

HW 1: Github and Errors

due 6/13/2023

Introduction

In this homework, you will have a chance to get practice with using Git in Posit Cloud while also practicing trouble shooting with error messages that can arise during coding.

Main Ideas

- Encountering errors and misbehaving code is common at all levels of ability.
 - By the way, just because you don't get an error message doesn't mean that your code is doing everything you want it to do - make sure that the output matches your intent!
 - In addition, fixing one error may cause other errors to become apparent. This is normal! Troubleshooting is a process.
- Employing a few simple strategies and steps to follow will help you diagnose and fix code problems quickly and independently.
- Troubleshooting code takes practice.

Please note that this assignment pre-supposes that you are somewhat familiar with the Posit Cloud RStudio IDE and that you have read the “Troubleshooting Errors in R” Handout on Canvas. If you are stuck, remember to follow the trouble-shooting steps laid out in that document!

Data

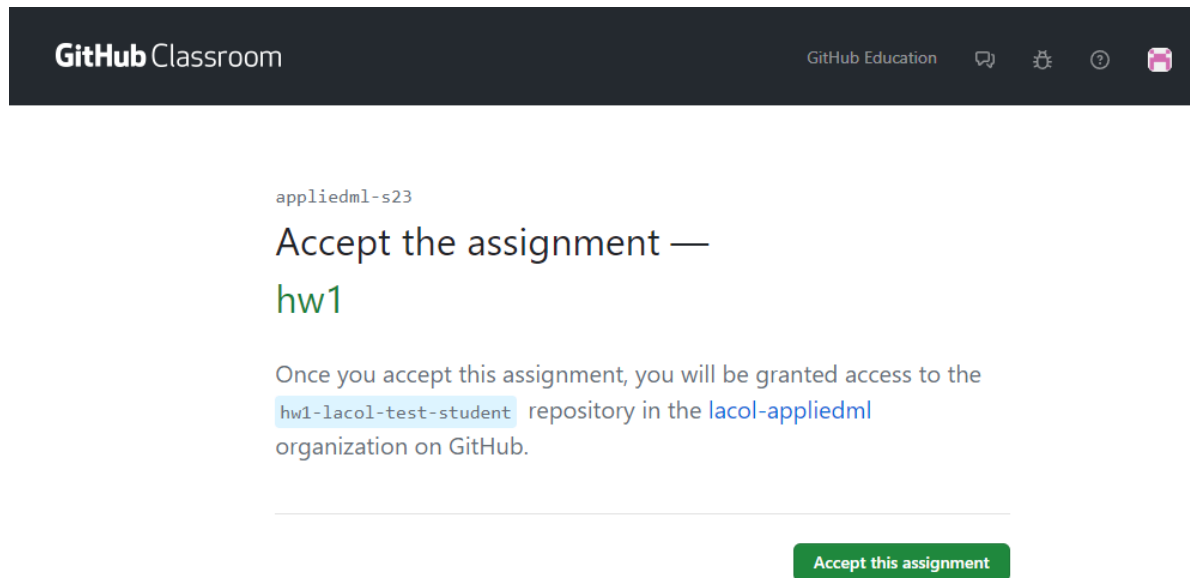
Purely for ease of use reasons, we will work with the `mpg` data set, which is found in the `ggplot2` package. You can access more information about this data set, including relevant variables, by running `?mpg` in the console.

Getting Started

Please note that this assignment pre-supposes that you have completed the steps in the GitHub Handout!

Each of your assignments will begin with the following steps. You saw these once already during the GitHub Handout activity, but they're outlined in detail here again. Going forward each assignment will start with a "Getting started" section but details will be a bit more sparse than this. You can always refer back to this assignment or the GitHub handout for a detailed list of the steps involved for getting started with an assignment.


- Click on the assignment link on Canvas. You will then be asked to accept the assignment.




- Once you accept the assignment, GitHub will create the repo for you. You will have to refresh the page for the link to the repo to show up.


GitHub Classroom

GitHub Education



You accepted the assignment, **hw1** . We're configuring your repository now. This may take a few minutes to complete. Refresh this page to see updates.

 Your assignment is due by **Jun 13, 2023, 12:00 EDT**



Join the GitHub Student Developer Pack


Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

- Click on the link - this will take you to your repo. This repo contains a template that you will build on to complete your assignment.

GitHub Classroom


GitHub Education




You're ready to go!


You accepted the assignment, **hw1**.

Your assignment repository has been created:

 <https://github.com/lacol-appliedml/hw1-lacol-test-student>

We've configured the repository associated with this assignment ([update](#)).

 Your assignment is due by **Jun 13, 2023, 12:00 EDT**

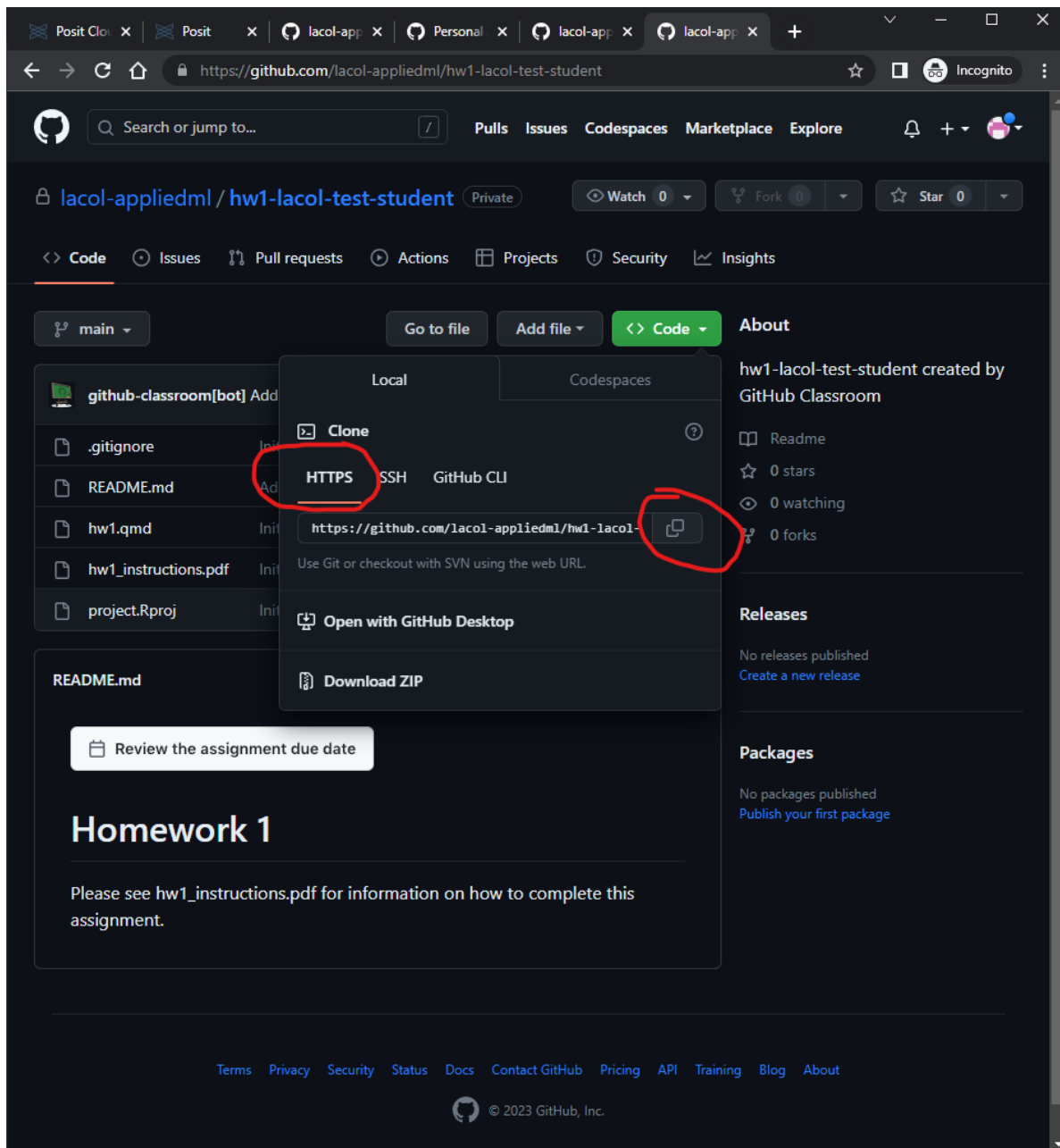


Join the GitHub Student Developer Pack

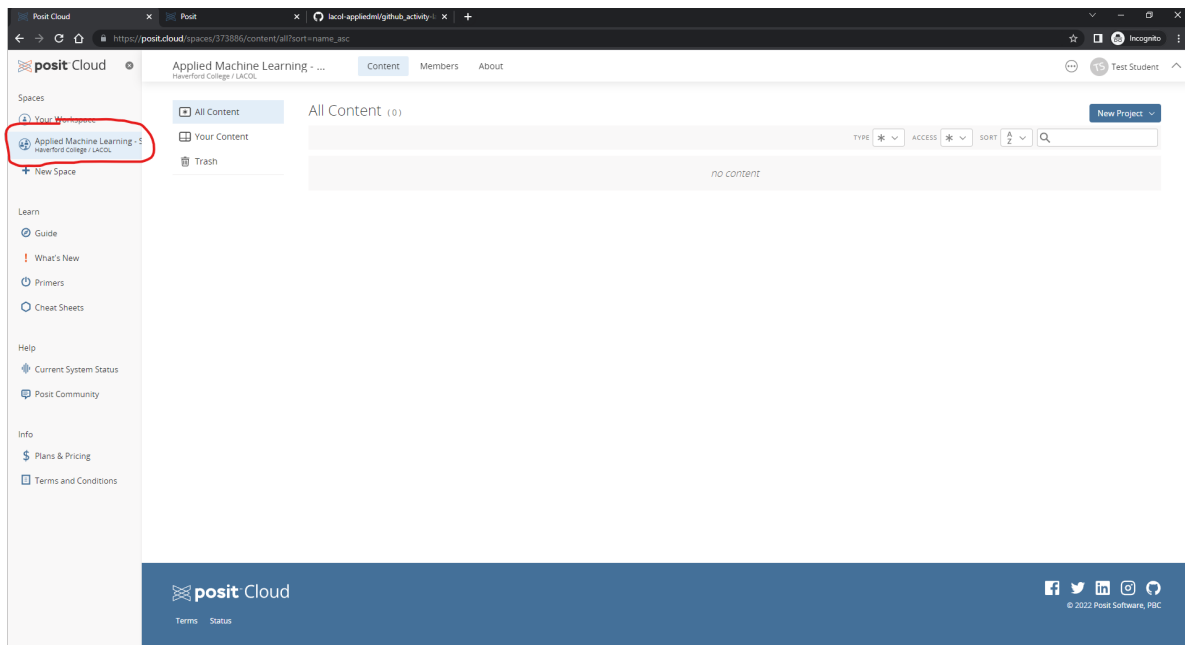
Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. [Learn more](#)

[Apply](#)

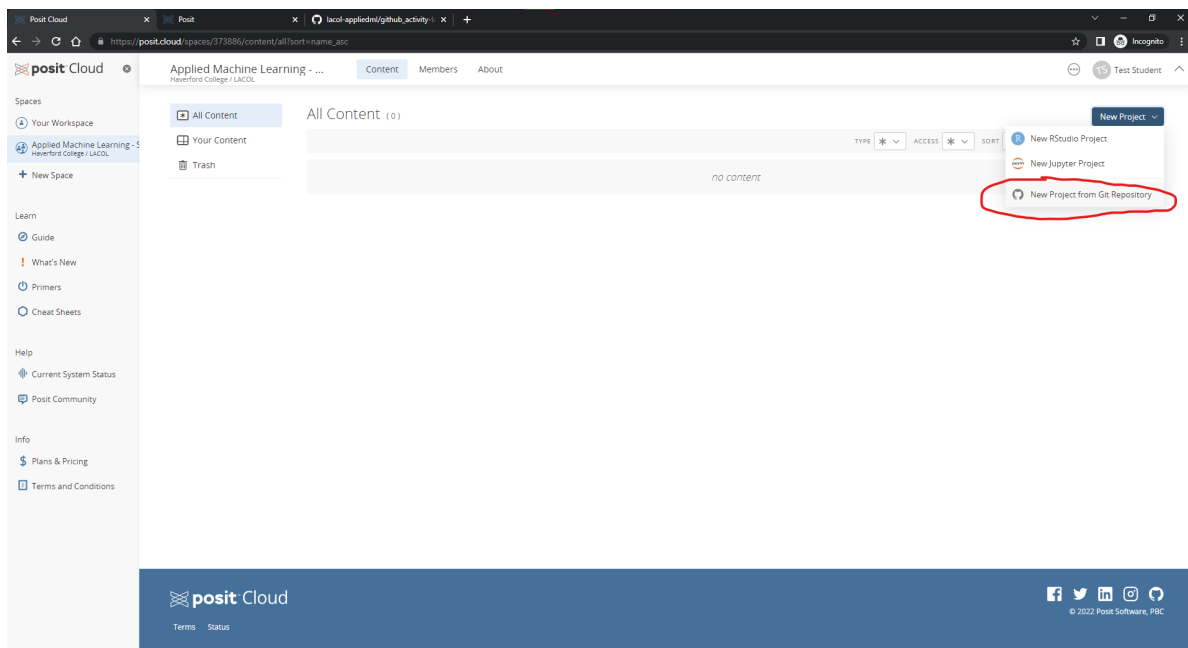
- On GitHub, click on the green **<> Code** button, select **HTTPS** (this might already be selected by default). Click on the clipboard icon (overlapping squares) to copy the repo URL.



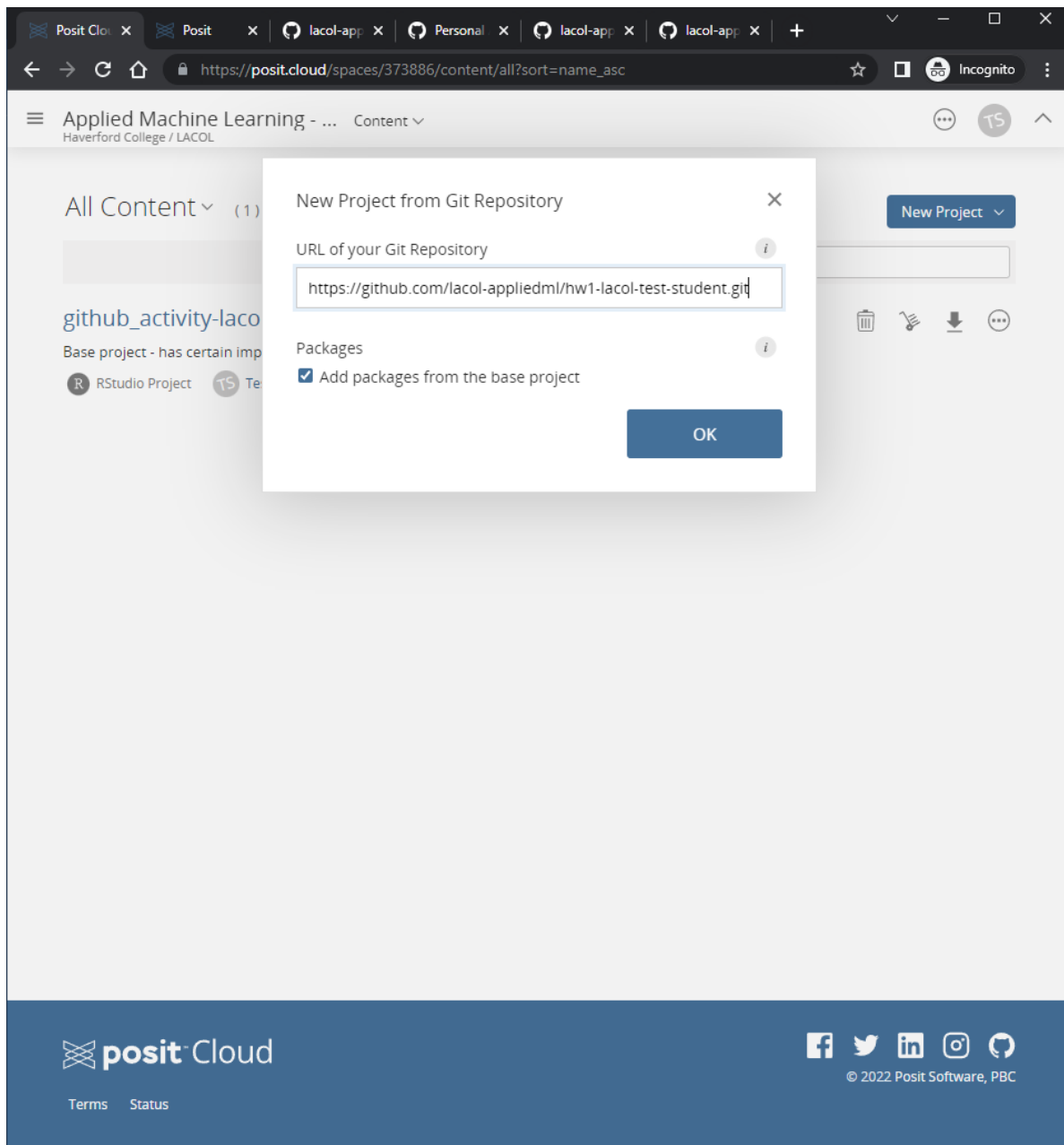
- Now go back to Posit Cloud, making sure that you are in the Applied Machine Learning - Summer 2023 workspace.



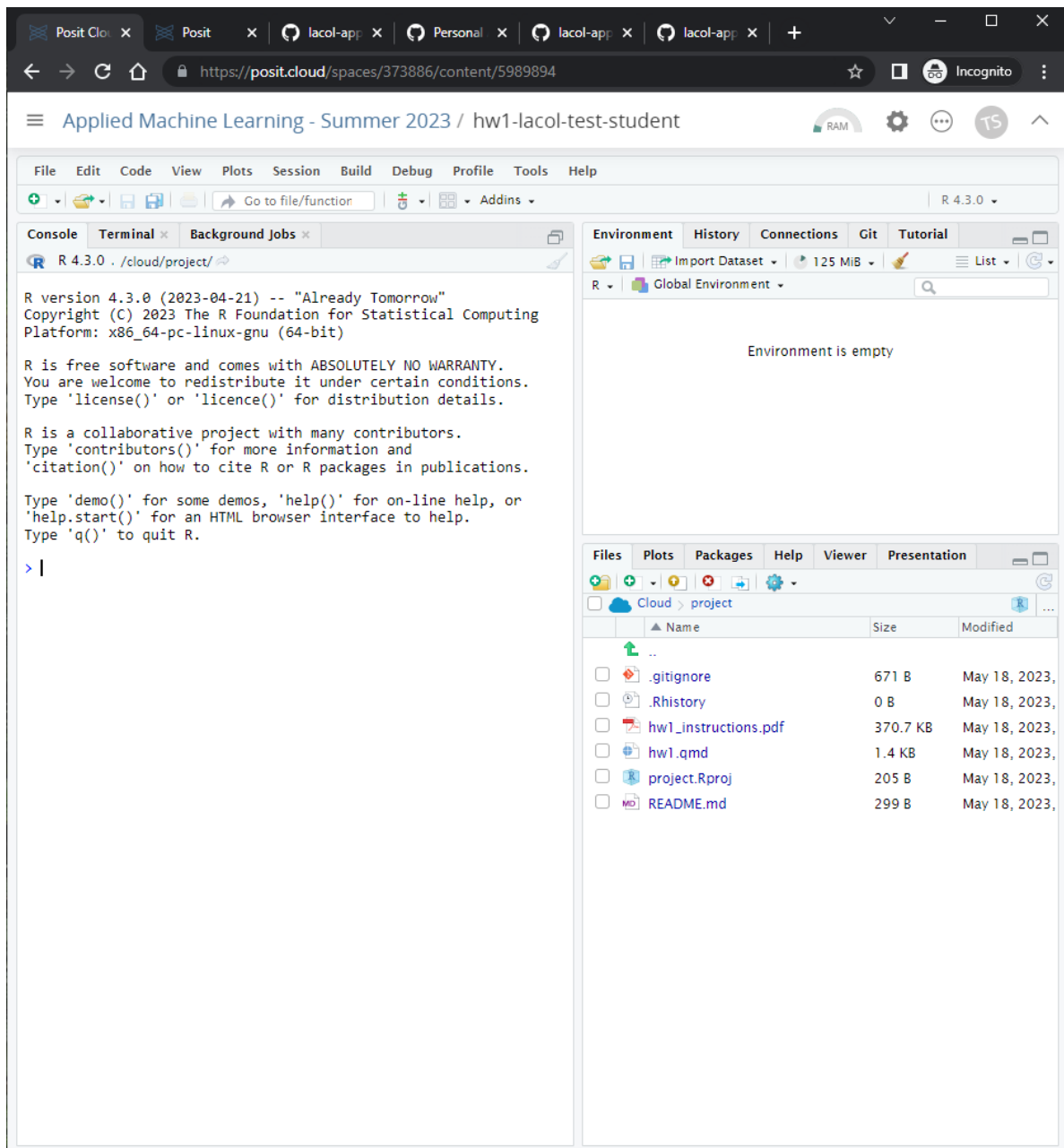
- Then, click on “New Project” near the top right-hand corner of the page. This will open a drop-down selection list. Here, you will want to click on “New Project from Git Repository”.



- Paste the URL you copied from GitHub into the dialogue box and press “OK”



Voila! After up to 30 seconds, a new project called **hw1-YOUR-USER-NAME** will open in Posit Cloud. This will look like the RStudio IDE with which you should already be familiar.



Warm Up

Before we get started with the assignment, let's warm up by adding our name and date to our document.

YAML

YAML

As a reminder, the top portion of your Quarto document (between the two sets of three dashed lines) is called the YAML. It stands for “YAML Ain’t Markup Language” ¹.

YAML is a human-friendly data serialization standard (a way to encode metadata) for all programming languages. All you need to know is that this area is called the YAML (we will refer to it as such) and that it contains meta information about your document.

Your YAML currently looks like the following:

```
---
title: "HW1: GitHub and Errors"
author: "Insert your name here"
date: "Insert date here"
format: pdf
---
```

- In the **author** field, put your name. In the **date** field, put today’s date. Then, render the document.

Committing Changes

After rendering the document, go to the Git pane in your RStudio. If you have made changes to your `.qmd` and created a new `.pdf` file, these should be listed here. Click on it to *stage* the files. Click “Diff” if you want to see the *difference* between the last committed state of the document and its current state that includes your changes. If you are happy with these changes, write “Updatge author name and data” in the **Commit message** box and hit “Commit”.

Push Changes

Now that you have made an update and committed this change, it’s time to push these changes to the web! Or more specifically, to your repo on GitHub. Why? So that others can see your changes. And by others, we mean the course teaching team (your repos in this course are private to you and us, only).

In order to push your changes to GitHub, click on **Push**. This will prompt a dialogue box where you first need to enter your user name, and then your personal access token.

¹Yes, it is recursive, what a funny computer science meta-joke!

Exercises

! Note

Each of these chunks below has multiple errors in it; fixing one will likely reveal another. So go slowly and carefully!

Also, feel free to remove the `#| error: true` chunk option if you do want RStudio to stop rendering if it encounters an error.

Finally, please be aware that the errors displayed in the rendered `hw1_instructions.pdf` may not match the ones you will see; I have included them so that you can see what it looks like when a document is rendered with the `#| error: true` chunk option set.

Exercises

After fixing each set of errors, write a short paragraph explaining what you fixed and how you figured out what you had to fix.

Q1

The code chunk below tries to calculate some statistics for the city miles per gallon for different types of drive trains (column `drv`), but it has a few small errors. Try to find and fix these error by running the code line by line (or complete though by complete thought).

```
mpg %>%  
  filter(class == "subcompact") %>%  
  group_by(drv)
```

Error in `mpg %>% filter(class == "subcompact") %>% group_by(drv)`: could not find function "%>%"

```
summarize(median_cty_mpg = median(cty),  
          sd_cty_mpg = sd(cty),  
          avg_cty_mpg = average(cty))
```

Error in `summarize(median_cty_mpg = median(cty), sd_cty_mpg = sd(cty), : could not find function "average"`

! Don't forget to commit!

If you haven't been staging and committing the files, now (after question 1) would be a good time to do so!

A fitting commit message would be something like "Finish question 1".

Q2

The following code chunk tries to make a scatter plot of city miles per gallon versus highway miles per gallon. Fix the errors.

```
mpg %>%  
  ggplot(aes(x = highway, y = cty)) %>%  
  geom_point() +  
  labs(x = "Highway", ylabel = "City",  
        title = "Miles per Gallon - Highway vs City")
```

Error in mpg %>% ggplot(aes(x = highway, y = cty)) %>% geom_point(): could not find function

! Don't forget to commit!

If you haven't been staging and committing the files, now (after question 2) would be a good time to do so!

A fitting commit message would be something like "Finish question 2".

Q3

Here we are creating two vectors in R, and then we try to multiply them element-wise. We get a series of errors here. Can you tell what they are and fix them?

```
vec1 <- c(1, 2, 3)  
vec2 <- (3, -2, 2.305)  
  
vec2 %* vec
```

```
Error: <text>:2:11: unexpected ' ','
1: vec1 <- c(1, 2, 3)
2: vec2 <- (3,
             ^
```

! Don't forget to commit!

If you haven't been staging and committing the files, now (after question 3) would be a good time to do so!

A fitting commit message would be something like "Finish question 3".

Q4

The following code gets the average miles per gallon on the highway and in the city for different fuel types. The overall goal is to then get the average of for both highway and city across fuel types (i.e. we want the output to be two numbers).

```
mean(mpg_fl$mean_hwy, mpg_fl$mean_cty)
```

```
Error in eval(expr, envir, enclos): object 'mpg_fl' not found
```

```
mpg_fl <- mpg %>%
  groupby(Fl) %>%
  summary(
    mean_hwy, mean_cty = mean(hwy, cty)
  )
```

```
Error in mpg %>% groupby(Fl) %>% summary(mean_hwy, mean_cty = mean(hwy, : could not find fun
```

! Don't forget to commit!

If you haven't been staging and committing the files, now (after question 4) would be a good time to do so!

A fitting commit message would be something like "Finish question 4".

Q5

In the following code, we create two vectors, cbind them together into one data object. Then we try to increase the values in the first column of `d` by one. However, we are running into an error.

```
a <- c(1,2,3)
b <- c("a","b","c")
d <- cbind(a,b) # Put them together
d[,1] + 1
```

Error in `d[, 1] + 1`: non-numeric argument to binary operator

Final Steps

When you have made your final edits, remember to render, stage the changed files, then commit (including an appropriate commit message), and push to the repo.

Grading

Please put your names on your answer document and add the date. You won't get points for doing this, but you will *lose* points for not doing so.

For clarity's sake, you should label your code chunks (this helps make debugging easier). You will *lose* points for not naming chunks.

Commenting your code is crucial for replicability and transparency purposes. Remember to comment your code. You don't get points for commenting, but you will *lose* points for not commenting your code.

If an exercise asks a question, you should answer it in narrative form and not just rely on the code. You will be *penalized* otherwise.

Please suppress warnings and messages. You will *lose* points if you do not suppress warnings and messages.

The different components of this homework are worth the following points:

- Q1: 4 pts
- Q2: 5 pts
- Q3: 5 pts
- Q4: 7 pts

- Q5: 2 pts
- Workflow (includes making sufficient commits and labeling chunks): 4 pts

Total: 27 pts