

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

- 2 Introduction
- 3 Version Control
- 4 Git and GitHub
- 5 Getting Started
- 6 Creating a Project
- 7 Using Git
- 8 Exercises and References

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@youreemail.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 \Rightarrow Global Options \Rightarrow 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

The screenshot shows the GitHub profile page for 'shellingman'. The 'Repositories' tab is selected and circled in white. The 'New' button in the repository filter bar is also circled in white. The profile section on the left includes a circular profile picture of a man with a dog, the name 'Sean Hellingman', the username 'shellingman', and an 'Edit profile' button. Below this, there are links to his university (Thompson Rivers University), location (Kamloops, Canada), and two LinkedIn profiles. The repository list on the right shows three public repositories: 'Tracking-Data-Challenge', 'Hockey-Odds-Challenge', and 'IPL-Runs-Earned', each with a 'Star' button. The 'BC-EV-Descriptive-Report' repository is partially visible at the bottom.

shellingman

Overview **Repositories** Projects Packages Stars

Find a repository... Type Language Sort **New**

Tracking-Data-Challenge (Public)
Data science challenge for soccer tracking data.
Updated on Jul 31

Hockey-Odds-Challenge (Public)
The task is to derive Over/Under 4.5 and 6.5 goals odds in ice hockey matches from the given data
Updated on Jul 31

IPL-Runs-Earned (Public)
IPL cricket data challenge
HTML Updated on Jul 3

BC-EV-Descriptive-Report (Public)
Summary of the EV network in B.C. created from the given data
HTML Updated on Jun 8

Sean Hellingman
shellingman
Edit profile

Thompson Rivers University
Kamloops, Canada
<https://ca.linkedin.com/in/sean-hellingman>
<https://www.researchgate.net/profile/Sean-Hellingman-2>

New Repo Settings

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 (*)

Owner *

shellingman

Repository name *

new_repo

new_repo is available.

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

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:

☒ Add a README file

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

Add .gitignore

.gitignore template: None

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

Choose a license

License: None

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

This will set `main` as the default branch. Change the default name in your settings.

☐ You are creating a public repository in your personal account.

Create repository

Copy Link

The screenshot shows a GitHub repository named 'new_repo' (Public). The repository has 1 branch (main) and 0 tags. The 'Code' button is highlighted in green, and a white arrow points to it. A dropdown menu is open, showing the 'Clone' section. The 'HTTPS' option is selected, and the URL 'https://github.com/shellingman/new_repo.git' is displayed. A white circle highlights the copy icon next to the URL. The 'Local' tab is also visible, showing the 'Clone' button and the same URL. The 'Codespaces' tab is also visible. The repository description is 'No description, website, or topics provided.' The repository has 0 stars, 0 watching, and 0 forks. The 'Releases' section shows 'No releases published' and a link to 'Create a new release'. The 'Packages' section shows 'No packages published' and a link to 'Publish your first package'.

new_repo Public

Pin Watch 0 Fork 0 Star 0

main 1 branch 0 tags

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](#)

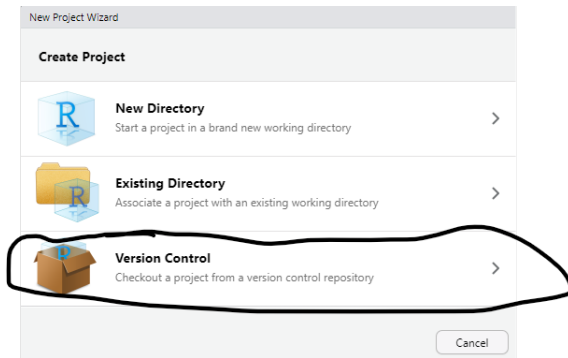
GitHub First II

- 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

New Project Wizard

[Back](#) **Clone Git Repository**



Repository URL:

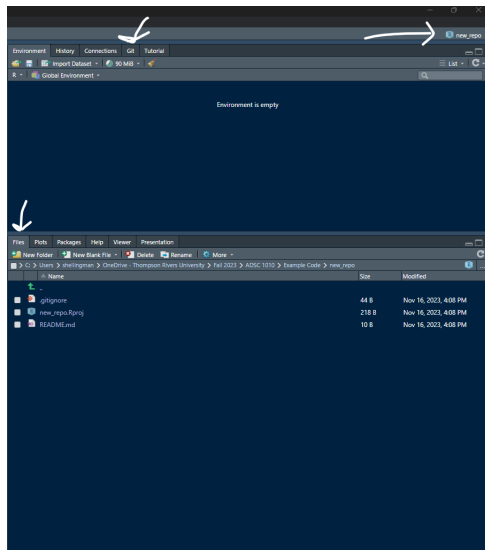
Project directory name:

Create project as subdirectory of:
 [Browse...](#)

☒ Open in new session

[Create Project](#) [Cancel](#)

Your Project

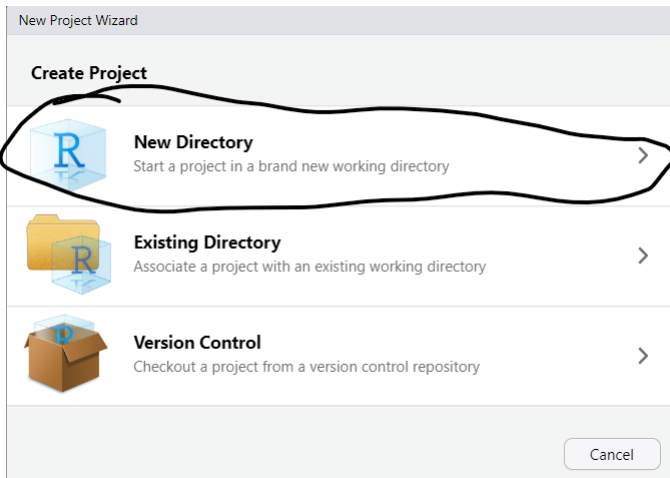


2. RStudio First

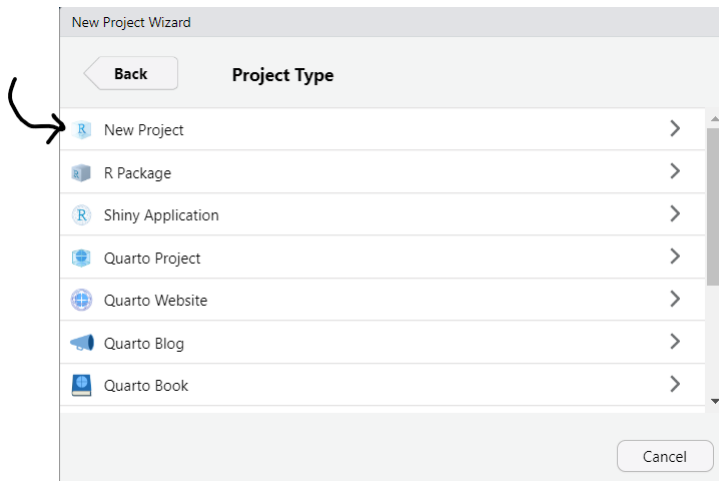
RStudio First I

- 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 \Rightarrow New Project \Rightarrow select New Directory \Rightarrow 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

New Project Wizard

Back **Create New Project**



Directory name:

Create project as subdirectory of:

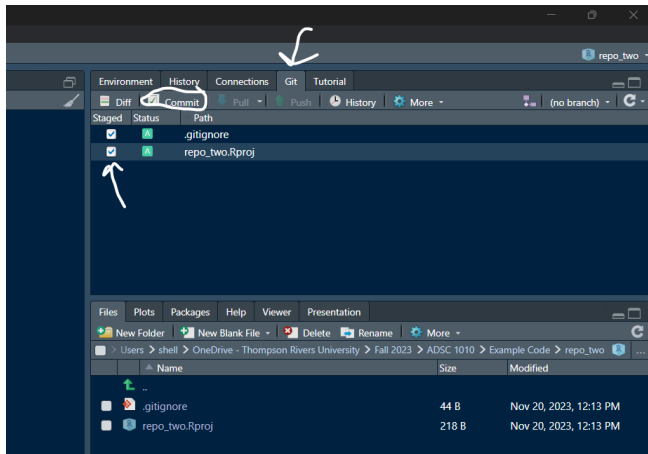
☒ Create a git repository
☐ Use renv with this project

☐ Open in new session

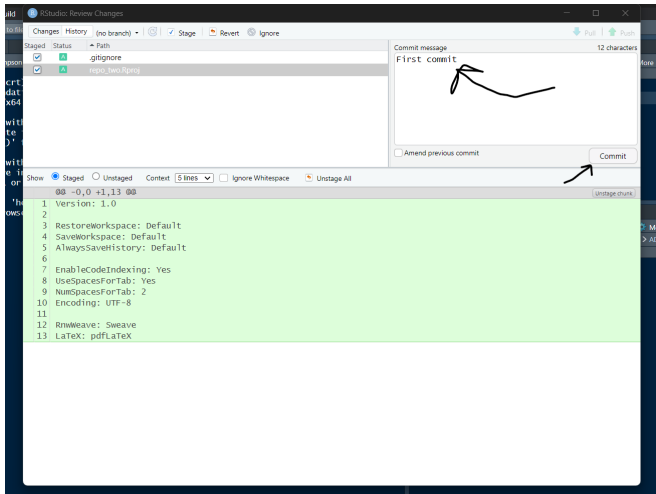
RStudio First II

- 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 \Rightarrow tick the boxes under the Staged column \Rightarrow click Commit. This leads to a Review Changes window.
- Type in the commit message `First commit` in the Commit message window \Rightarrow 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 \Rightarrow click Repositories \Rightarrow click New \Rightarrow and name your repo the same name as you gave it in RStudio.
- This time **do not** tick Initialize this repository with a README \Rightarrow 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


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 (*).


Owner *

Repository name *

 shellingman


 /

repo_two


 repo_two is available.

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

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:

☐ Add a README file

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

Add .gitignore


.gitignore template: None

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

Choose a license

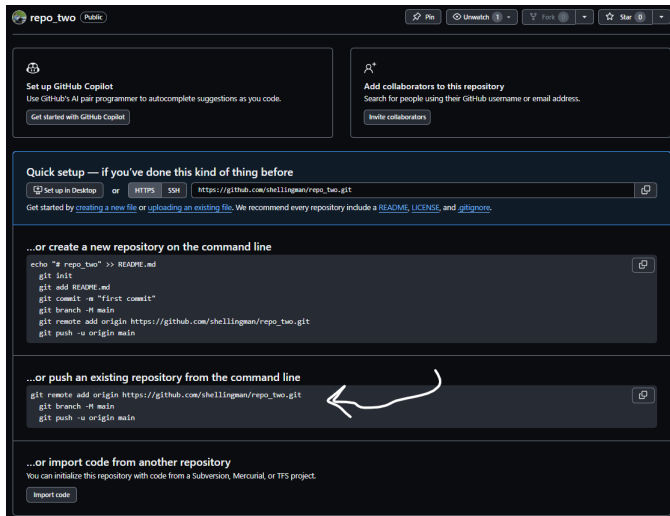
License: None

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

 You are creating a public repository in your personal account.

Create repository

Push Code



The screenshot shows the GitHub interface for a repository named 'repo_two'. At the top, there are buttons for 'Pin', 'Unwatch', 'Fork', and 'Star'. Below this, there are two main sections: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. The 'Quick setup' section provides instructions for setting up the repository on a desktop or via command line. The '...or create a new repository on the command line' section shows a series of git commands to initialize a new repository. The '...or push an existing repository from the command line' section shows git commands to push an existing repository, with a white arrow pointing to the 'git remote add origin' command. The '...or import code from another repository' section provides information on how to initialize the repository with code from other sources.

repo_two Public

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](#)

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

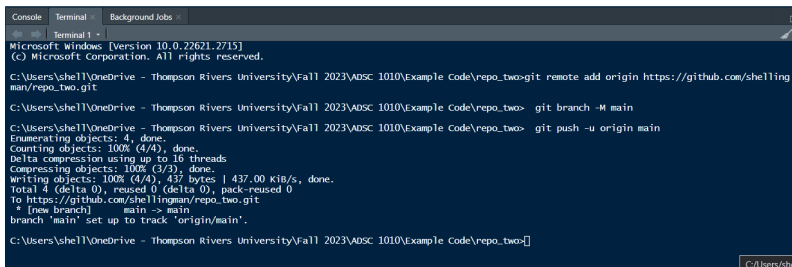
```
git remote add origin https://github.com/shellingman/repo_two.git
git branch -M main
git push -u origin main
```

...or import code from another repository

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

[Import code](#)

RStudio Terminal



```
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/shellingman/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>
```

Final Results

The screenshot shows a GitHub repository named 'repo_two' which is public. At the top, there are buttons for 'Pin', 'Unwatch' (with a count of 1), 'Fork' (with a count of 0), and 'Star' (with a count of 0). Below these, there are tabs for 'main' (selected), '1 branch', and '0 tags'. Action buttons include 'Go to file', 'Add file', and 'Code'. The repository has one commit by 'shellingman' with hash '8d6cb7', made 18 minutes ago. The commit list shows two files: '.gitignore' and 'repo_two.Rproj', both marked as 'First commit' and made 18 minutes ago. A blue box at the bottom of the commit list prompts the user to 'Add a README' to help people understand the project. On the right side, the 'About' section states 'No description, website, or topics provided.' and lists 'Activity', '0 stars', '1 watching', and '0 forks'. The 'Releases' section shows 'No releases published' and a link to 'Create a new release'. The 'Packages' section shows 'No packages published' and a link to 'Publish your first package'. Hand-drawn white annotations include a bracket on the left grouping the commit and file list, and an arrow pointing from the bottom right towards the 'Add a README' button.

repo_two Public

main 1 branch 0 tags

Go to file Add file <> Code

shellingman First commit 8d6cb7 18 minutes ago 1 commit

.gitignore First commit 18 minutes ago

repo_two.Rproj First commit 18 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

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.

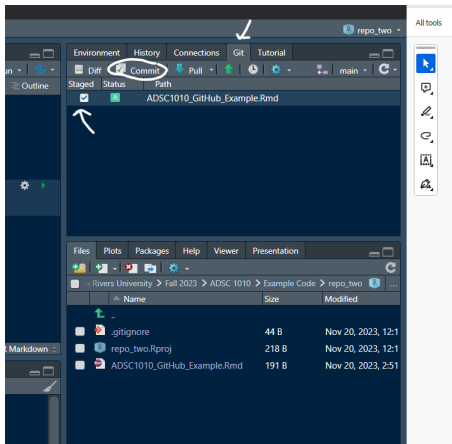
Git Workflow in Practice Example I

- In RStudio, open the project you want to work on under version control using Git.
- Create a new R markdown file within this project. \Rightarrow 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 \Rightarrow **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)

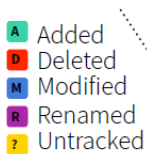
Git Workflow in Practice Example II

- To commit your 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 your files have been committed to GitHub.

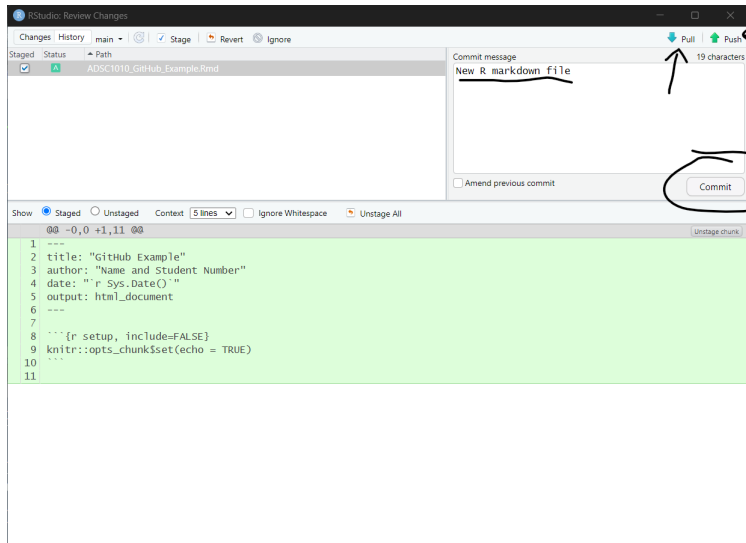
New Commit



Status Icons



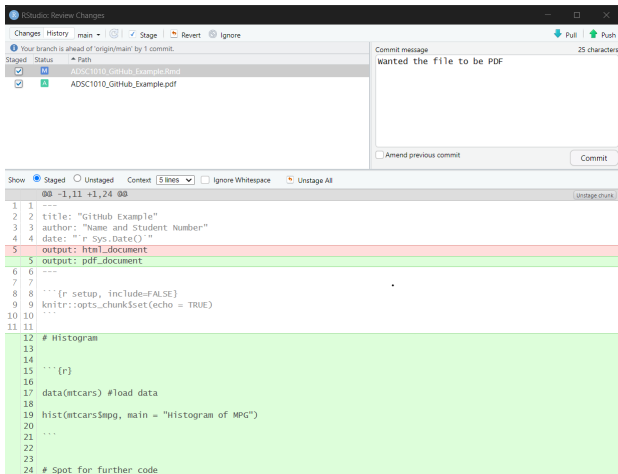
Push



Tracking Changes Example

- Simply make changes to your R markdown document and save them as you go.
- Knit your R markdown file to a 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 \Rightarrow add an appropriate commit message \Rightarrow 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

The screenshot shows the RStudio 'Review Changes' window. The top pane displays the commit history with three entries:

Subject	Author	Date (UTC)	SHA
HEAD → refs/heads/main compare Wanted pdf file	Sean <hell5140@mylaurier.ca>	2023-11-20	d22caa0f
New R markdown file	Sean <hell5140@mylaurier.ca>	2023-11-20	53035953
First commit	Sean <hell5140@mylaurier.ca>	2023-11-20	806bcb75

A hand-drawn arrow points to the 'Pull' button in the top right of the commit history pane.

The bottom pane shows the diff for the selected commit (SHA: d22caa0f5e5bb4c3e6dfb6bdd3c38757b4b6467). The changes are as follows:

```

SHA      d22caa0f5e5bb4c3e6dfb6bdd3c38757b4b6467
Author    Sean <hell5140@mylaurier.ca>
Date (UTC) 2023-11-20 23:34
Subject    Wanted pdf file
Parent     53035953a99c57ed483eeeca3b9e9e624033518bf

ADSC1010_GitHub_Example.Rmd
ADSC1010_GitHub_Example.pdf

ADSC1010_GitHub_Example.Rmd | View file @ d22caa0f
00 -2,10 +2,23 00
2 2 title: "GitHub Example"
3 3 author: "Name and Student Number"
4 4 date: "r Sys.Date()"
5 5 output: html_document
6 6 output: pdf_document
7 7 ---
8 8 '''{r setup, include=FALSE}
9 9 knitr::opts_chunk$set(echo = TRUE)
10 10 '''
11 11
12 12 # Histogram
13 13
14 14
15 15 '''{r}
16 16
  
```

The right pane shows the 'Environment' tab with a file explorer view of the repository, listing files like .gitignore, repo_two.Rproj, ADSC1010_GitHub_Example.Rmd, and ADSC1010_GitHub_Example.pdf.

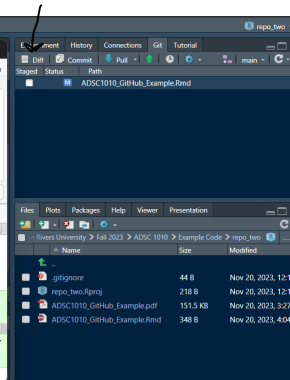
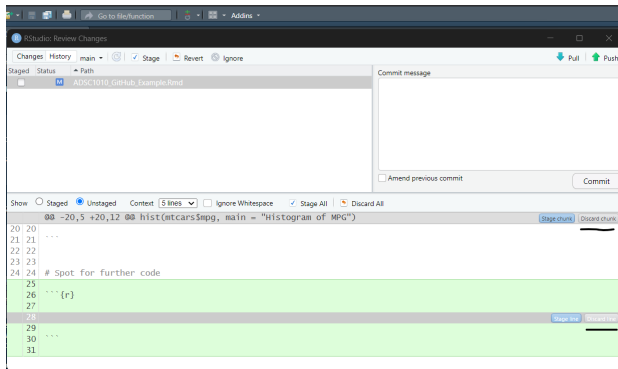
Reverting Changes

- You can use the functionality of Git to revert to previous versions of your files.
- How you do this depends 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)
⇒ 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 checkbox 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/>