link


new_repo
Public
Pin
Watch 0
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
shellingman Initial commit
README.md
Initial commit
README.md
new_repo
Local
Codespaces
Clone
HTTPS
SSH
GitHub CLI
https://github.com/shellingman/new_repo.git
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP
Code 55% faster with AI pair programming.
Start my free trial
Don't show again
About
No description, website, or topics provided.
Readme
Activity
0 stars
0 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

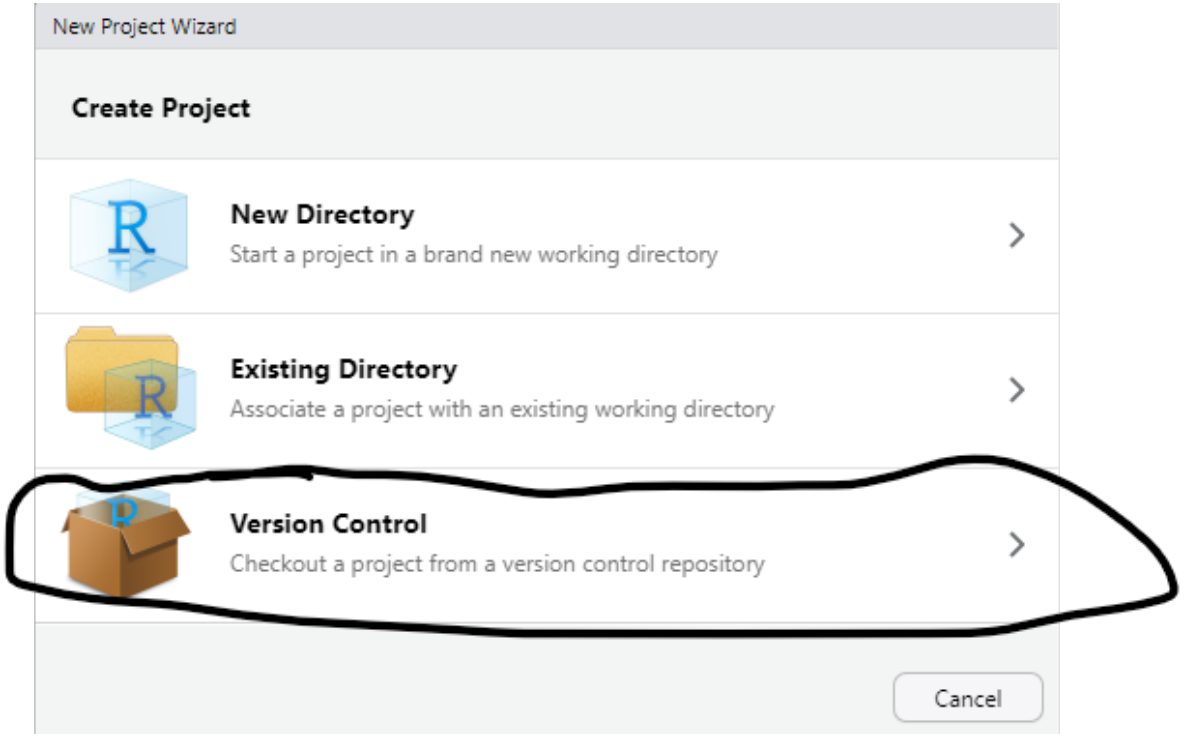
## GitHub First II

- Now we can open RStudio.
- Click File ⇒ New Project menu ⇒ select Version Control from the options.
- Paste your URL from GitHub into the Repository URL: box. Make sure you match the Project Directory Name with the repo name you used in GitHub.
- Select a location for your directory on your own computer ⇒ select Open in new session ⇒ create your new project.

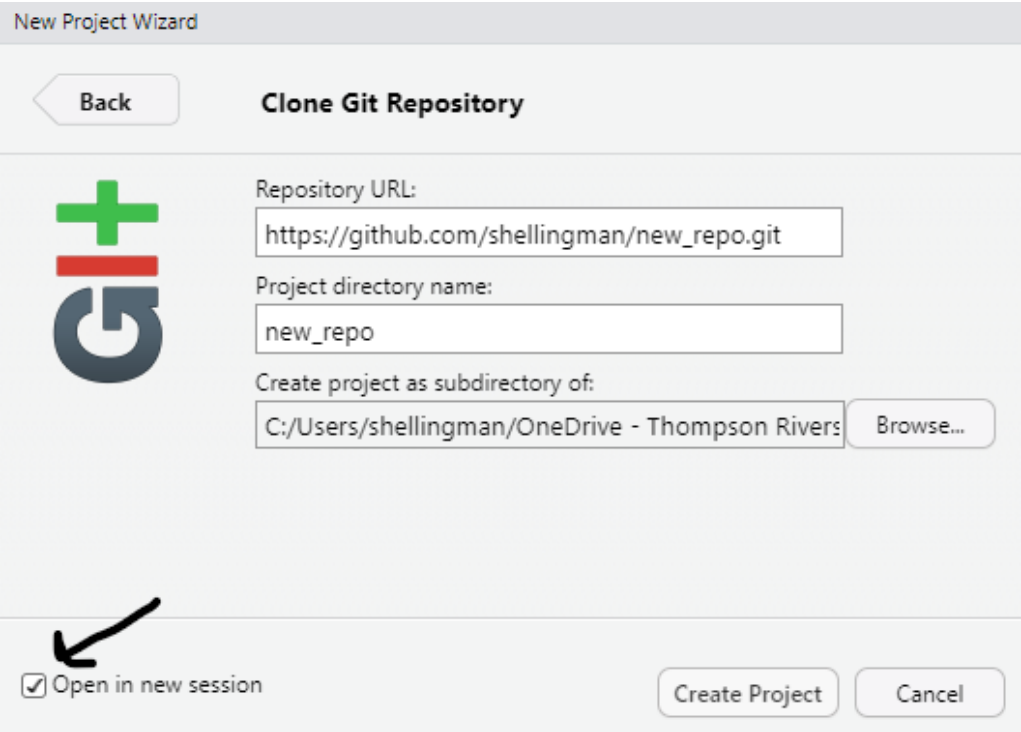
## GitHub First III

- RStudio will create a new directory with the same name as your repository (local machine) and will **clone** your remote repository to this directory.
- The directory will now contain three files: `repository_name.Rproj`, `README.md`, and `.gitignore`
- We will also get a *new* Git tab in the top right pane.
- Any changes you make to files in the directory will be controlled by Git.
- To disconnect select the repository name in the top right corner and click Close Project.

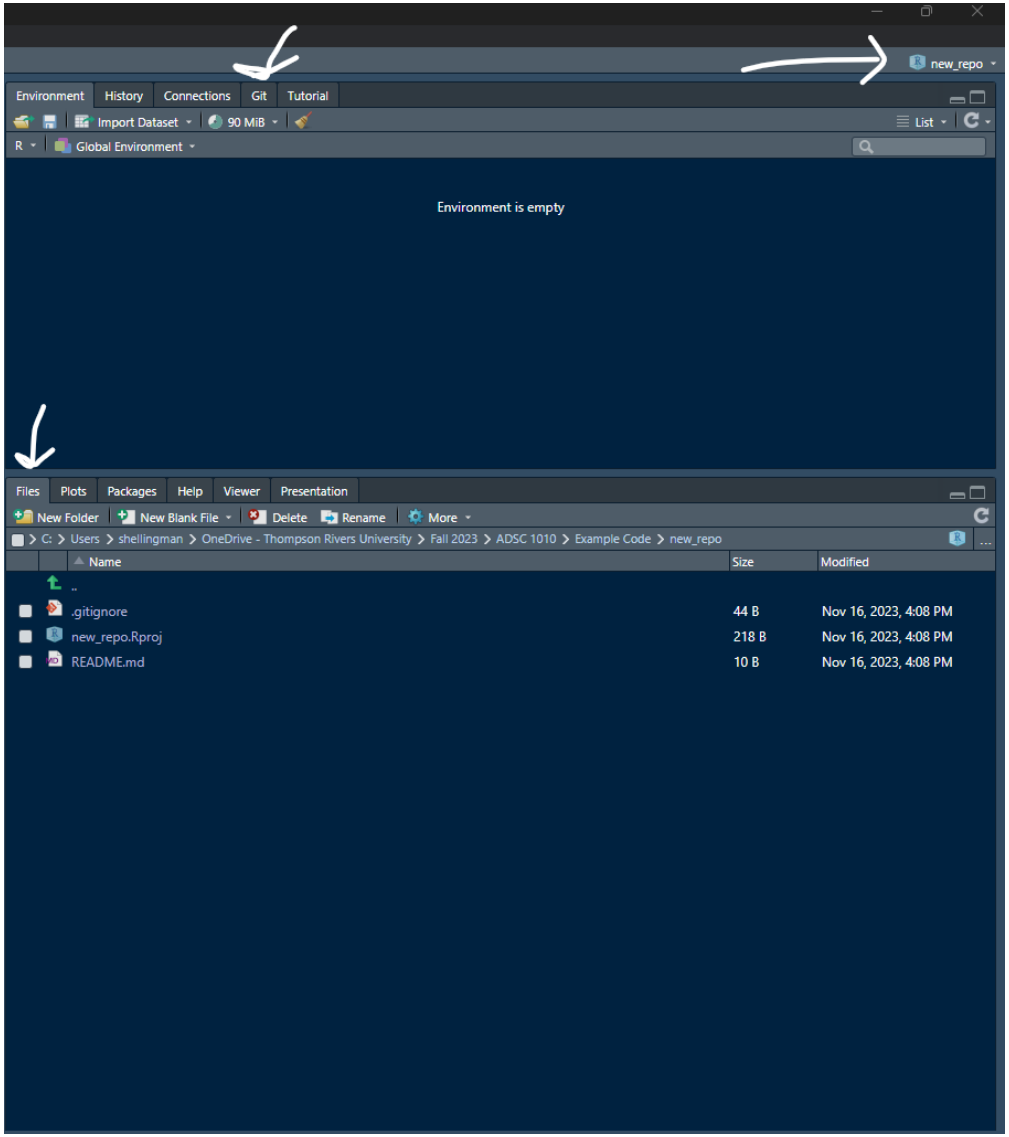
# Create a New Project



# Connect to GitHub



# Your Project

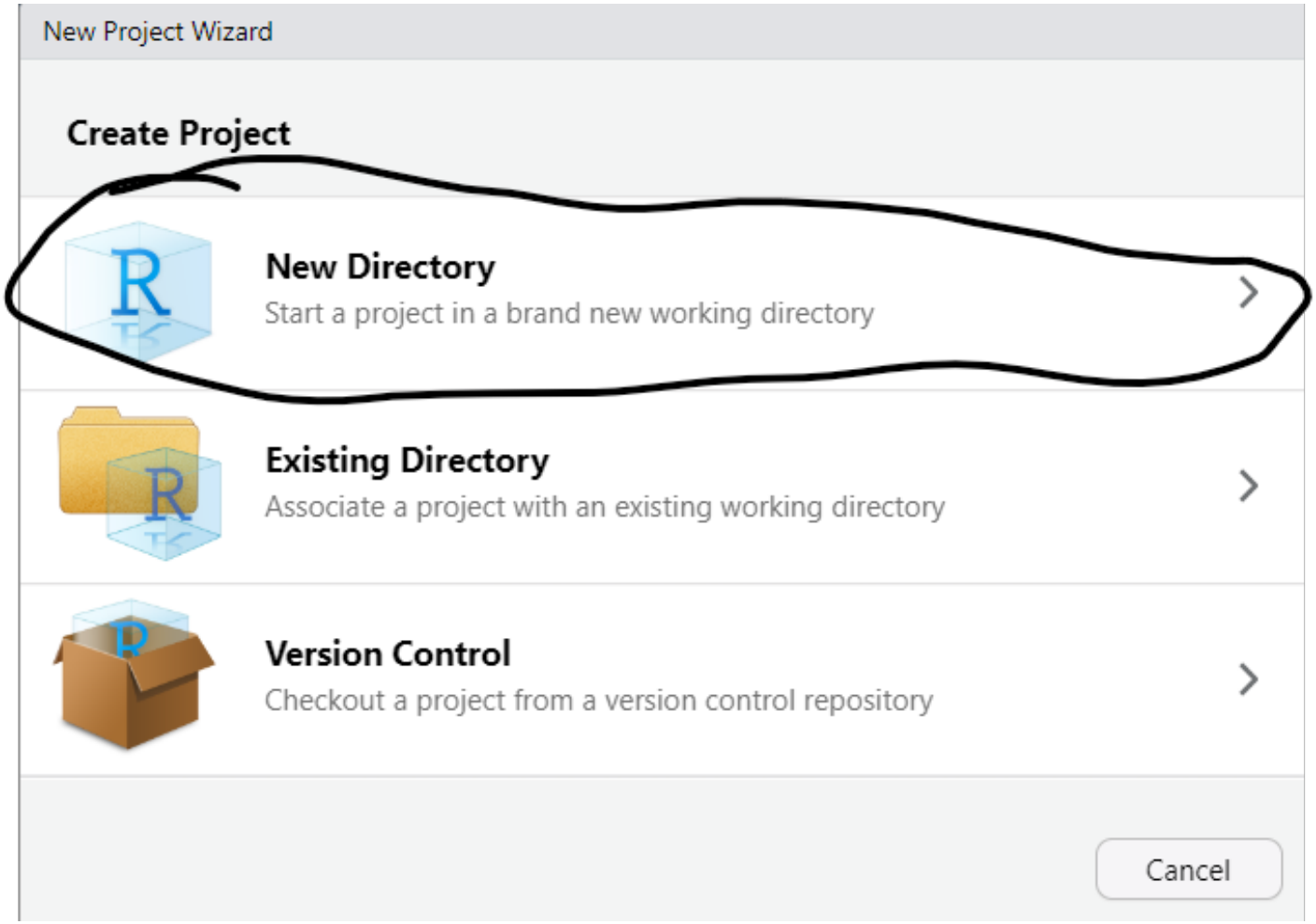


## 2. RStudio First

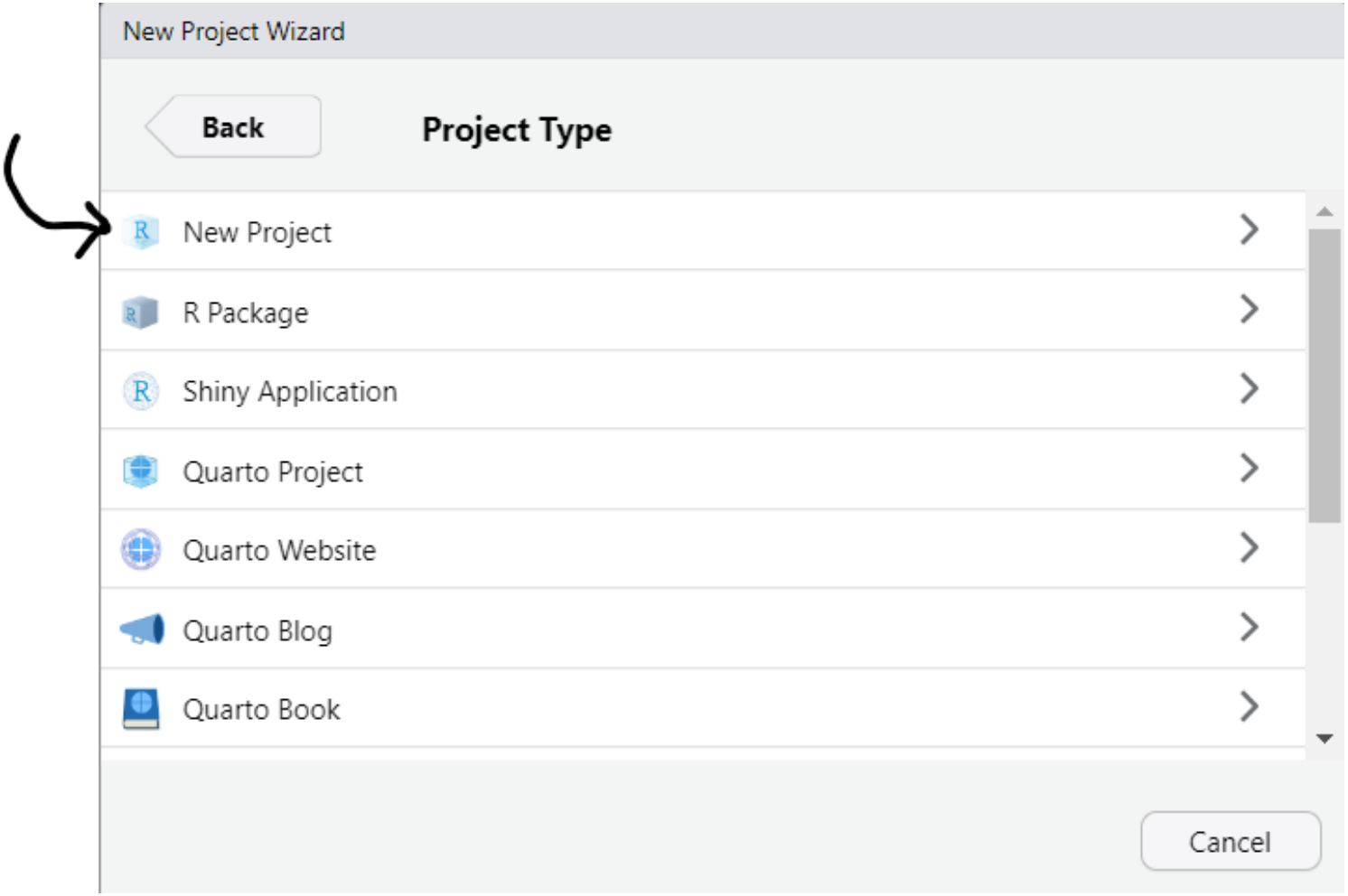
## RStudio First I

- We need to start by creating a new local RStudio project then we can link it to a remote repo.
- This option is a bit more involved than the other way, but this can be used to set up local version control.
- File ⇒ New Project ⇒ select New Directory ⇒ click New Project.
- In the new Project Window make sure to specify a Directory name and make sure the Create git repository option is ticked.
- This will create a version controlled directory on your computer.

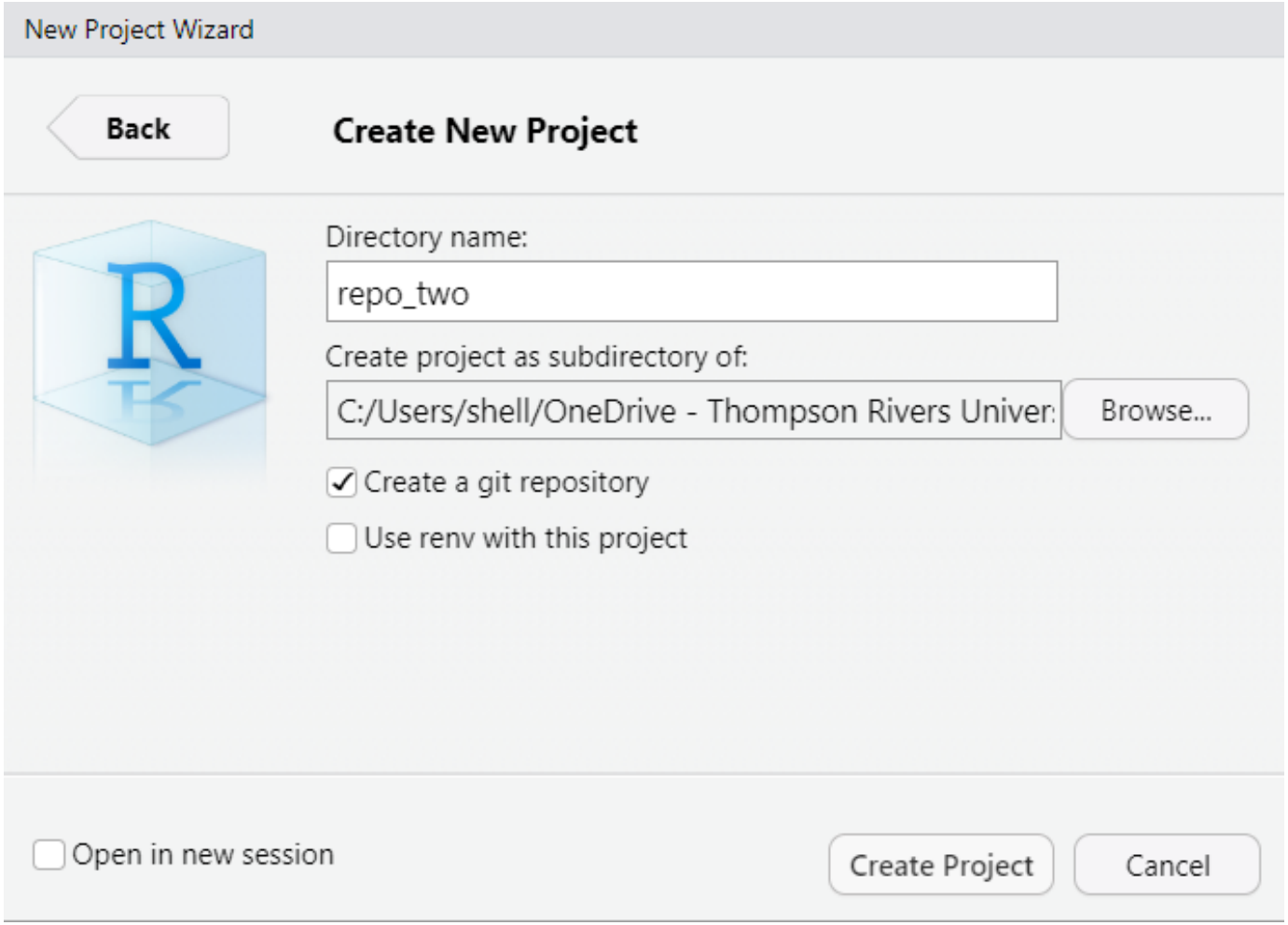
# Create a New Directory



# Select Project



# Create a new Repository



## RStudio First II

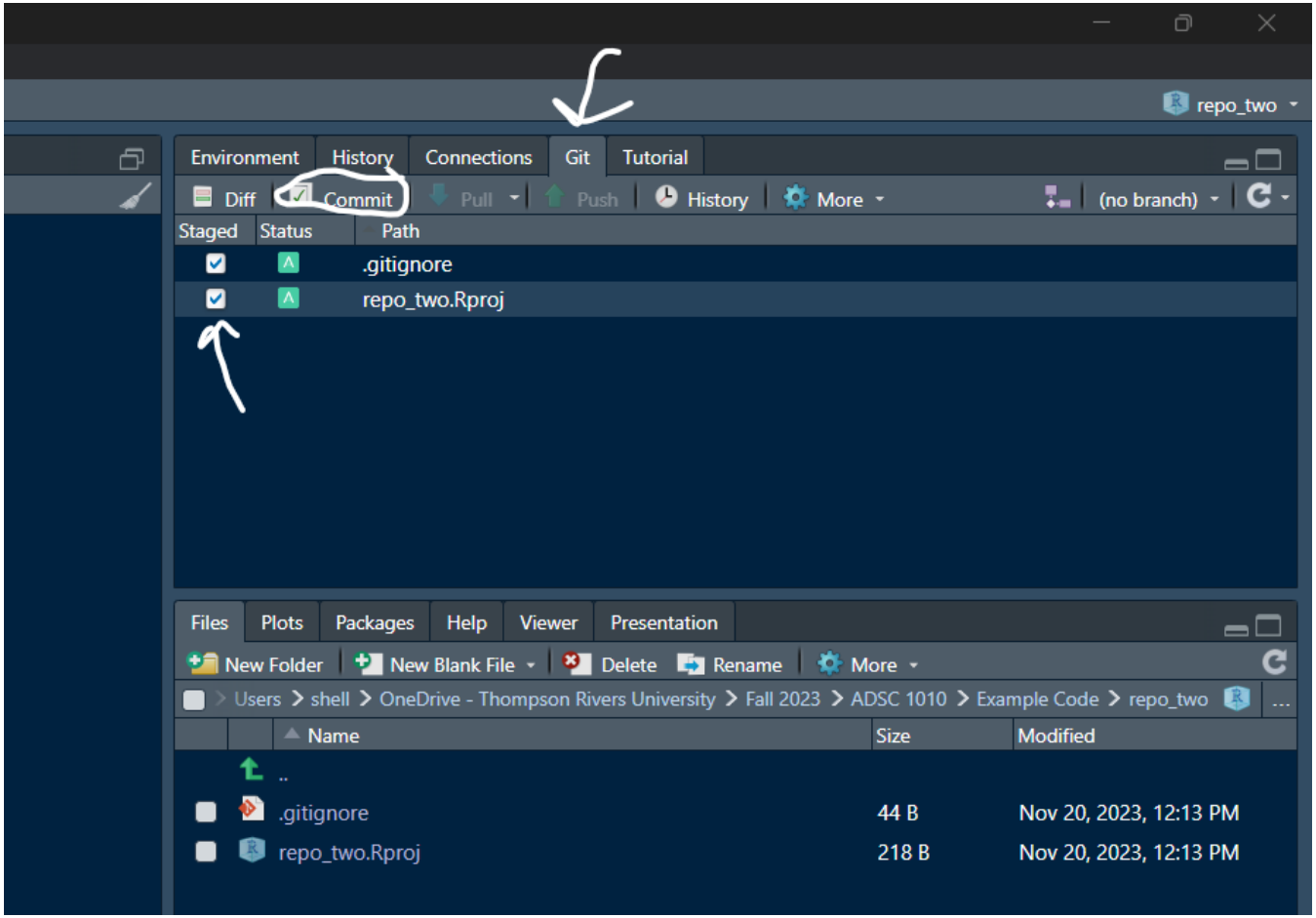
- Before you create a repository on GitHub you will need to place the `.gitignore` and `repo_two.Rproj` files in version control.
  - *We will cover how this works in more detail later on.*

- Click the Git tab ⇒ tick the boxes under the Staged column ⇒ click Commit. This leads to a Review Changes window.

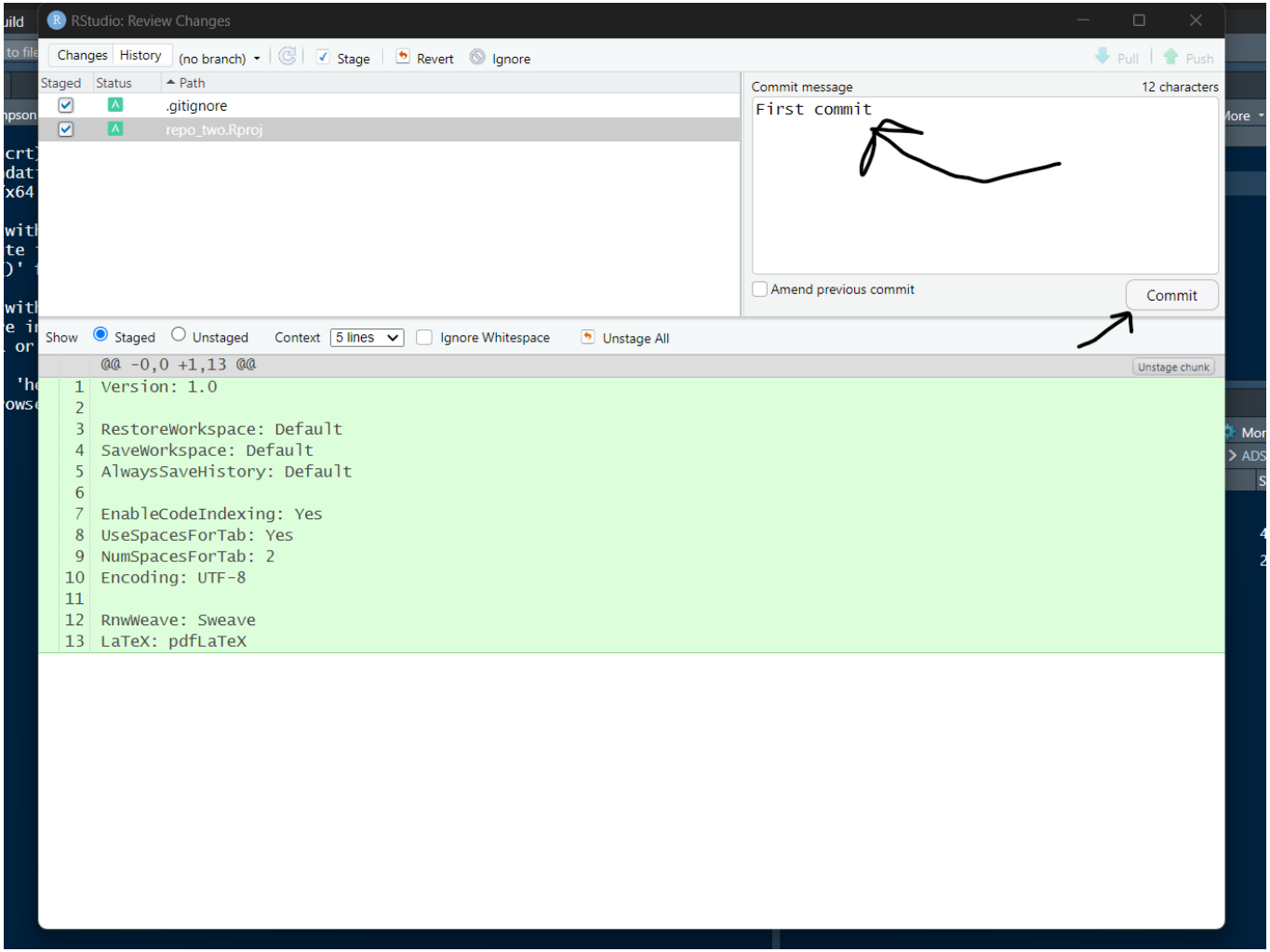
- Type in the commit message `First commit` in the Commit message window ⇒ click on the Commit button. (For now) close the new window.

- The files should now have disappeared from the Git tab in RStudio.

# Version Control



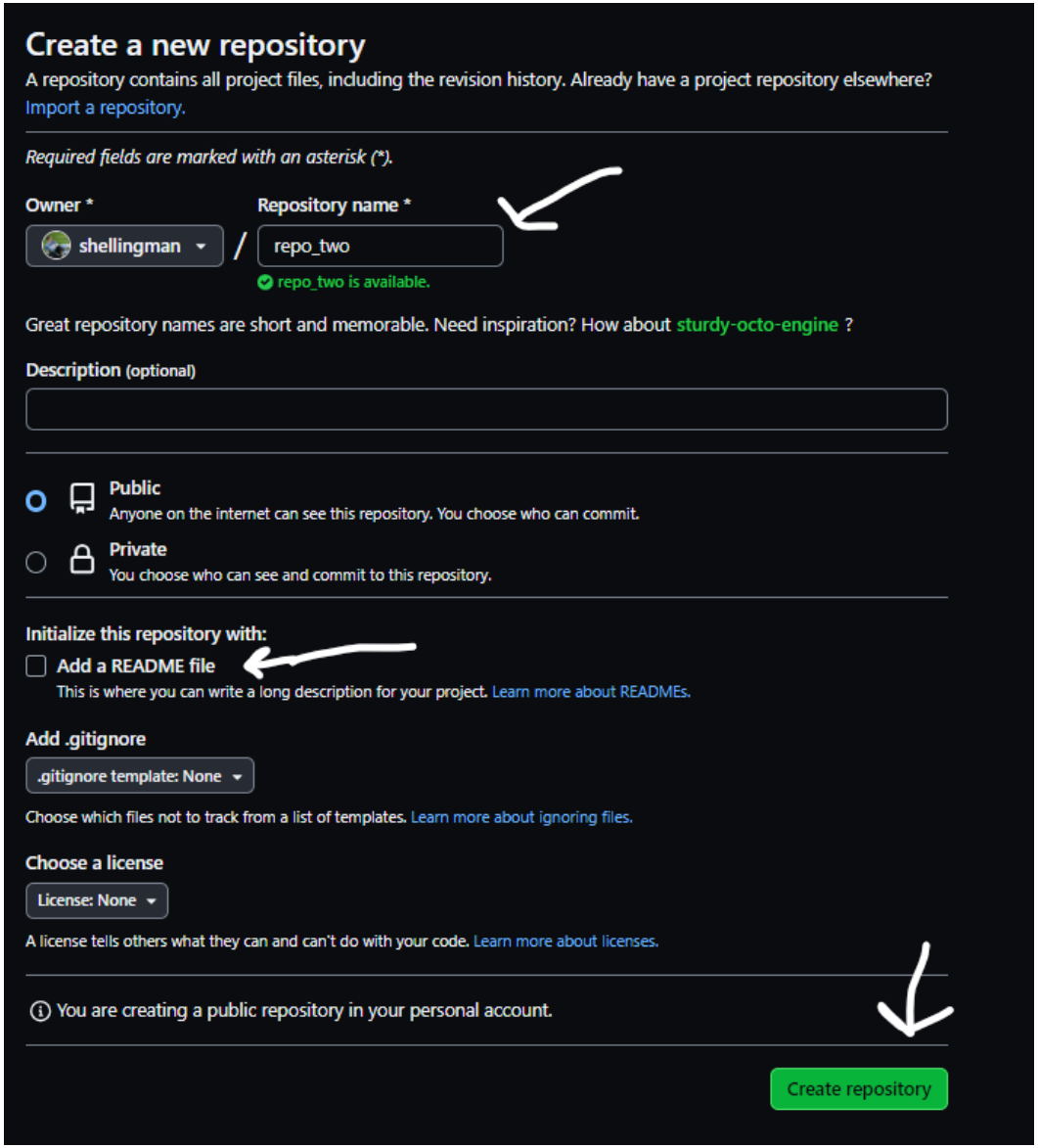
# First Commit



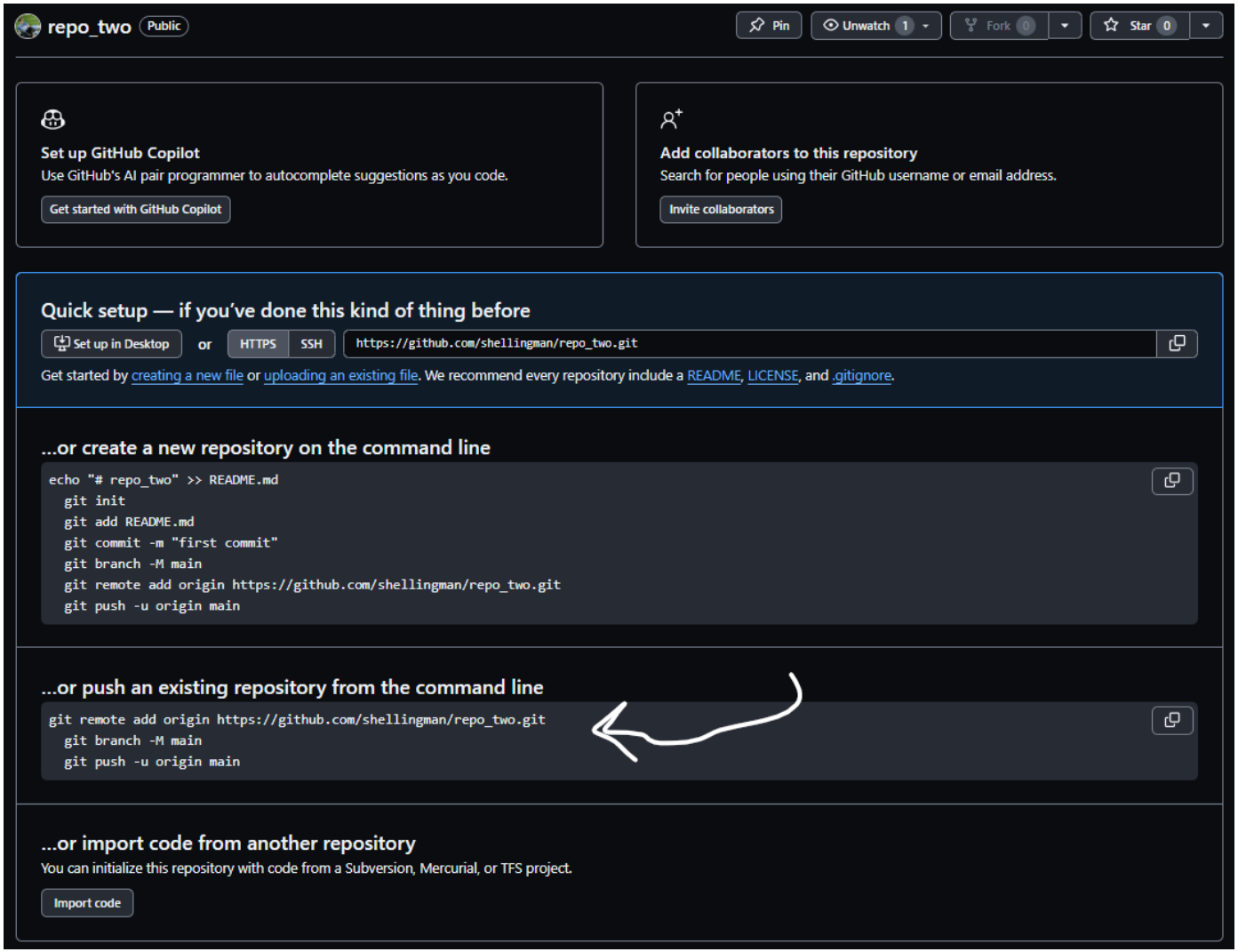
## RStudio First III

- Next, go to your GitHub page ⇒ click Repositories ⇒ click New ⇒ and name your repo the same name as you gave it in RStudio.
- This time **do not** tick Initialize this repository with a README ⇒ click Create repository.
- In the Quick setup page we are interested in the code under the `...or push an existing repository from the command line` heading.
  - Copy the code and paste it into your Terminal tab in RStudio (run it).
- The files should now have been **pushed** to your GitHub from your local repository.

# New GitHub Repository



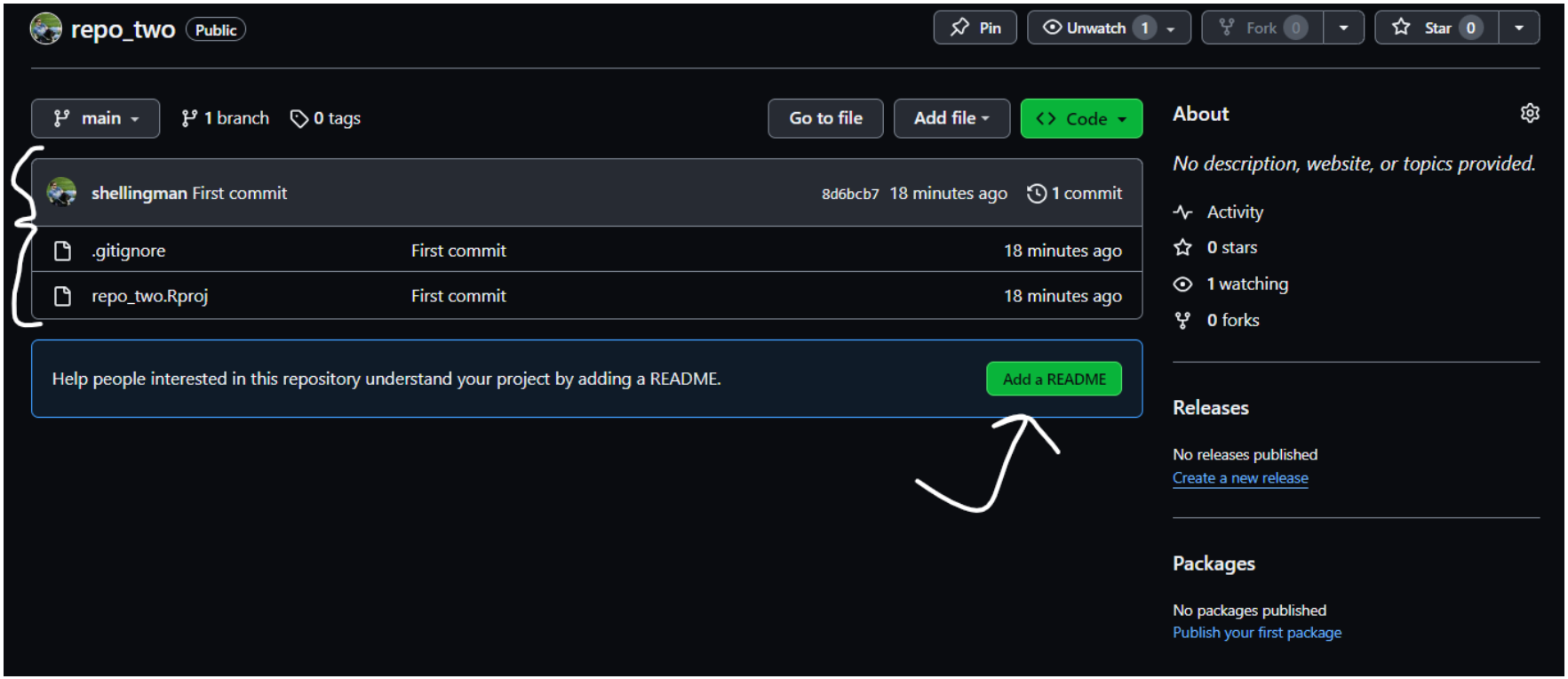
# Push Code


repo_two
Public
Pin
Unwatch 1
Fork 0
Star 0
Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/shellingman/repo_two.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 "# repo_two" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/shellingman/repo_two.git
git push -u origin main
...or push an existing repository from the command line
git remote add origin https://github.com/shellingman/repo_two.git
git branch -M main
git push -u origin main
...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

# RStudio Terminal

```
Console   Terminal   Background Jobs
Terminal 1
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shell\OneDrive - Thompson Rivers University\Fall 2023\ADSC 1010\Example Code\repo_two>git remote add origin https://github.com/shelling
man/repo_two.git

C:\Users\shell\OneDrive - Thompson Rivers University\Fall 2023\ADSC 1010\Example Code\repo_two>  git branch -M main

C:\Users\shell\OneDrive - Thompson Rivers University\Fall 2023\ADSC 1010\Example Code\repo_two>  git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 437 bytes | 437.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shellingman/repo_two.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

C:\Users\shell\OneDrive - Thompson Rivers University\Fall 2023\ADSC 1010\Example Code\repo_two>
C:/Users/she
```

# Final Results



## Comments

- Be sure to add a README.md file if you create your repository locally first.
- It is usually easier to create your project in GitHub.
- Now we are ready to use Git with RStudio!

# Using Git

## Git Workflow

- Generally your *Git* workflow will look something like this:

1. Create, delete, and/or edit files in your project directory on your computer as usual (saving as you go is recommended).
2. You **stage** your files once you reach a natural *break point* (you do not want to lose this progress).
3. You **commit** the changes made to the staged files which creates a permanent *snapshot* of these changes (be sure to include a useful commit message).
4. Keep working with this cycle until you are ready to **push** your changes to GitHub.
5. If you are collaborating with other people, you will need to **pull** their changes to your local computer.

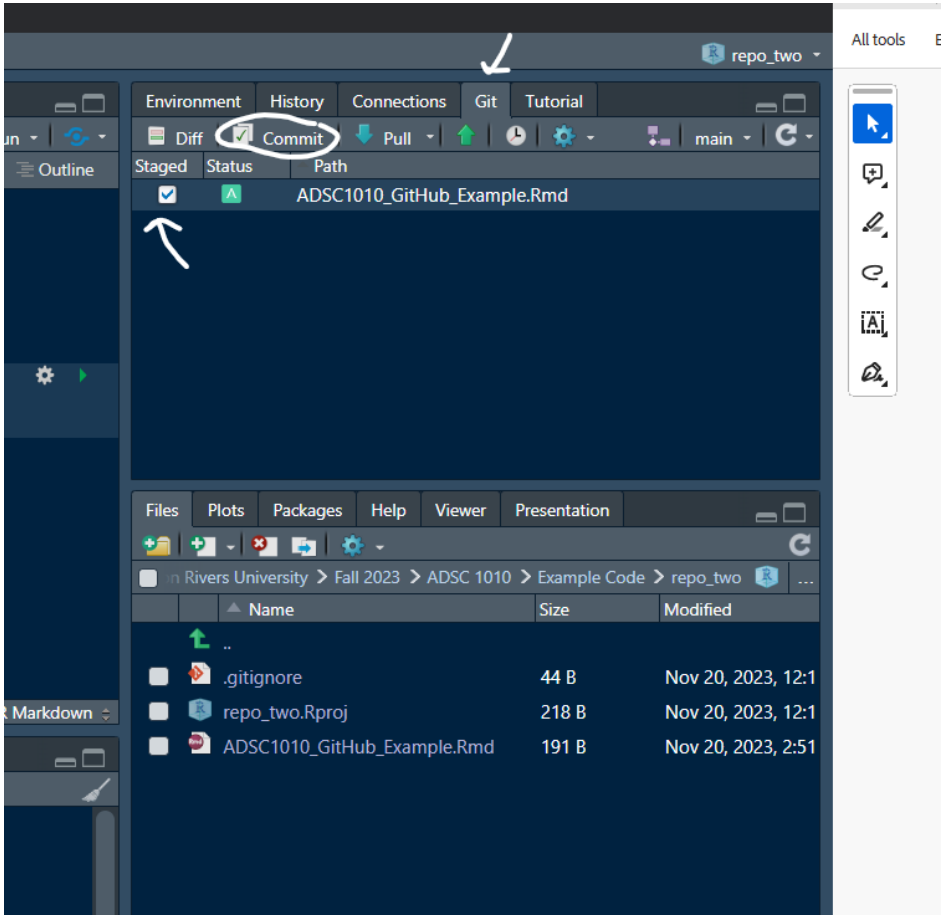
## Git Workflow in Practice Example I

- In RStudio, open the project you want to work on under version control using Git.
- Create a new R markdown file within this project. ⇒ Click the Git tab (your markdown file should be included).
- Following the Git workflow: **stage** the files by selecting the boxes under the staged column (all files) A status icon will appear beside the staged files ⇒ **commit** the files by selecting the Commit button (the review changes window should open).
- If you select your .Rmd file all changes will be highlighted in the bottom pane (new content in green, deleted content in red)

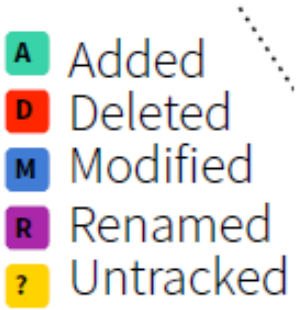
## Git Workflow in Practice Example II

- To commit you changes add a mandatory message to the commit message box.
  - This message (to your collaborators or future self) should indicate **why** you made the changes, not what changes you did make.
- Click the Commit button to commit your changes, a summary of your changes should appear.
- It is a good idea to **pull** any changes from GitHub before pushing any changes (especially if you are collaborating with others).
- Click the Pull button on the top right of the Review Changes window. You can then **push** your changes to GitHub using the Push button ⇒ close the windows ⇒ check that you files have been committed to GitHub.

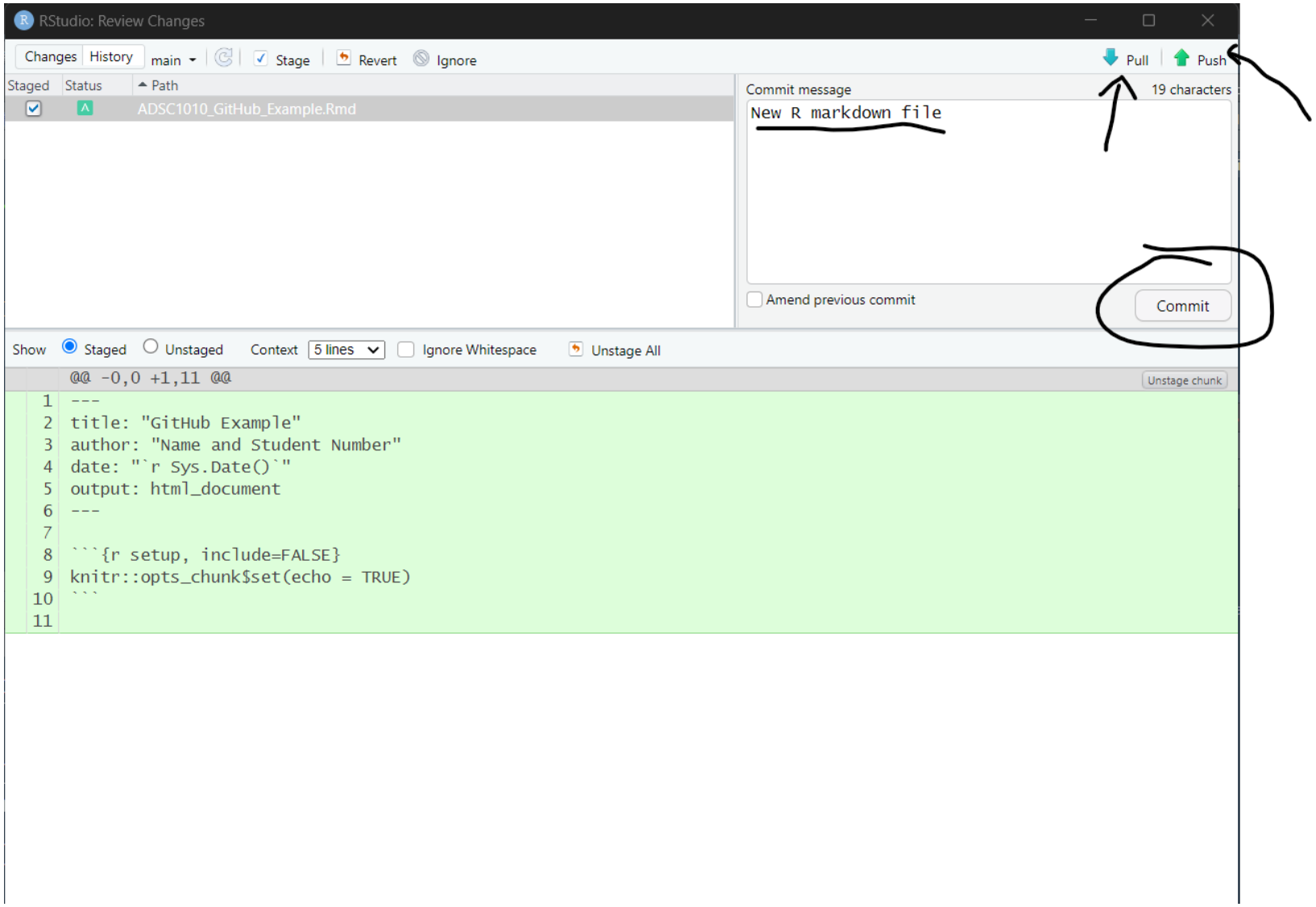
# New Commit


repo_two
All tools
Environment
History
Connections
Git
Tutorial
Diff
Commit
Pull
main
Outline
Staged
Status
Path
ADSC1010_GitHub_Example.Rmd
Files
Plots
Packages
Help
Viewer
Presentation
Rivers University > Fall 2023 > ADSC 1010 > Example Code > repo_two
Name
Size
Modified
..
.gitignore
44 B
Nov 20, 2023, 12:1
repo_two.Rproj
218 B
Nov 20, 2023, 12:1
ADSC1010_GitHub_Example.Rmd
191 B
Nov 20, 2023, 2:51
Markdown

# Status Icons



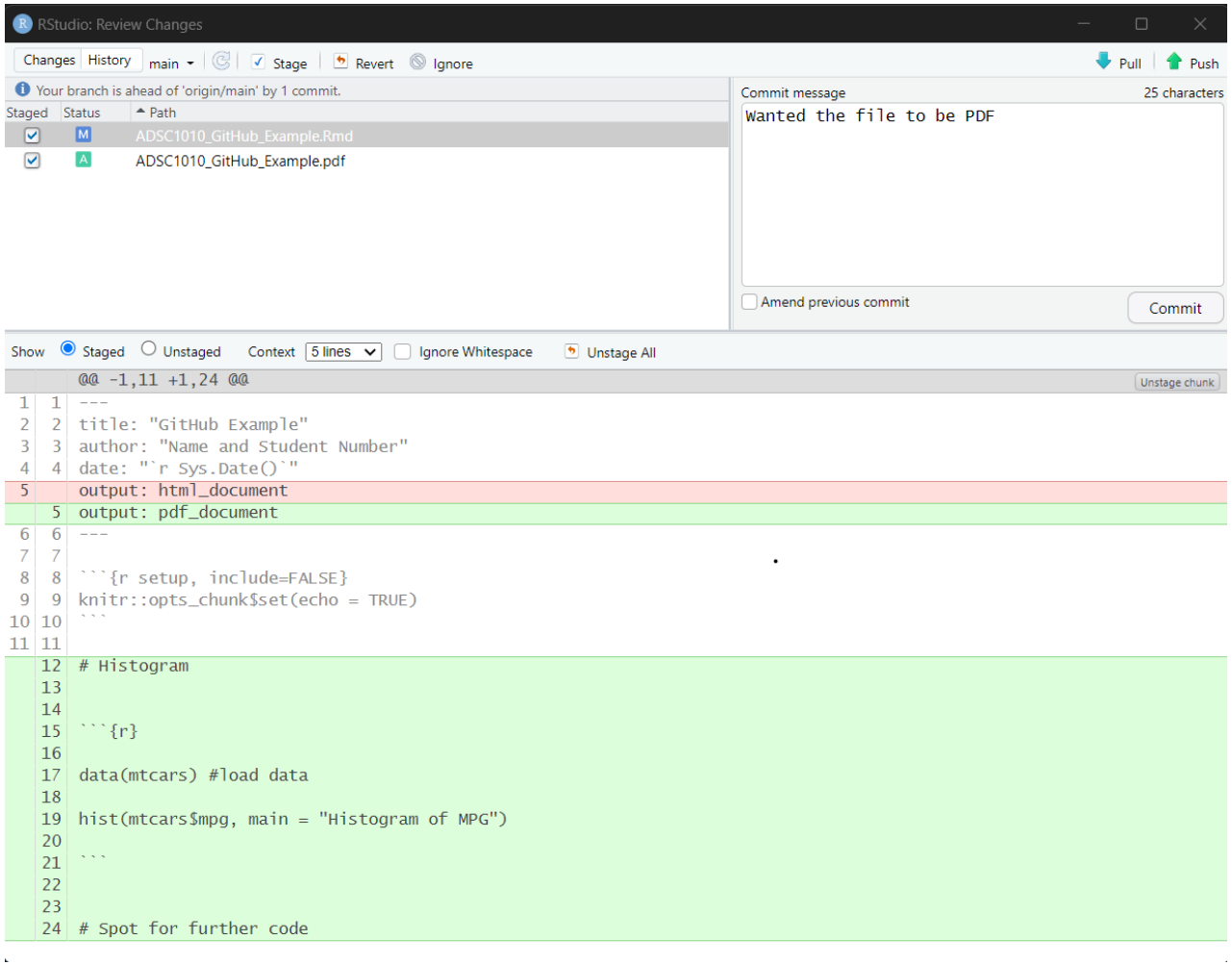
# Push


RStudio: Review Changes
Changes
History
main
Stage
Revert
Ignore
Pull
Push
Staged
Status
Path
ADSC1010_GitHub_Example.Rmd
Commit message
19 characters
New R markdown file
Amend previous commit
Commit
Show
Staged
Unstaged
Context
5 lines
Ignore Whitespace
Unstage All
@@ -0,0 +1,11 @@
Unstage chunk
---
title: "GitHub Example"
author: "Name and Student Number"
date: "`r Sys.Date()`"
output: html_document
---
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## Tracking Changes Example

- Simply make changes to your R markdown document and save them as you go.
- Knit your R markdown file to a by pdf by changing the header `output: pdf_document` (this will create a new file).
- Notice that the two files have been added to the Git tab in RStudio.
  - Their status icons will have also changed.
- When you click the Commit button you can see all the changes that you have made to the document ⇒ add an appropriate commit message ⇒ commit your files.
- Perform the same Pull/Push sequence as before and the new commits should be added to your GitHub.

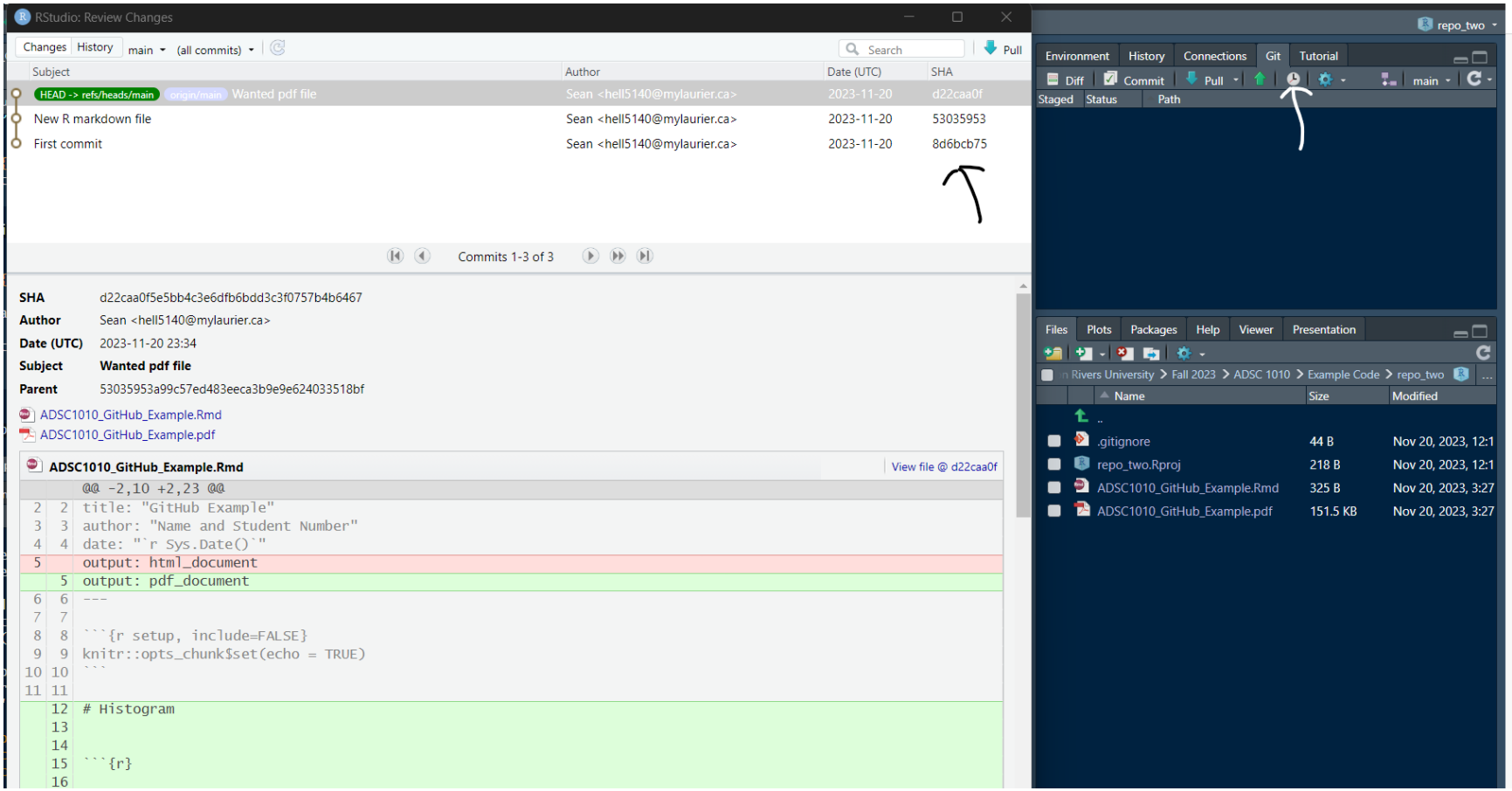
# Tracking Changes



## Commit History

- You can view the history of all of your commits (locally or in GitHub)
- In RStudio click the *View history of previous commits* option under the Git tab.
- The history is split into two panels:
  1. The top pane has a list of all of your commits (newest to oldest).
  2. Bottom pane shows all the changes you have made.
     - The **Secure Hash Algorithm** (SHA) identifiers are used to revert to previous versions of files.
- In GitHub, go to your repository and click the commits link (the SHA identifiers are the same).

# Commit History



## Reverting Changes

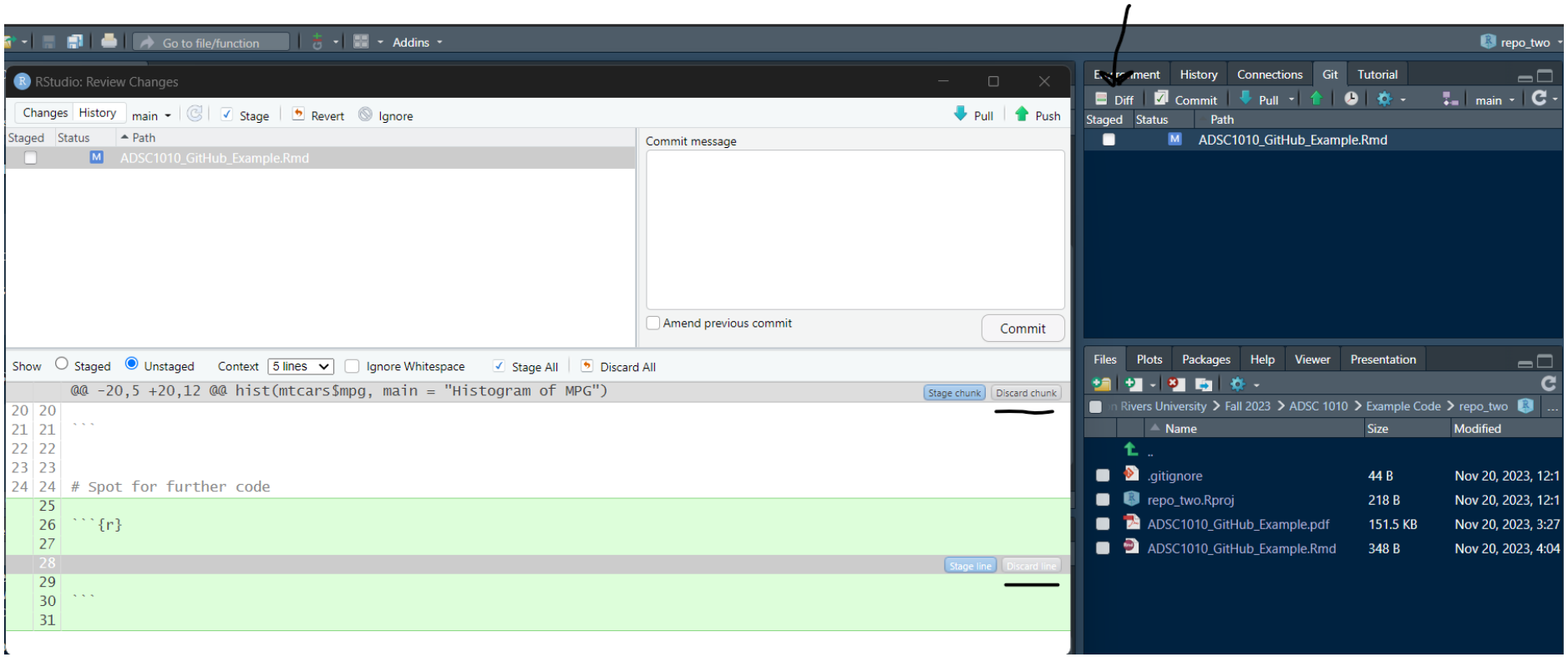
- You can use the functionality of Git to revert to previous versions of your files.
- How you do this depends on on whether the changes you want to discard have been staged, committed or pushed to GitHub:

1. Changes saved but not staged, committed or pushed.
2. Staged but not committed and not pushed.
3. Staged and committed but not pushed.
4. Staged, committed and pushed.

## 1. Changes saved but not staged, committed or pushed

- In RStudio, right click on the file you wish to revert (in the Git tab) $\Rightarrow$ select *Revert...*
- This will revert that file back to your previous commit.
- You can also select the Diff option and discard specific lines or chunks of changed files.
- **You cannot undo this, so proceed with caution.**

# 1. Changes saved but not staged, committed or pushed



## 2. Changes staged but not committed and not pushed

- Unstage your changes by clicking (remove the tick) the Staged check box in the Git tab.
- Then follow the same steps as 1.

## 3. Changes staged and committed but not pushed

- If the mistake is in your last commit, you can simply fix your mistake, save your changes, and then click the Amend previous commit tab to delete your previous commit (**Do not do this if you have pushed your changes)**.

- If you want to revert further back:
  1. Look in your commit history in RStudio, find the commit that you would like to go back to and click on the View file @ button to show the file contents ⇒ copy and paste the contents into a new file and save it ⇒ stage and commit this file.

  2. Go to Git history ⇒ find the commit you want to revert to ⇒ copy the SHA identifier ⇒ in the Terminal in RStudio type `git checkout <SHA> <filename>` (Example `git checkout 2b4693d1 first_doc.Rmd`)
     - Use `git checkout 2b4693d1 .` to revert all of your files to this commit.

## 4. Changes staged, committed and pushed

- We can follow the steps in part 3. and just push the reverted commit to GitHub (inefficient but effective).
- We can use `git revert --no-commit <SHA>..HEAD` in the Terminal to revert to a specific version.
  - **Your file will now revert back to the same state as it was when you did your 'First commit'.**
- Our entire commit history will still be present.

## Final Thoughts

- GitHub is very useful for collaboration or to show off your ongoing projects.
  - Could be very useful for ADSC1910
- To collaborate, you can work in the following scenarios:
  1. Follow the workflow we covered. Everyone is connected to the remote repository and pushes changes to the main documents.
  2. Use **forks**, where everyone has their own copy of the main repository and there is a review process before any changes can be pushed to the main repository.
- You should commit often but only push when it feels absolutely necessary.

## Exercise 1

- Create a GitHub account and begin to populate your *portfolio* with projects that you are working on/have completed.

## References & Resources


1. Douglas, A., Roos, D., Mancini, F., Couto, A., & Lusseau, D. (2023). *An introduction to R*. Retrieved from https://intro2r.com/

2. Michael Freeman, Joel Ross, *Programming Skills for Data Science: Start Writing Code to Wrangle, Analyze, and Visualize Data with R*, 2019, ISBN-13: 978-0-13-513310-1


- https://docs.github.com/en
- https://github.com/