PSC 253 Minimal Manual

Matthew B. Platt

2022-09-01

Contents

Preface

This book is a supplement to the book, Quantitative Social Science: An Introduction, by Kosuke Imai. It also relies heavily on the work of Jeffrey Arnold, who translated the Imai code into tidyverse code.

This manual is also a supplement to the book An R Companion to Political Analysis by Philip Pollock III and Barry Edwards.

I aspire for this text to act as a minimal manual for the course PSC 253 Scope and Methods in Political Science taught at Morehouse College. It is intended to cover all of the main analytical tasks that the course requires.

Chapter 1

R Basics

At its most basic functionality, R is a calculator.

1.1 Use R as a Calculator

1.1.1 Problem

You want to add, subtract, multiply, divide, use exponents, and take square roots

1.1.2 Solution

Use `+` for addition, `-` for subtraction, `*` for multiplication and `/` for division.

```
# addition
43 + 5
```

```
## [1] 48
```

```
# subtraction
43 - 5
```

```
## [1] 38
```

```
# multiplication
43 * 5
```

```
## [1] 215
```

```
# division
43/5
```

```
## [1] 8.6
```

For exponents, we raise `X` to the power of `y` by using `^`. That is `X^y`.

```
# raise 43 to the power of 5
43 ^ 5
```

```
## [1] 147008443
```

Take the square root of some number `x` by using the function `sqrt()`. That is `sqrt(x)`.

```
# take the square root of 43
sqrt(43)
```

```
## [1] 6.557439
```

1.1.3 Troubleshooting

- Keep in mind that R follows the order of operations, 2 + 2 * 2 is equal to 6 and not 8.

```
# correct
2 + 2 * 2
```

```
## [1] 6
```

```
# incorrect
(2 + 2) * 2
```

```
## [1] 8
```

1.2 Creating an Object

1.2.1 Problem

You want to create an object to hold a number

1.2.2 Solution

To create an object:

1. type in a name for the object, like `newobject` then
2. use the assignment operator `<-`,
3. input a number, mathematical expression, dataset, or text on the right side of `<-` that you want assigned to the `newobject`

```
# assigning the number 4 to a new object named "myobject"
myobject <- 4

# assigning the text "hallelujah hollaback" to a new object named "second_object"
second_object <- "hallelujah hollaback"
```

Type the name of an object in order to see what it contains.

```
myobject
```

```
## [1] 4
```

```
second_object
```

```
## [1] "hallelujah hollaback"
```

1.2.3 Troubleshooting

- There cannot be any spaces in the name of an object. Instead you could use dots, dashes, underscores, or capitalization to distinguish between words: `small.data`, `big-data`, `bigger_data`, `mediumData`.
- Text needs to be in quotation marks in order to be assigned to an object.
- Object names are case sensitive `Myobject` is not the same as `myobject`

1.3 Creating a Vector

A vector is a list of numbers or characters. We will create vectors for a variety of reasons in this course.

1.3.1 Problem

You want to create a vector.

1.3.2 Solution

Use the function `c()` to create a list by separating the entries with a comma.

```
# create a vector called 'prime'
prime <- c(1, 3, 5, 7)

prime
```

```
## [1] 1 3 5 7
```

```
# create a vector called "first_name"
first_name <- c("Matthew", "Mosi", "Manu", "Ekundayo", "Kwasi")

first_name
```

```
## [1] "Matthew"  "Mosi"     "Manu"     "Ekundayo" "Kwasi"
```

1.3.3 Troubleshooting

- As the name of a function, `c()` is case sensitive. Use the lowercase `c`.
- Make sure that all elements are separated by a comma.
- Vectors are typically assigned to some object.

1.4 Indexing

We can use indexing to pull out specific sets of observations from a vector or dataset.

1.4.1 Problem

You want to select a specific one observation based on its position within a vector or matrix.

1.4.2 Solution

We index by using the brackets `[x, y]` after an object where x is the row and y is the column.

```
# we have a vector
prime <- c(1, 3, 5, 7)

# we want the number 3, which is the second observation in the vector
prime[2]
```

```
## [1] 3
```

```
# we have a matrix
yup <- matrix(c(prime,2, 6, 10, 14), nrow = 2, ncol = 4, byrow = T)
yup
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    6   10   14
```

```
# We want the observation in the first row and fourth column
yup[1, 4]
```

```
## [1] 7
```

1.4.3 Problem

You want to select an entire row or column.

1.4.4 Solution

You can select an entire row by leaving the column index position blank `yup[2, ]`. You can select an entire column by leaving the row index position blank `yup[ , 2]`.

```
# We have our matrix
yup
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    6   10   14
```

```
# We want the second row
yup[1, ]
```

```
## [1] 1 3 5 7
```

```
# We want the third column
yup[ , 3]
```

```
## [1]  5 10
```

1.5 Creating a Project

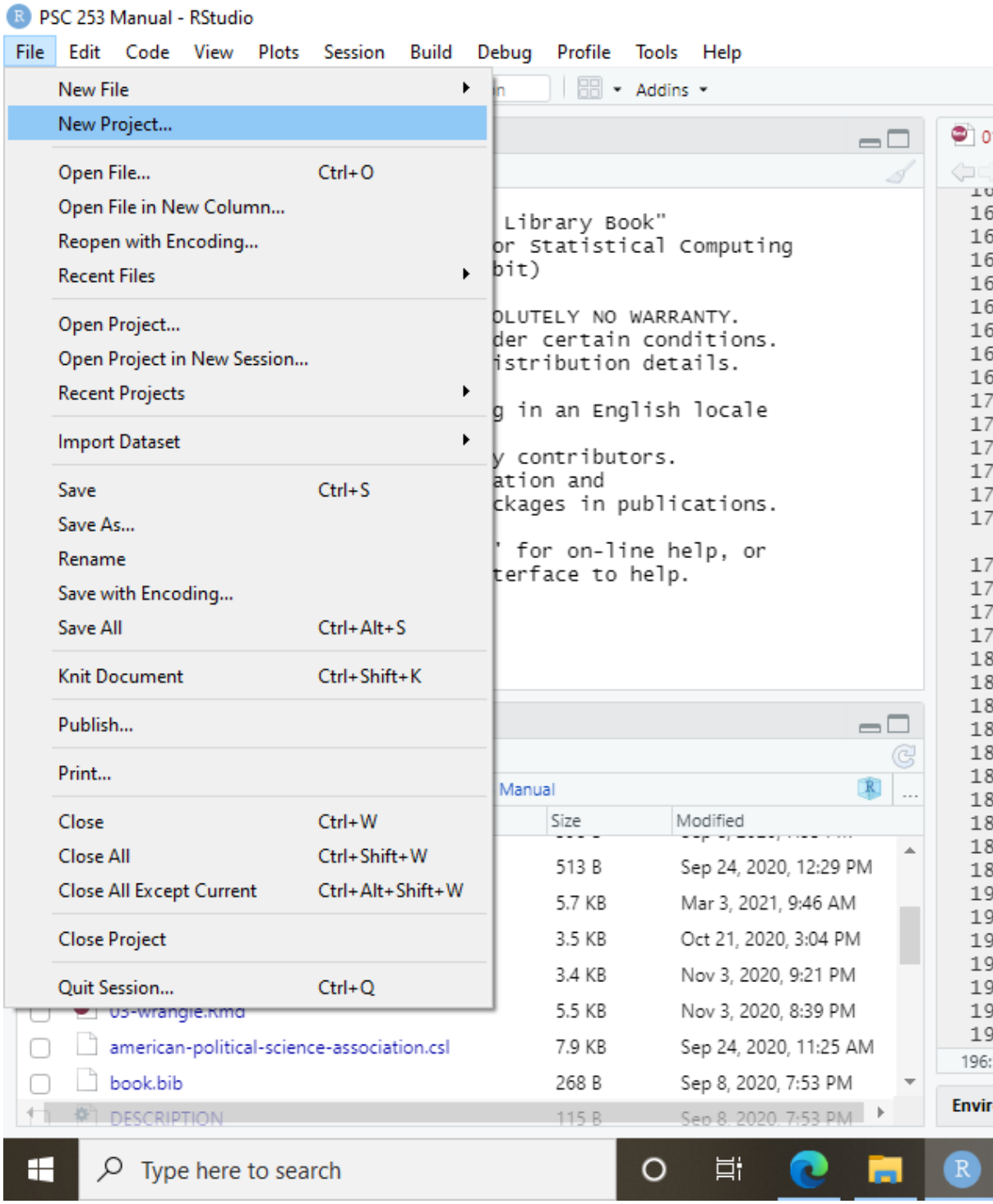
Creating a project creates a folder on your computer that Rstudio and R will treat as the default directory for your code. That is, whenever you tell R to look for something on your computer, it will begin by looking in the project folder.

1.5.1 Problem

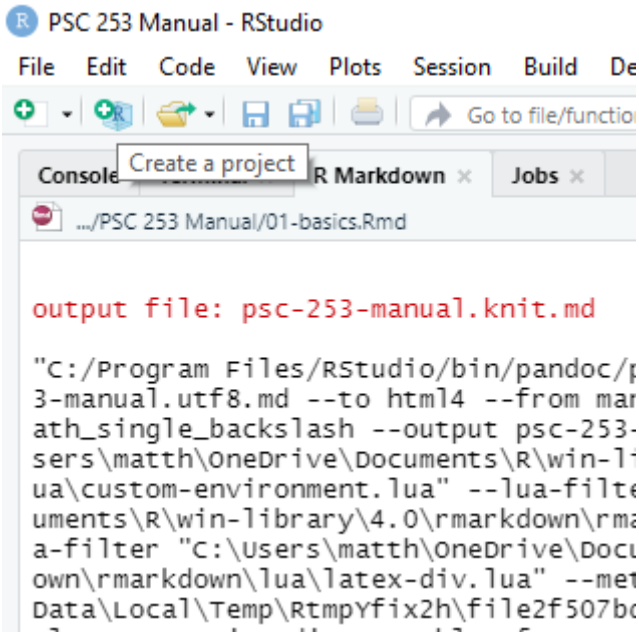
You want to create a new project.

1.5.2 Solution

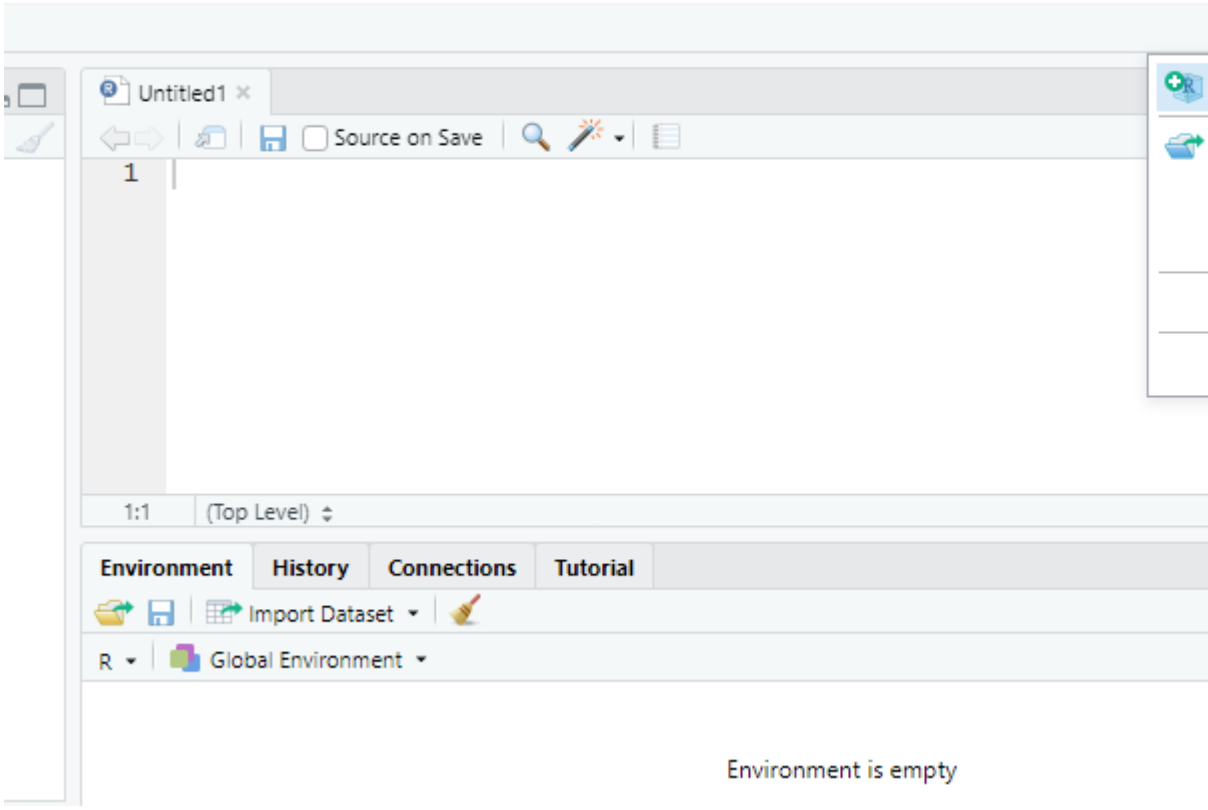
There are multiple ways to create a new project. You can go to the menu bar and select “File”, then “New Project”:


PSC 253 Manual - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
New File
New Project...
Open File... Ctrl+O
Open File in New Column...
Reopen with Encoding...
Recent Files
Open Project...
Open Project in New Session...
Recent Projects
Import Dataset
Save Ctrl+S
Save As...
Rename
Save with Encoding...
Save All Ctrl+Alt+S
Knit Document Ctrl+Shift+K
Publish...
Print...
Close Ctrl+W
Close All Ctrl+Shift+W
Close All Except Current Ctrl+Alt+Shift+W
Close Project
Quit Session... Ctrl+Q
Addins
Manual
Size
Modified
513 B Sep 24, 2020, 12:29 PM
5.7 KB Mar 3, 2021, 9:46 AM
3.5 KB Oct 21, 2020, 3:04 PM
3.4 KB Nov 3, 2020, 9:21 PM
5.5 KB Nov 3, 2020, 8:39 PM
american-political-science-association.csl 7.9 KB Sep 24, 2020, 11:25 AM
book.bib 268 B Sep 8, 2020, 7:53 PM
Type here to search

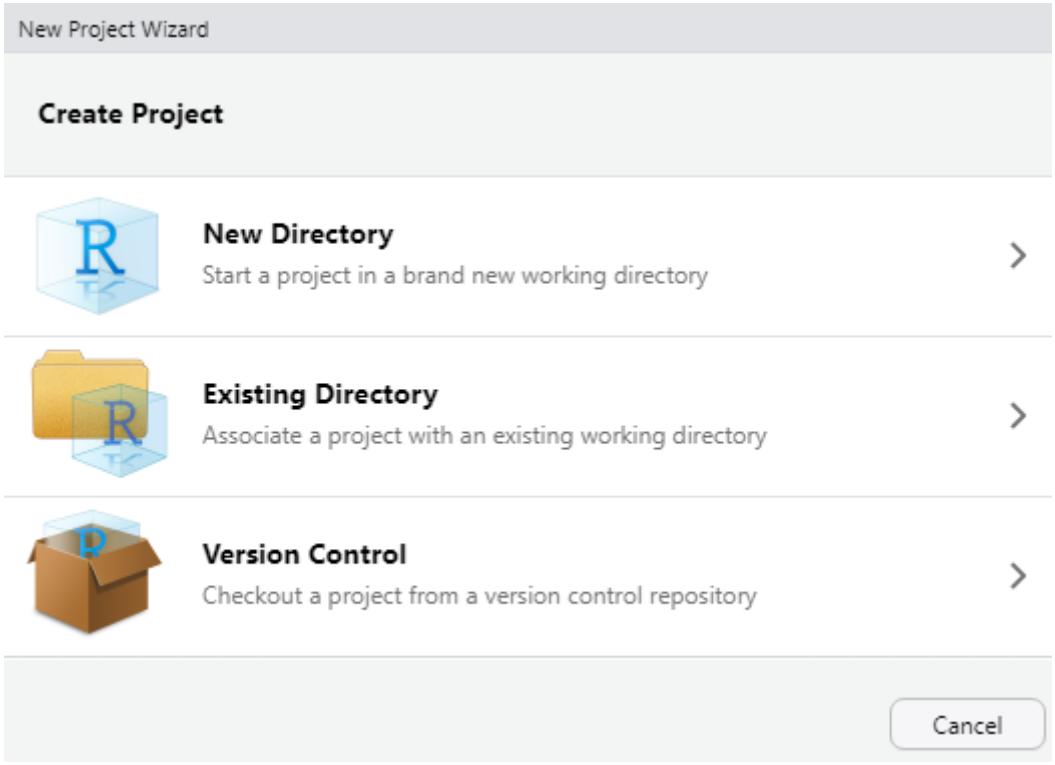
You can click the "create project" icon on the toolbar:



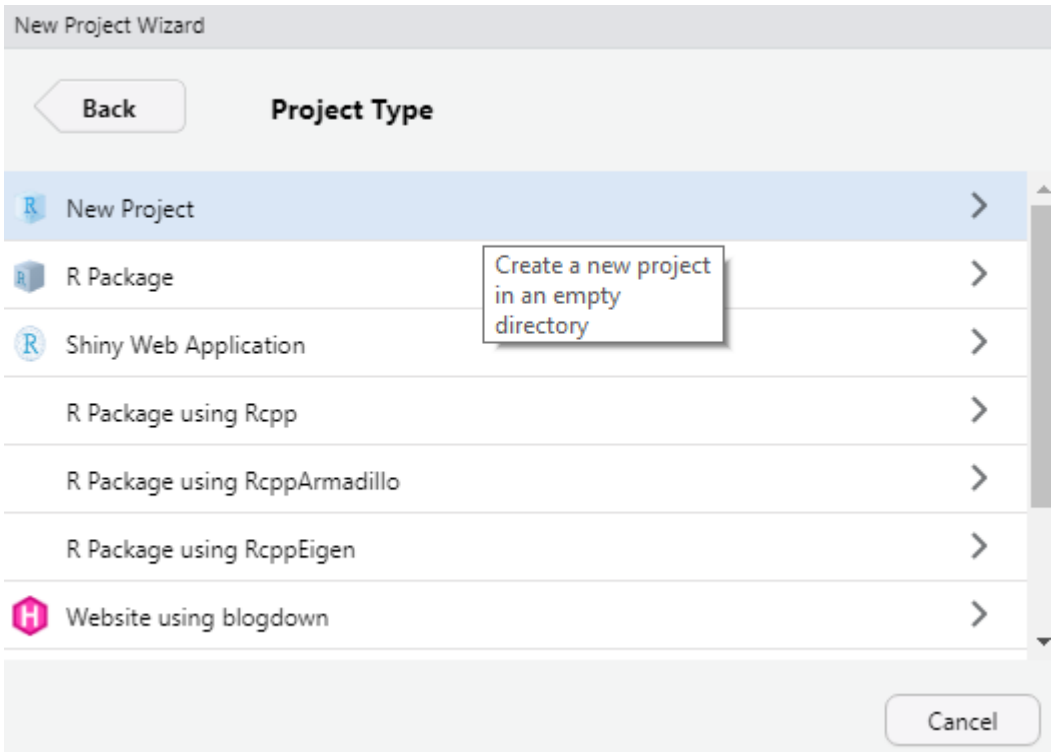
Or you can select "New Project" from the project menu at the top left of Rstudio:



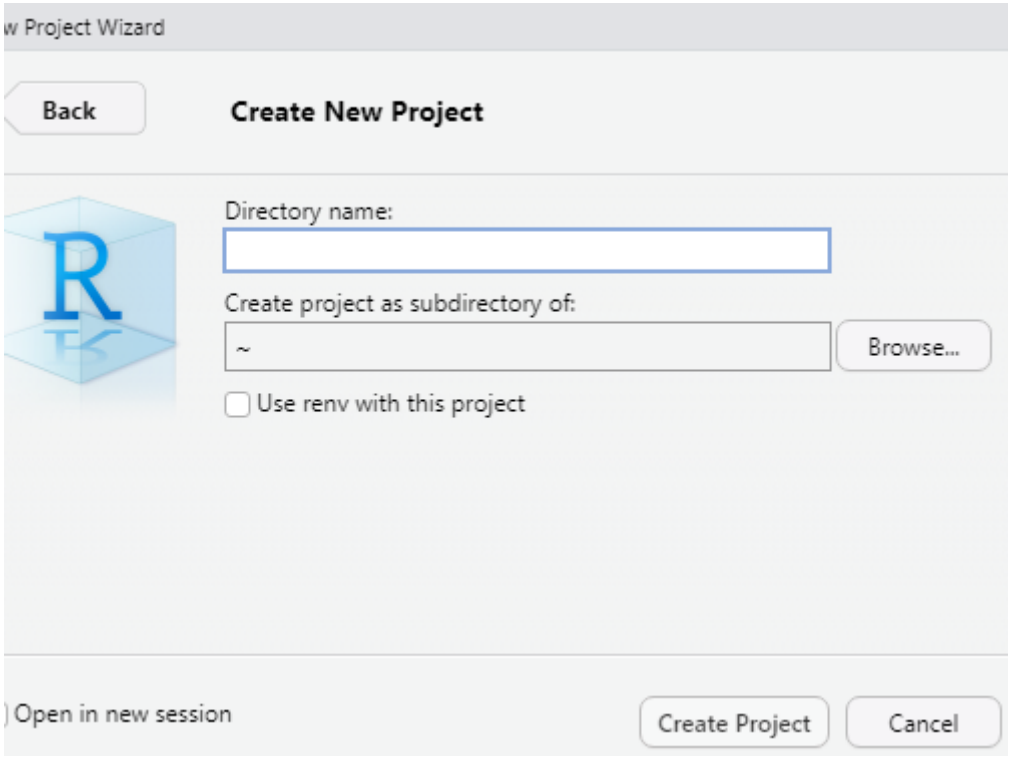
When you select "New Project", the below dialogue box appears. Select "New Directory":



Then select “New project”,



enter a name for the project, and browse to where you want the directory located on your computer.



Complete the process by clicking "Create Project."

1.5.3 Troubleshooting

- You need to know where your project folders are on your computer. I recommend that you create a master folder called "PSC 253", and then have all of your project folders inside of that master folder.
- We will use separate projects for each lab assignment and for the Data Project assignment.
- Avoid having multiple project folders for the same assignment. It will cause confusion when you are trying to submit the correct project folder for your assignment.

1.6 Installing a Package

A *package* is a specialized collection of R elements – datasets, functions, objects, etc. We will use various packages to help in our data analysis. Packages have to be installed and loaded before their elements can be accessed, so this section will teach you how to install a package.

1.6.1 Problem

You want to install a package.

1.6.2 Solution

1. You can type in the command `install.packages("packagename")`. Thus, to install the package named "here" we would type `install.packages("here")`.

```
# installing the poliscidata package

install.packages("poliscidata")
```

Alternatively, you can use the menu bar:

1. Click on "Tools" in the menu bar.
2. select "Install Packages"
3. A dialogue box will appear.
4. Type in the name of the package you want.
5. Click "Install".

1.6.3 Troubleshooting

- Some packages depend on the installation of other packages first. R will install these dependencies automatically, so it may take a while for your package to install. You know the installation is done when the console shows its arrow and blinking cursor.
- If you get a message asking you to install Rtools, then you can do so [here][https://cran.r-project.org/bin/windows/Rtools/].
- If you get a message that says "exited with non-zero status", then the package did not fully install. This could be due to one of the dependency packages not installing properly. Try to install that dependency package manually – `install.packages("dependencyname")`. Once the dependent package is installed, you can try to install the main package again.
- You will know that the package has been successfully installed if you are able to load the package.
- In rare occasions, a package may require a later edition of R than you have installed on your computer. The message will say something like "This package requires R 4.1.0 and you have R 3.1.0". In that case, you will need to download and install the latest version of R before you can install the package.
- Remember to use quotation marks around the package name in `install.packages()`.

1.7 Loading a Package

It is not enough to just install a package. Installed packages must be loaded in order for you to access their elements.

1.7.1 Problem

You want to load a package that you have installed.

1.7.2 Solution

The function we use to load packages is `library()`. Its main argument is the name of the package – `library(packagename)`.

```
# load the poliscidata package

library(poliscidata)
```

1.7.3 Troubleshooting

- Packages must be fully and properly installed before they can be loaded.
- If you get an error saying that a package does not exist then 1) you have not installed the package or 2) you have spelled the name of the package incorrectly.
- Remember that we do not put the name of the package in quotation marks when we are loading it.

1.8 Using relative file paths

Whenever we load data or save data we will need to specify a file path – where the file is located on the computer. Obviously, the file path on your computer will not be the same as the path on someone else's computer. We want our code to be able to run on any computer, so we create *relative* file paths. That is, we specify where the file is located relative to the project folder.

1.8.1 Problem

You want to create a file path relative to the project folder.

1.8.2 Solution

1. You need to already have a project folder. See Section 1.5.
2. Load the "here" package
3. The function `here()` automatically sets the relative point of the file path as the location of your `.Rproj` file.
4. You can then provide the file path as the argument to `here()`.

```
# Creating a path to the "images" folder in this project.

library(here)
```

```
## here() starts at C:/Users/matth/OneDrive/R Code/PSC 253 Manual
```

```
here("images")
```

```
## [1] "C:/Users/matth/OneDrive/R Code/PSC 253 Manual/images"
```

For example, we have a project folder named `hypothesis_test_lab`. Figure 1.1 shows the inside of that project folder. There are three subfolders: Data, Documents, and Scripts. Inside the "Data" folder there are two more folders called `Analysis_Data` and `Original_Data`.

We will use the `here()` function to load a data file that is found within the `Analysis_Data` folder.

```
# loading "analysis1.RData" file

load(here("Data/Analysis_Data/analysis1.RData"))
```

1.8.3 Troubleshooting

- Remember to use forward slashes `/` and quotation marks when writing the relative path for `here()`.
- The use of `here()` is a new function introduced in Fall 2021. Code from prior versions of the course that used relative paths may need to be revised if you want to use the `here()` function.
- Your first code chunk should load the `here` package – `library(here)`.

1.9 Loading .csv Data

There are some packages that come with datasets attached to them. However, we will mostly need to import datasets into R. This section deals with how to specify the file path for the data and the various functions that correspond to the different types of data files you may encounter.

1.9.1 Problem

You want to load a datafile that takes the form `data_name.csv`.

1.9.2 Solution

1. You already have a project folder. (Section 1.5)
2. You already know how to create relative paths. (Section 1.8)

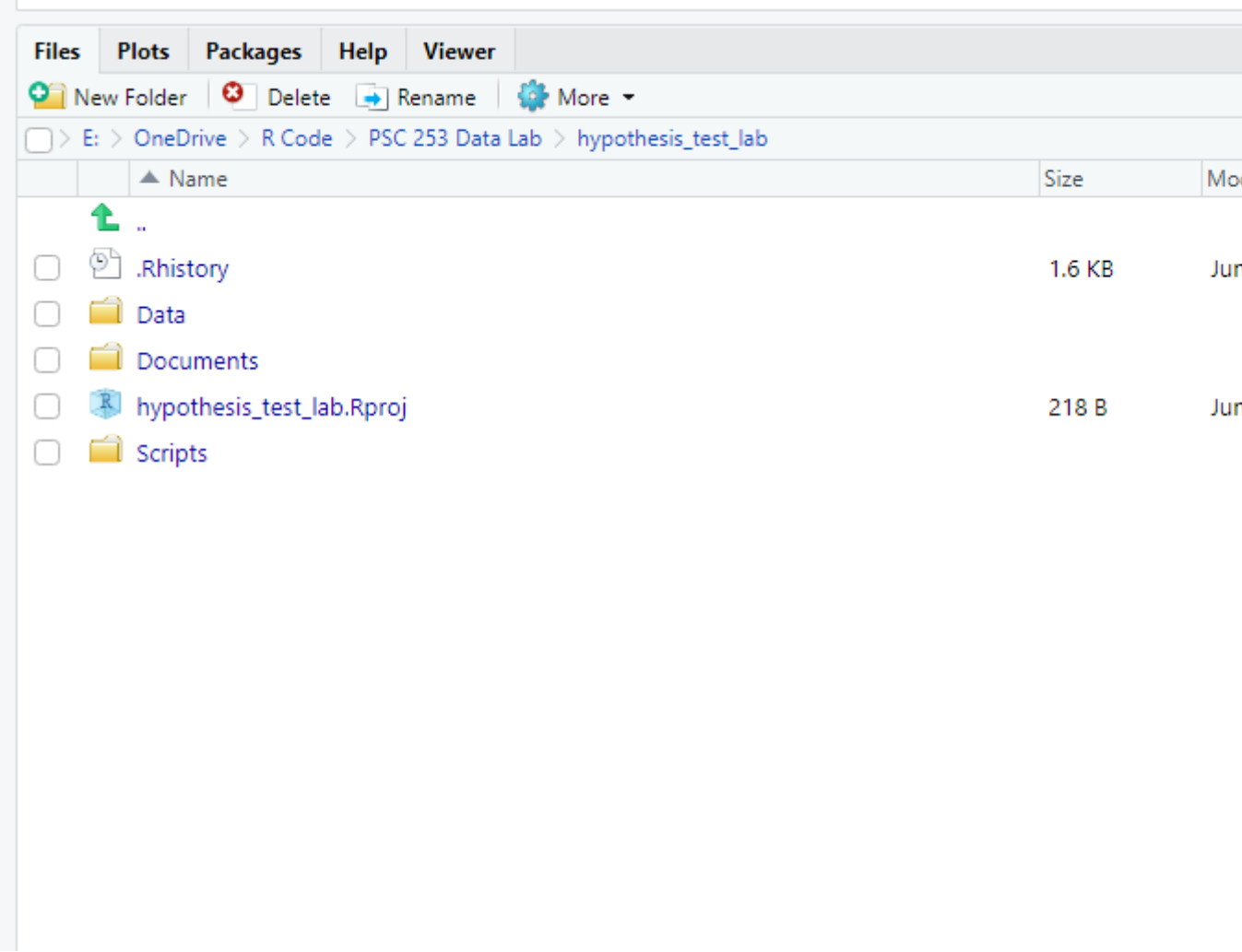


Figure 1.1: A screen shot of the hypothesistestlab project folder.

3. In this course, `.csv` files should always be located in the `Original_Data` subfolder, which is found inside the "Data" folder.
4. Create an object to hold the data that you will import. This object will be assigned the imported data.
5. Use the function `read.csv()` or `read_csv()` to import the data into R. Its argument is the file path.

Generically, the code takes the form:

```
your_object <- read.csv(here("Data/Original_Data/your_data.csv"))
```

If I were loading a csv file named "congbills.csv" and assigning it to an object called "bills", then the code would be:

```
# Loading the csv file "congbills.csv" into R as an object called "bills"

bills <- read.csv(here("Data/Original_Data/congbills.csv"))
```

1.9.3 Troubleshooting

- The most common error is that the file path to the data has not been specified properly. Check the spelling in your file path.
- Make sure that you use `here()` for the file path. Start from the project folder, and then write the path until you get to your file.
- Using `read_csv()` requires the `tidyverse` package.

1.10 Loading .dta Data

Stata is another popular software program for data analysis. It is often used in economics. The types of files that are used in Stata have the suffix `.dta`.

1.10.1 Problem

You want to load a datafile that takes the form `data_name.dta`.

1.10.2 Solution

1. You already have a project folder. (Section 1.5)
2. You already know how to create relative paths. (Section 1.8)
3. In this course, `.dta` files should always be located in the `Original_Data` subfolder, which is found inside the "Data" folder.
4. Create an object to hold the data that you will import. This object will be assigned the imported data.
5. Load the package `haven` – `library(haven)`
6. Use the function `read_dta()` to import the data into R. Its argument is the file path.

Generically, the code takes the form:

```
# load the package 'haven'
library(haven)

# import the data
your_object <- read_data(here("Data/Original_Data/your_data.dta"))
```

If I were loading a dta file named "congbills.dta" and assigning it to an object called "bills", then the code would be:

```
# Loading the csv file "congbills.csv" into R as an object called "bills"

bills <- read_dta(here("Data/Original_Data/congbills.dta"))
```

1.10.3 Troubleshooting

- The most common error is that the file path to the data has not been specified properly. Check the spelling in your file path.
- Make sure that you use `here()` for the file path. Start from the project folder, and then write the path until you get to your file.
- Using `read_dta()` requires the `haven` package.

Chapter 2

Organizing for Reproducibility

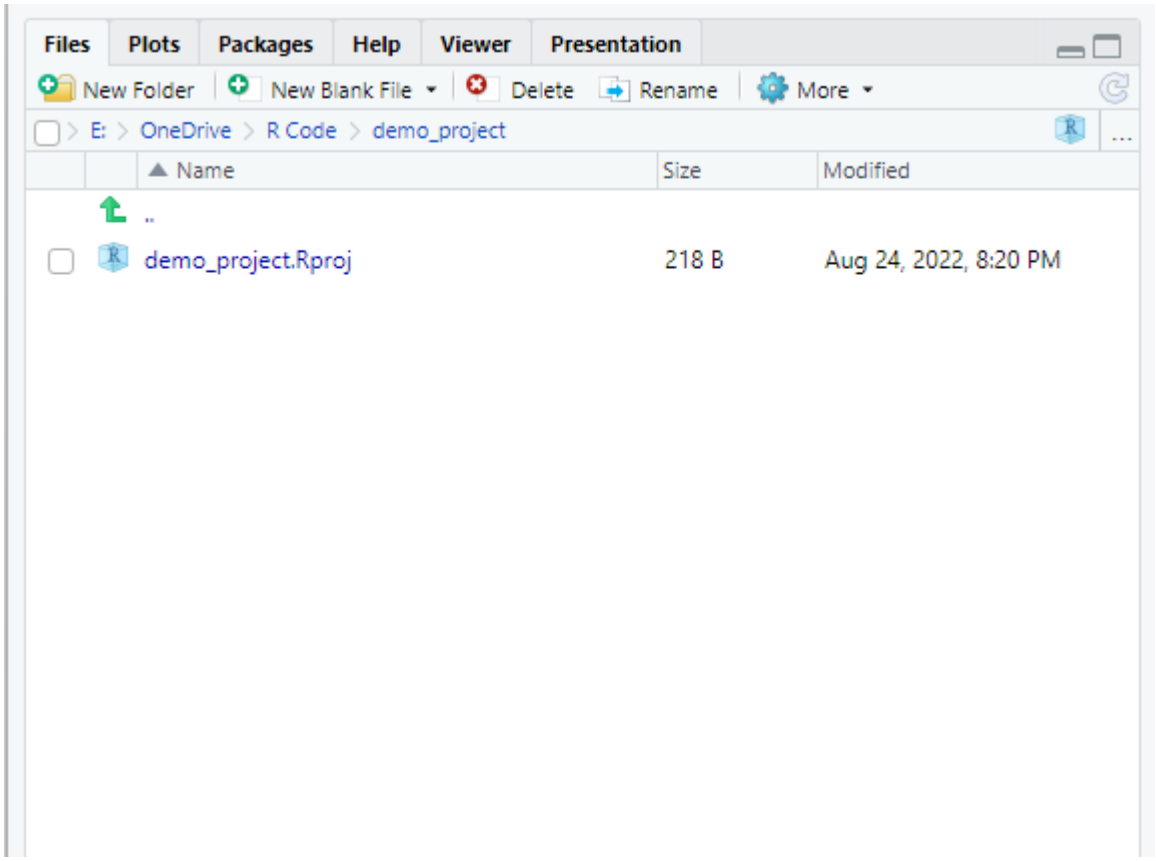
2.1 Create TIER Folders

2.1.1 Problem

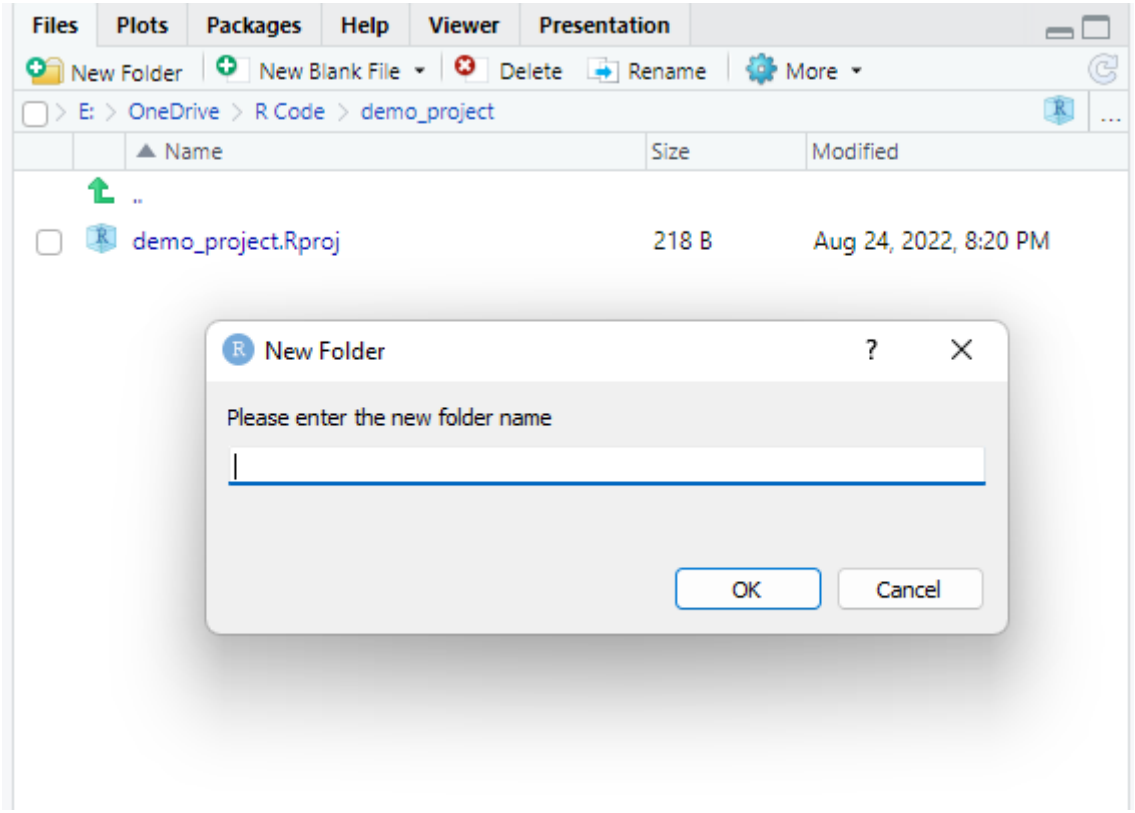
You want to create folders in accordance with Project TIER's protocol.

2.1.2 Solution

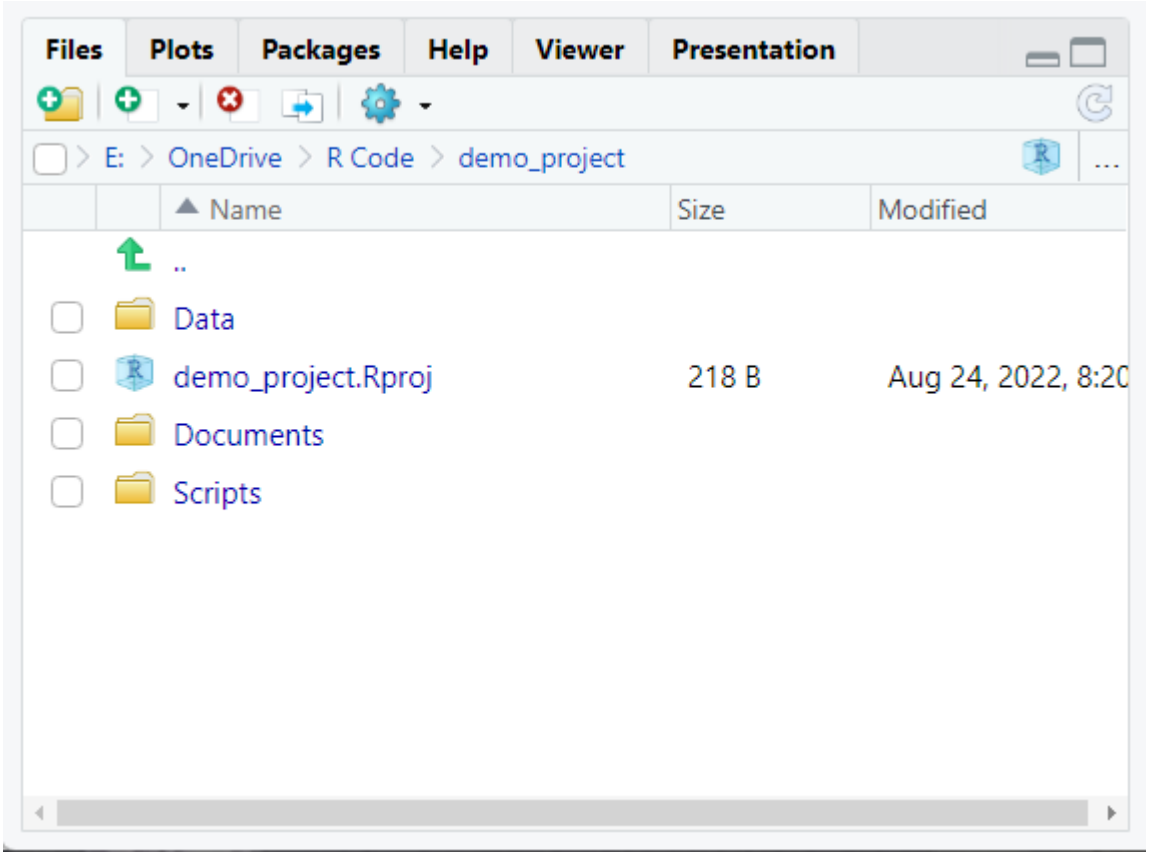
1. You should have already created a project.



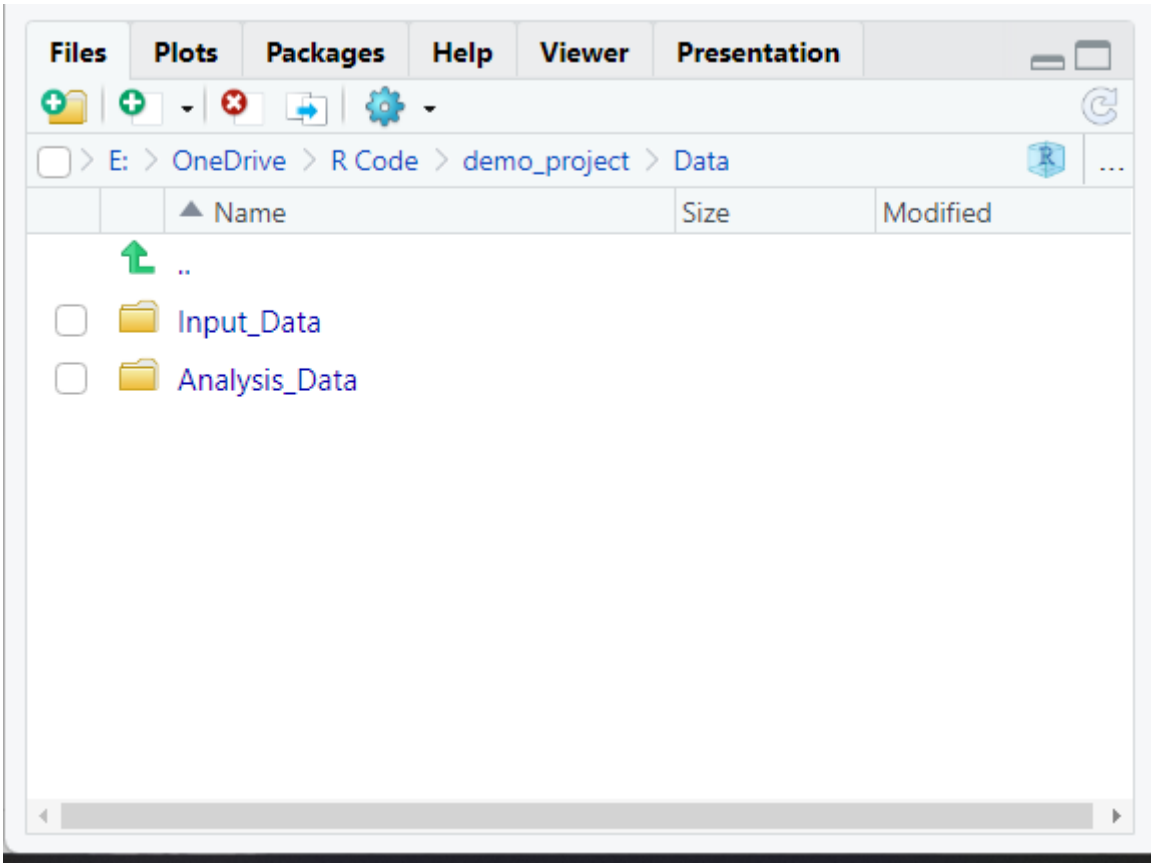
2. In the Files, Plots, Packages, Help, Viewer pane there is a button that says “New Folder.”



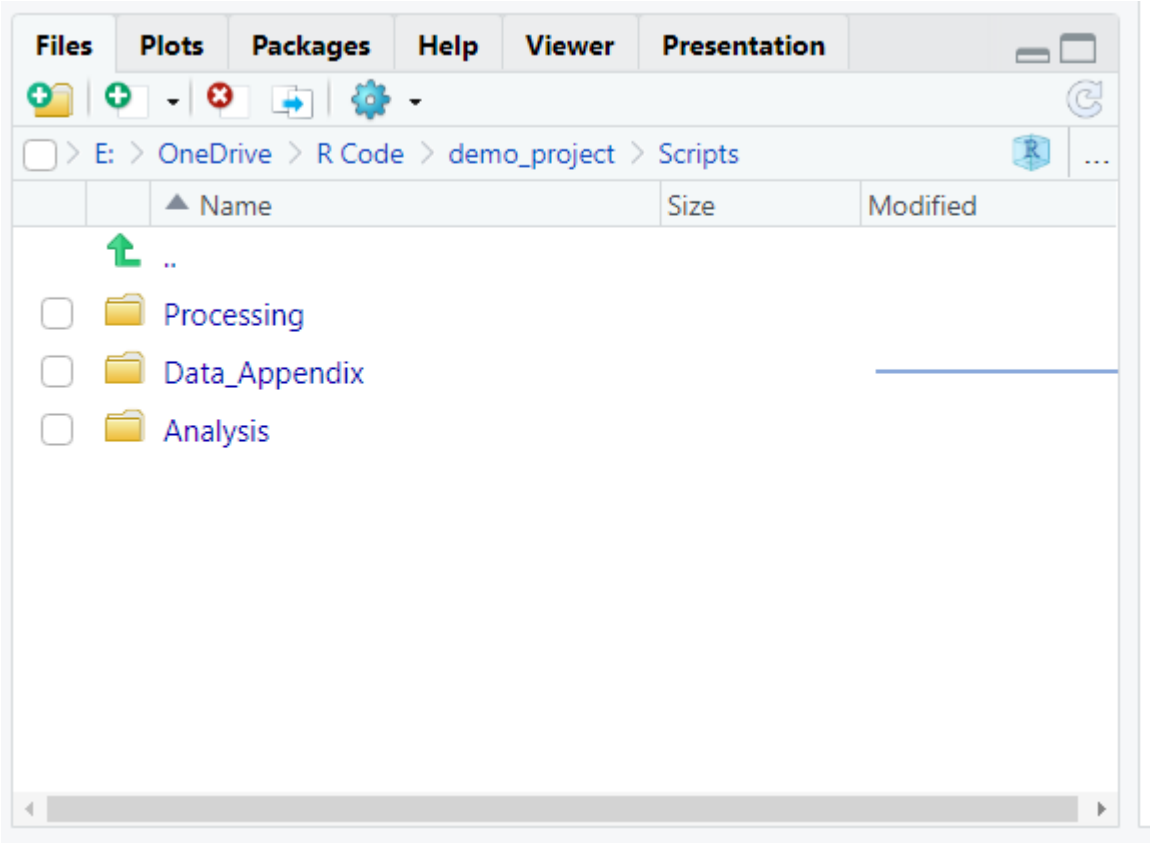
3. Click that button to create three folders named “Data”, “Documents”, and “Script”. When you are finished the project folder should look like this:



4. Inside the "Data" folder, create two folders named "Input_Data" and "Analysis_Data".



5. Inside the "Scripts" folder, create three folders named "Processing", "Data_Appendix", and "Analysis".



2.1.3 Troubleshooting

- The actual names of the folder are not important. However, you should be consistent with how you name the folders across projects. That will help you to remember the role that each folder serves.
- Keep in mind that you will use these folder names in your relative paths.

2.2 Populate TIER Folders

2.2.1 Problem

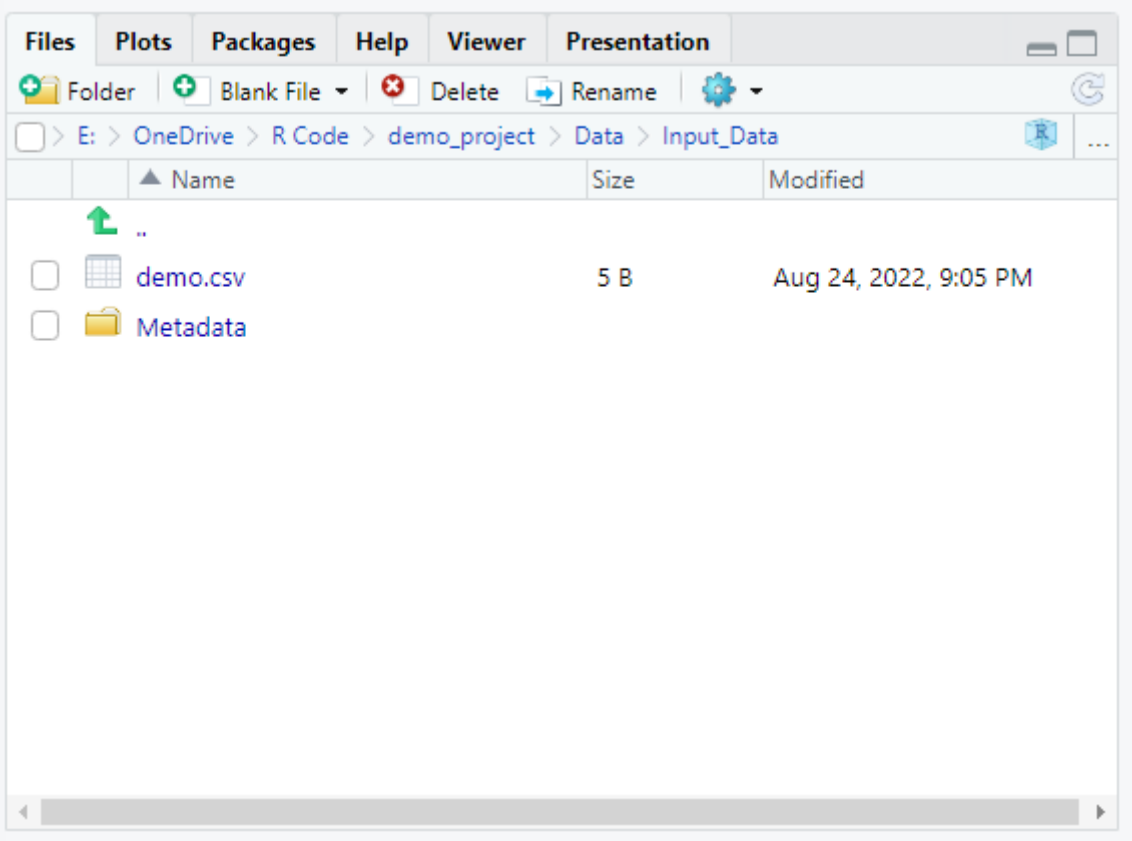
You want to put files into their appropriate TIER folder.

2.2.2 Solution

1. You have already created a project folder.
2. You have already created TIER folders inside of the project folder.

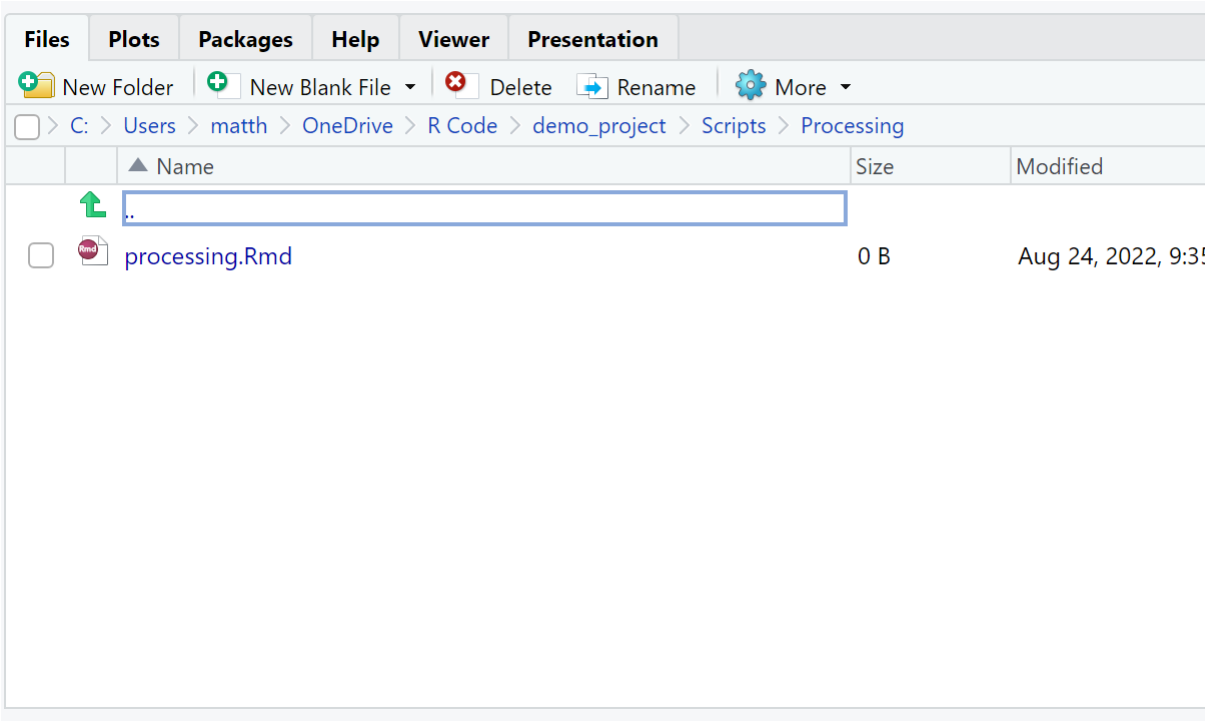
3. Now we are ready to describe what goes into each of these folders. We will start with the "Data" folder.

Inside of the "Data" folder there are two subfolders – "Input_Data" and "Analysis Data". The "Input_Data" folder is where you place the unprocessed, original version of a dataset. Also, you should create a subfolder inside of the "Input_Data" folder that is called "Metadata". Any codebooks that are associated with your dataset should be placed inside of the "Metadata" folder.

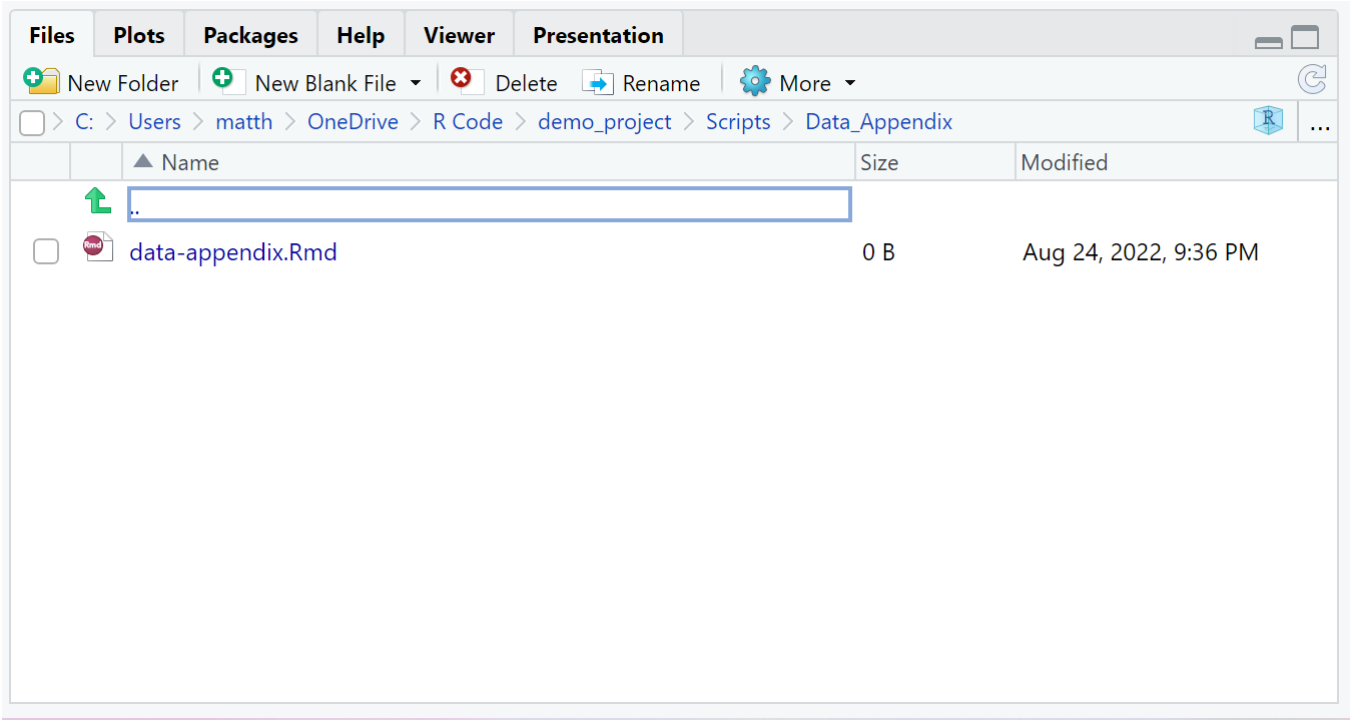


You do not place anything inside of the "Analysis_Data" folder. This folder will be populated when the input data is processed.

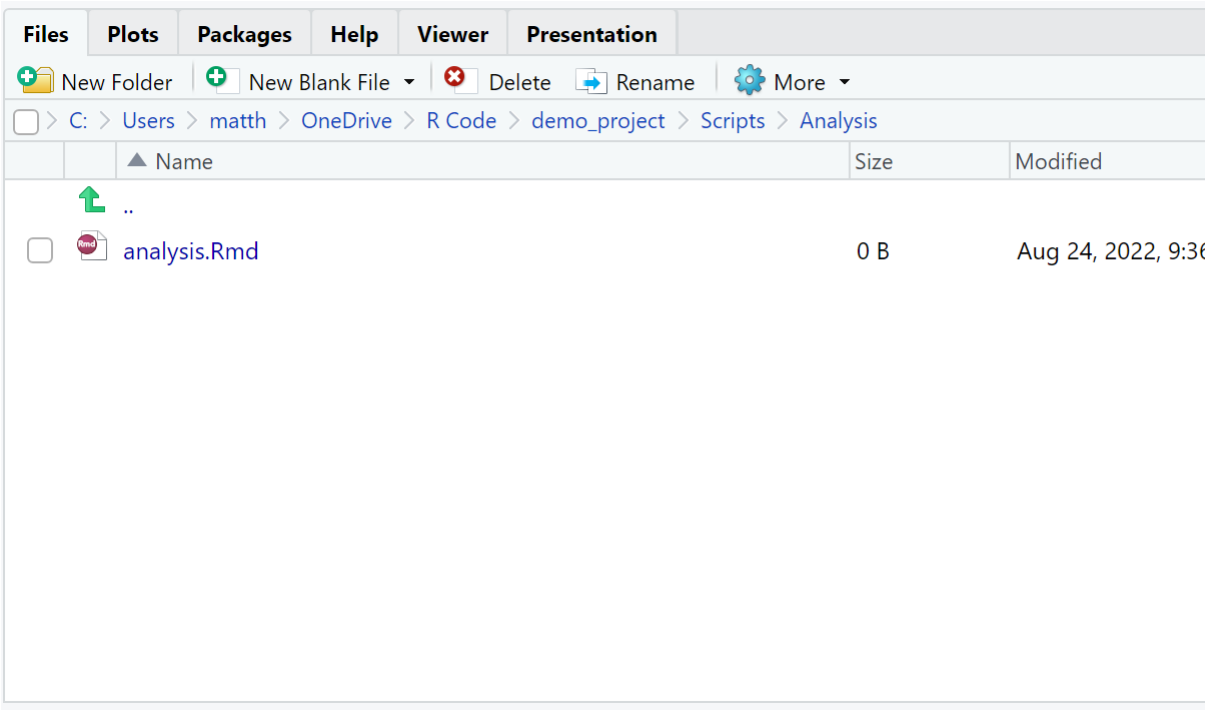
4. Inside of the "Scripts" folder there are three subfolders – "Processing", "Analysis", and "Data_Appendix". The "Processing" folder should have one file named `processing.rmd`. This file is used to transform the input data into the form that you will use for the data analysis.



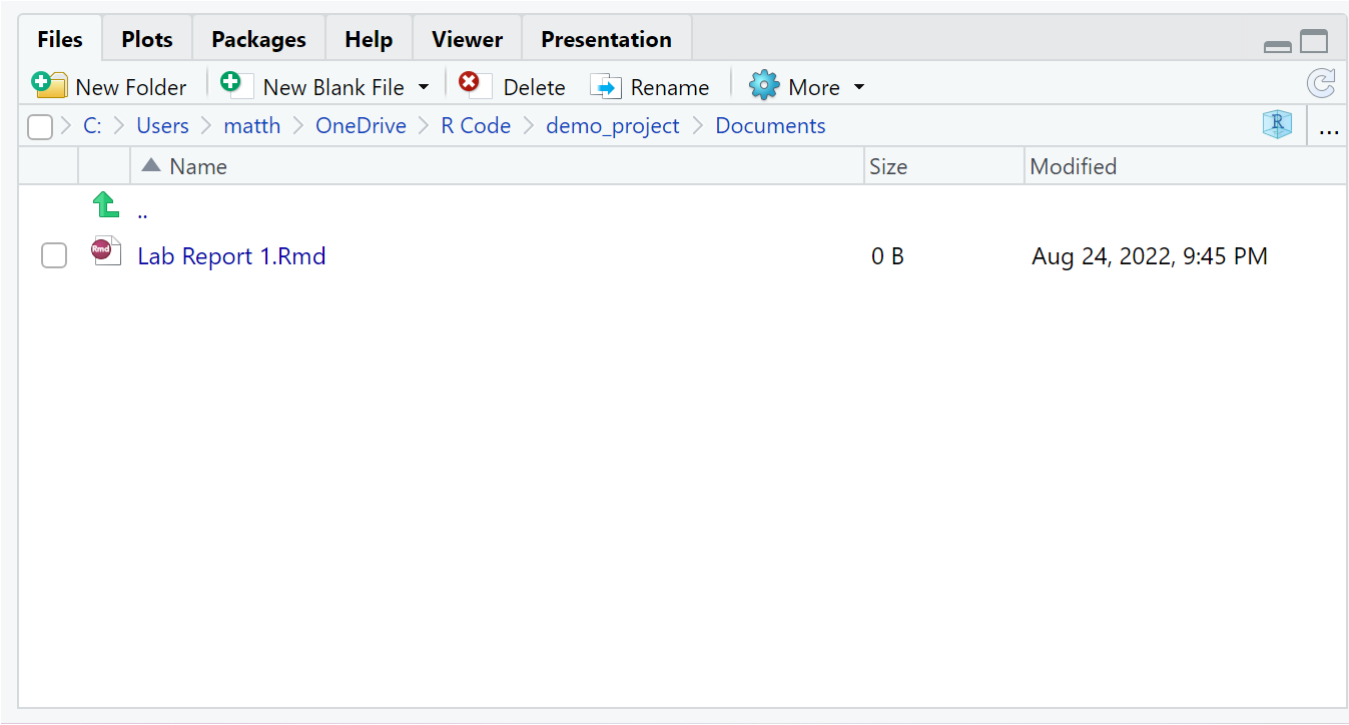
The "Data_Appendix" folder should have one filed named `data-appendix.rmd`. This file creates the Data Appendix, which serves as a codebook and provides descriptive statistics for the variables that are used in the actual data analysis.



Lastly, the "Analysis" folder should have one file named `analysis.rmd`. For the final data project, this file will contain all of the data analysis that you were required to perform as part of that project. For the weekly labs, the `analysis.rmd` provides sample code for any analysis that you will be required to perform in the lab report.



5. The "Documents" folder contains the finished version of your work – the lab report, the research paper, or the final poster. This folder may also contain any supplementary files or folders that are necessary for that final product. For example, files for the Morehouse logo that appears on the final data project poster are located inside of the "Documents" folder.



2.2.3 Troubleshooting

- Do not confuse the "Analysis" subfolder in "Scripts" with the "Analysis_Data" subfolder in "Data".
- Keep in mind that the actions described above do not have to take place within Rstudio. You can simply use the Explorer or Finder windows on your computer to achieve the same ends.

2.3 Knitting Files in Sequence

2.3.1 Problem

You want to knit your `.Rmd` files in correct order to produce the finalized report or paper.

2.3.2 Solution

1. Only the `processing.Rmd` file should work with your input data, so it is always knit first.
2. The processing script creates a file called "analysis.RData". This is the analysis data that will be loaded by all of the other scripts.

3. Knit the `data-appendix.Rmd` so that you can become familiar with the variables that you are analyzing.
4. Knit the `analysis.Rmd` third. It will serve as a guide/template to help you complete the lab resort.
5. Knit the final report.

2.3.3 Troubleshotting

- If there is a knitting error about loading data for the analysis or data-appendix script, then check to make sure that you have successfully knitted the processing script. Look in the "Analysis_Data" folder to make sure that the "analysis.RData" has been created.
- The relative file paths assume that you are working within the project folder. Look in the top right corner to make sure that it shows that you are working in the appropriate project.

Chapter 3

Processing Data

Most of the time, we need to make some changes to a dataset to prepare it for analysis. This could involve adding new variables, "cleaning" existing variables, changing the level of measurement for a variable, altering the labels of a variable, or even combining multiple datasets. We refer to these kinds of changes as "processing" the data. This section is about the common tools that are used for data processing in this course.

3.1 Label Missing Observations in a Dataset

Sometimes we will work with survey data that has observations with numeric codes that are not aligned with a response of interest. For example, the numeric code -9 might signify that a person did not actually answer the survey question. We want to label this particular observation as "missing".

3.1.1 Problem

You want to label missing observations.

3.1.2 Solution

1. These instructions begin with the premise that you already know which variable values correspond to missing values.The codebook should indicate which codes correspond to missing data.
2. Use indexing to isolate the observations that have missing values.
3. Assign those observations a value of `NA`.

```
dataset$variable[dataset$variable == missing_value] <- NA
```

Here we are looking at the variable `hillary_therm` in the `anes` dataset, and we label all observations that take the value -9 as missing.

```
# labeling -9 as missing
anes$hillary_therm[anes$hillary_therm == -9] <- NA
```

3.1.3 Troubleshooting

- In order to label the missing observations you have to first understand what all of the values of the variable are and/or should be. That information comes from the codebook for the data. For example, if you know that a variable is supposed to take values from 0 to 10, then a value of -99 is probably a code for a missing observation.
- You will generally want to look at a frequency table prior to labeling the missing values and after labeling the missing values. That will indicate whether you have done it correctly.

3.2 Add a New Variable to a Dataset

When we want to make a change to a variable, it is a good practice to create a new variable rather than changing the existing variable.

3.2.1 Problem

You want to add a variable to a dataset.

3.2.2 Solution - Basic

1. Provide a name for the new variable.
2. Use the `$` operator to create the variable in the data `data$new_variable_name`.
3. Assign values to the new variable. Often we are assigning the new variable the values of the old variable as a preliminary step to some other kind of transformation/processing. `data$new_variable <- data$old_variable`.

```
# assign an old variable to a new variable name
dataset$new_variable <- dataset$old_variable
```

Below we assign the values of the variable `ft_hclinton` to a new variable called `hillary_therm`. The dataset is named "nes".

```
nes$hillary_therm <- nes$ft_hclinton
```

3.2.3 Solution - tidyverse

1. Assign the dataset to itself.
2. Use the pipe operator `%>%` at the end of that line of code.
3. Use the `mutate()` function.

4. The argument for the function is setting the name of the new variable and then assigning it values using an equal sign.

```
# use mutate to turn an old variable into a new variable

dataset <- dataset %>%
  mutate(new_variable = old_variable)
```

Below we assign the values of the variable `ft_hclinton` to a new variable called `hillary_therm`. The dataset is named "nes".

```
# assign data to itself
nes <- nes %>%

  # use mutate() to assign values to the new variable
  mutate(hillary_therm = ft_hclinton)
```

3.2.4 Troubleshooting

- No common mistakes yet.

3.3 Create an Additive Index

3.3.1 Problem

You want to make some arithmetic adjustment to a variable.

3.3.2 Solution

1. You have already added a variable to a dataset
2. Inside your `mutate()` function add, subtract, multiply, or divide variables by other variables or by constants.

```
# assign dataset to itself
dataset <- datasets %>%

  # use mutate() with some arithmetic in the argument
  mutate(new_variable = old_variable + some_number)
```

Below we create a variable `avg_demfeel` that is the average feeling thermometer of `obama_therm`, `ft_dem`, and `ft_hclinton`.

```
# assign dataset to itself
nes <- nes %>%

  # use mutate() to add the three variables and divide to get the average
  mutate(avg_demfeel = (obama_therm + ft_dem + ft_hclinton)/3)
```

3.3.3 Troubleshooting

- It may be helpful to test out your math problem first to make sure it gives you the results you are looking for.

3.4 Create a Factor Variable

In R, nominal level variables are called "factors." This section explains how to create a factor.

3.4.1 Problem

You want to transform a variable into a factor.

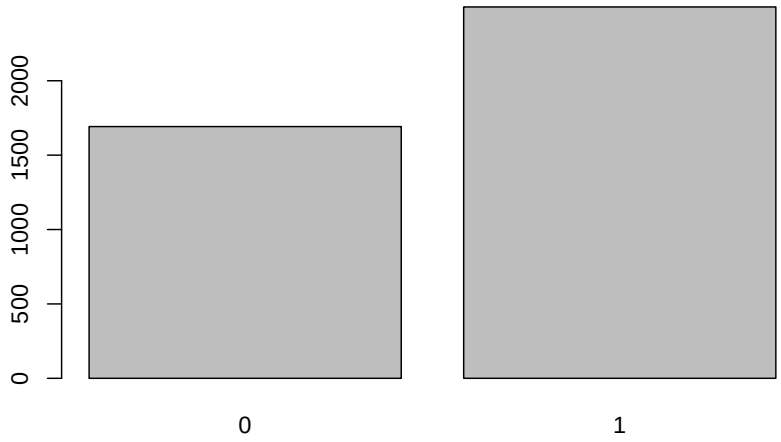
3.4.2 Solution

1. Decide on what to name the factor variable.
2. Use `as.factor()` to assign the old variable values to the new variable. See Section 3.2.
3. Use `levels()` to assign labels to the values of the variable.
4. The levels should be provided as a list in `c()` with the names of the levels in quotation marks.
5. It would follow the general template below.

```
# define the variable as a factor

data$newvariable <- as.factor(data$oldvariable)

# add the levels

levels(data$newvariable) <- c("label1", "label2", "labelk")
```

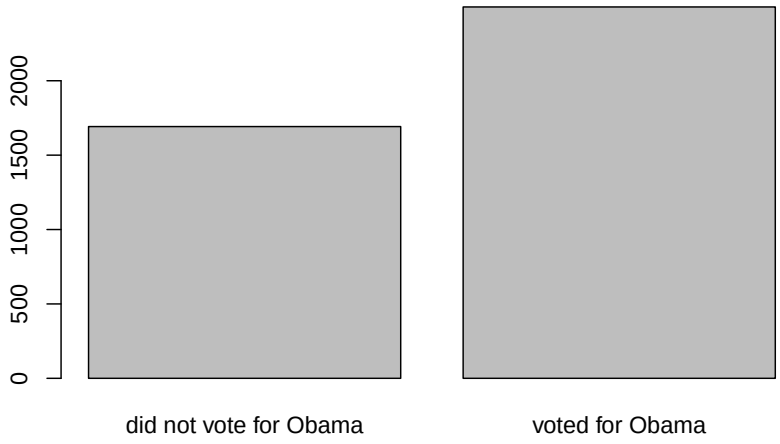
Transform `obama_vote` in the `nes` from a numeric dummy variable into a factor.

```
# the frequency table for the old variable
freq(nes$obama_vote)
```




```
## nes$obama_vote
##       Frequency Percent Valid Percent
## 0          1692   28.60          40.4
## 1          2496   42.19          59.6
## NA's       1728   29.21
## Total      5916  100.00         100.0
```

```
# call the new variable `obamafct`
nes$obamafct <- as.factor(nes$obama_vote)

# assign the levels
levels(nes$obamafct) <- c("did not vote for Obama", "voted for Obama")

# the frequency table for the new variable
freq(nes$obamafct)
```




```
## nes$obamafct
##                         Frequency Percent Valid Percent
## did not vote for Obama       1692   28.60          40.4
## voted for Obama              2496   42.19          59.6
## NA's                         1728   29.21
## Total                        5916  100.00         100.0
```

3.4.3 Troubleshooting

- There has to be a level provided for each value of the variable. That is why this process should occur after a variable has been cleaned.
- The level names do not have to be unique. That means this could be used as a (somewhat clunky) method of recoding a variable by collapsing its categories.

3.5 Create a Numeric Variable

Interval level variables are classified as "numeric".

3.5.1 Problem

You want to transform an existing variable into a numeric variable.

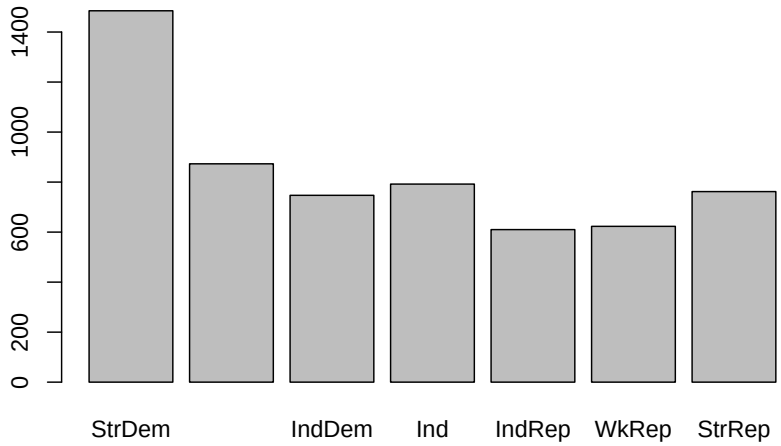
3.5.2 Solution

1. Decide on what to name the numeric variable. It could be the same name as the old variable.
2. Use `as.numeric()` to assign the old variable values to the new variable.
3. It would follow the general template below.

```
data$newvariable <- as.numeric(data$oldvariable)
```

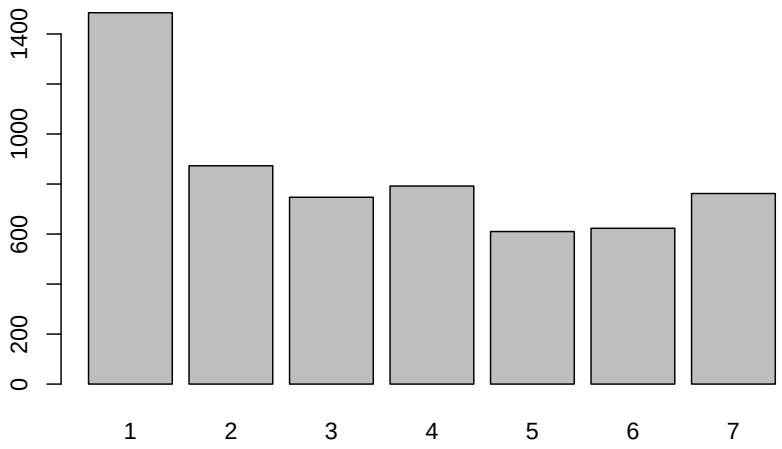
Here is an example that converts the seven-point party identification scale into a numeric variable.

```
# frequency table of the original variable `pid_x`
freq(nes$pid_x)
```




```
## nes$pid_x
##        Frequency  Percent Valid Percent
## StrDem      1485  25.1014         25.20
## WkDem        873  14.7566         14.82
## IndDem       747  12.6268         12.68
## Ind          792  13.3874         13.44
## IndRep       610  10.3110         10.35
## WkRep        623  10.5308         10.57
## StrRep       762  12.8803         12.93
## NA's          24   0.4057
## Total       5916 100.0000        100.00
```

```
# make `pid_x` numeric
nes$pid7 <- as.numeric(nes$pid_x)

# frequency table of the new variable `pid7`
freq(nes$pid7)
```




```
## nes$pid7
##       Frequency  Percent Valid Percent
## 1          1485  25.1014         25.20
## 2           873  14.7566         14.82
## 3           747  12.6268         12.68
## 4           792  13.3874         13.44
## 5           610  10.3110         10.35
## 6           623  10.5308         10.57
## 7           762  12.8803         12.93
## NA's         24   0.4057
## Total      5916 100.0000        100.00
```

3.5.3 Troubleshooting

- Have not come across any problems yet.

3.6 Create an Ordinal Variable

3.6.1 Problem

You want to transform an existing variable into an ordinal variable.

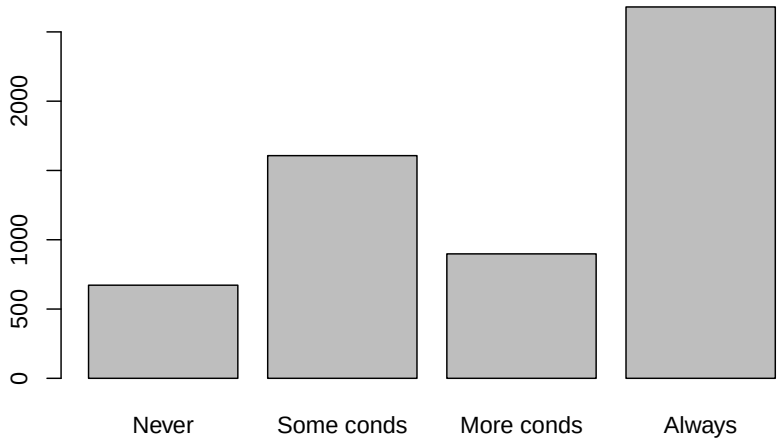
3.6.2 Solution

1. Decide on what to name the ordinal variable. It could be the same name as the old variable. However, this could become confusing in your code.
2. Use `as.ordered()` to assign the old variable values to the new variable.
3. Use `levels()` to assign labels to the values of the variable.
4. The levels should be provided as a list in `c()` with the names of the levels in quotation marks.
5. It would follow the general template below.

```
# define the variable as ordinal

data$newvariable <- as.ordered(data$oldvariable)

# add the levels

levels(data$newvariable) <- c("label1", "label2", "labelk")
```

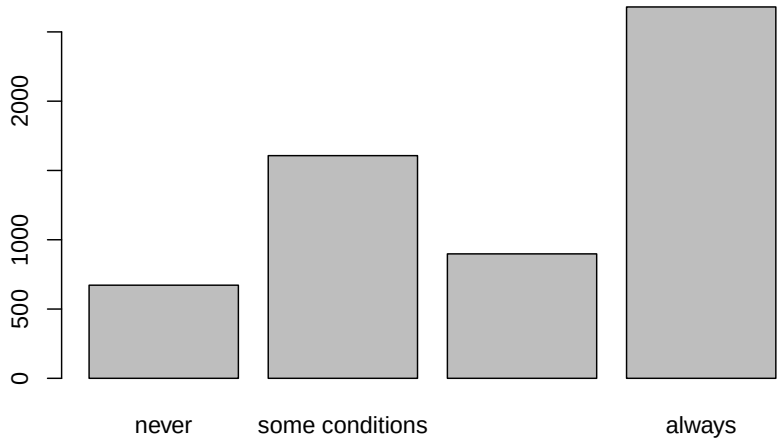
Transform `abort4` in the `nes` from a factor into an ordinal variable.

```
# the frequency table for the old variable
freq(nes$abort4)
```




```
## nes$abort4
##            Frequency  Percent Valid Percent
## Never            672  11.3590         11.47
## Some conds      1607  27.1636         27.44
## More conds       898  15.1792         15.33
## Always          2680  45.3009         45.76
## NA's              59   0.9973
## Total           5916 100.0000        100.00
```

```
# call the new variable `abort4`
nes$abort4 <- as.ordered(nes$abort4)

# assign the levels
levels(nes$abort4) <- c("never", "some conditions", "more conditions", "always")

# the frequency table for the new variable
freq(nes$abort4)
```




```
## nes$abort4
##                  Frequency  Percent Valid Percent Cum Percent
## never                  672  11.3590         11.47       11.47
## some conditions       1607  27.1636         27.44       38.91
## more conditions        898  15.1792         15.33       54.24
## always                2680  45.3009         45.76      100.00
## NA's                    59   0.9973
## Total                 5916 100.0000        100.00
```

3.6.3 Troubleshooting

- There has to be a level provided for each value of the variable. That is why this process should occur after a variable has been cleaned.
- The key reason for treating a variable as ordinal rather than nominal in R is to calculate the "cumulative percent" column in a frequency table.

3.7 Create a Subset of Data

3.7.1 Problem

You want to create a subset of your data based on some criteria.

3.7.2 Solution - Basic

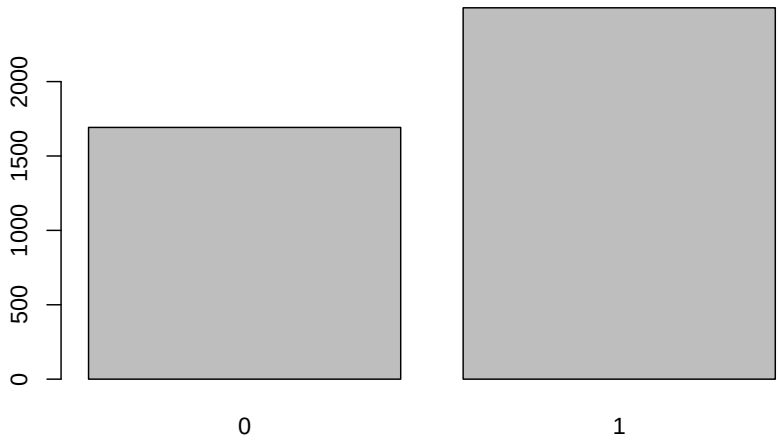
1. Provide a name for the subset of data you are creating.

2. Use `subset()`. The main arguments are the data object that is being subsetted, the criteria for determining which observations to include in the subset, and the selection of columns for inclusion in the new data object.
3. The criteria use relational logic such as `==`, `>`, `!=`, etc.
4. Follow the template below.

```
newdata <- subset(old_data, criteria, select = c(list of columns))
```

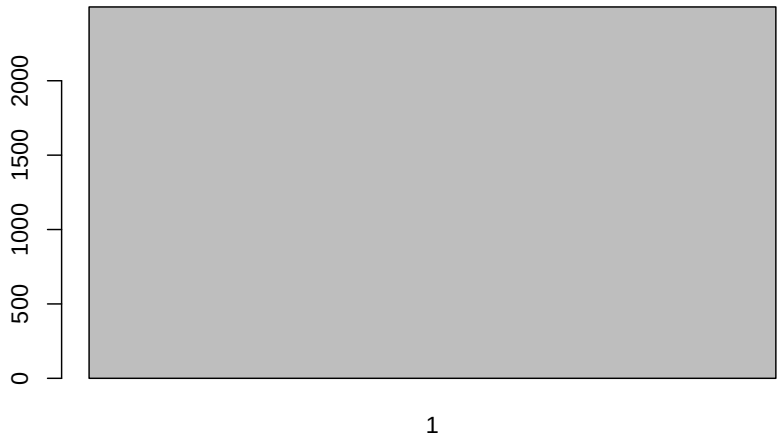
Here we create a subset that only includes people who voted for Obama.

```
# frequency of `obama_vote` in full dataset
freq(nes$obama_vote)
```




```
## nes$obama_vote
##       Frequency Percent Valid Percent
## 0          1692   28.60          40.4
## 1          2496   42.19          59.6
## NA's       1728   29.21
## Total      5916  100.00         100.0
```

```
# create a subset of `nes` data that only includes Obama voters
ovoters <- subset(nes, obama_vote == 1)

# frequency of a `obama_vote` in subset
freq(ovoters$obama_vote)
```




```
## ovoters$obama_vote
##       Frequency Percent
## 1          2496     100
## Total      2496     100
```

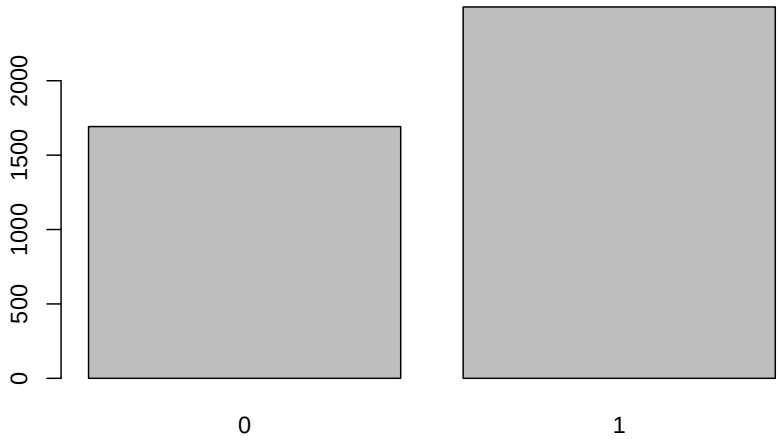
3.7.3 Solution - Tidyverse

1. Provide a name for the subset of data you are creating.
2. Assign the existing dataset to the new data object.
3. Use the pipe `%>%`.
4. Use `filter()`. The pipe inherits the dataset from the prior step, so the only argument is the criteria for the subset.
5. The criteria use relational logic such as `==`, `>`, `!=`, etc.
6. Follow the template below.

```
newdata <- old_data %>%
  filter(criteria)
```

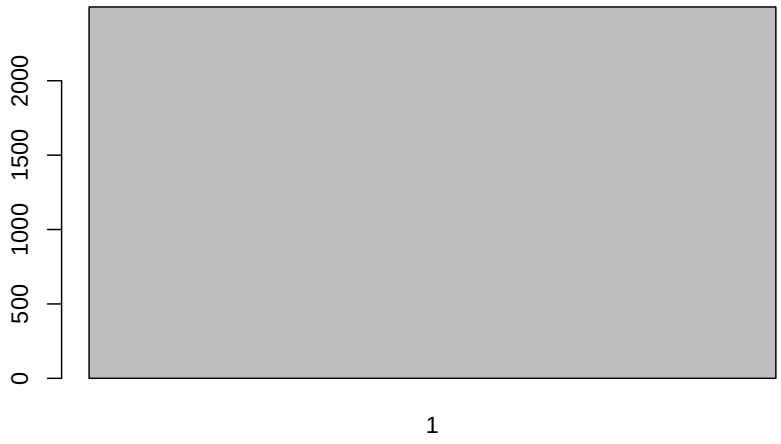
Here we create a subset that only includes people who voted for Obama.

```
# frequency of `obama_vote` in full dataset
freq(nes$obama_vote)
```




```
## nes$obama_vote
##       Frequency Percent Valid Percent
## 0          1692   28.60          40.4
## 1          2496   42.19          59.6
## NA's       1728   29.21
## Total      5916  100.00         100.0
```

```
# create a subset of `nes` data that only includes Obama voters
obamites <- nes %>%
  filter(obama_vote == 1)

# frequency of a `obama_vote` in subset
freq(obamites$obama_vote)
```




```
## obamites$obama_vote
##       Frequency Percent
## 1          2496     100
## Total      2496     100
```

3.7.4 Troubleshooting

- Make sure that your subset is assigned to something.
- The criteria is written as variable, logical operator, and value of the variable.
- The values of categorical variables must be in quotation marks.

3.8 Summarize Data Using Means

3.8.1 Problem

You want to make a summary dataset that shows the mean of one variable for each value of some other variable.

3.8.2 Solution

1. Assign the old data to a new data object.
2. Use the pipe `%>%` at the end of the line of code.
3. Use `group_by()`. The argument is the variable that you want to use to find the means of some other variable.

4. Use the pipe %>% at the end of the line of code.
5. Use summarise().
6. Create a name for the summary variable.
7. Set the summary variable as being equal to the mean of the variable that you want to take the mean of.

```
newdata <- old_data %>%
  group_by(group_variable) %>%
  summarise(summary_variable = mean(mean_variable))
```

Here we calculate the mean feelings towards Obama, obama_therm, by party identification, pid_x.

```
# assign old data to a new data object
partymeans <- nes %>%

  # use group_by() to group the data by pid_x
  group_by(pid_x) %>%

  # calculate the means of obama_therm
  summarise(average = wtd.mean(obama_therm, na.rm = T,
                               weights = wt))

# show the summary dataset
partymeans
```

```
## # A tibble: 8 x 2
##   pid_x  average
##   <fct>    <dbl>
## 1 StrDem    90.1
## 2 WkDem     74.2
## 3 IndDem    75.2
## 4 Ind       53.3
## 5 IndRep    32.8
## 6 WkRep     36.7
## 7 StrRep    17.6
## 8 <NA>      61.0
```

3.8.3 Troubleshooting

- For survey data use wtd.mean(). Use mean() for non-survey data.

Chapter 4

Bivariate Comparisons

4.1 Crosstab

4.1.1 Problem

You want to make a crosstab.

4.1.2 Solution

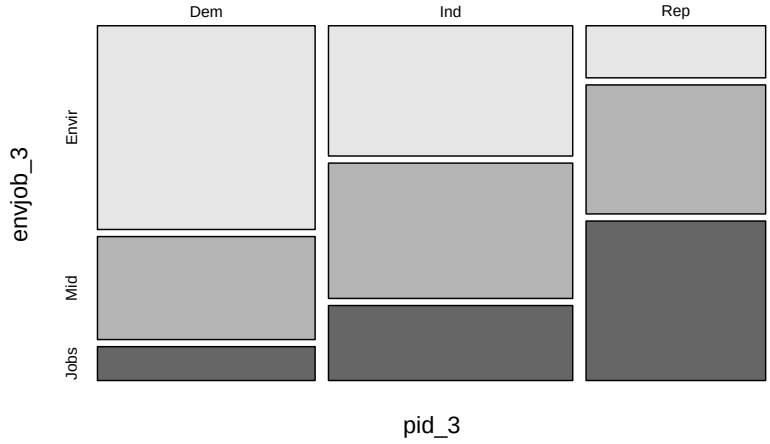
1. Load the `poliscidata` package.
2. In order to create a crosstab in R, we use the function called `xtp()`.

3. The function follows the following template:

```
xtp(data = your data, y = dependent variable, x = independent variable, w = weights)
```

1. Specify the dataset, the dependent variable, the independent variable, and the weights (if applicable)

```
# Create a crosstab where the dependent variable is "envjob_3", the independent variable is "pid_
# The dataset is "nes", which is found in the "poliscidata" package.
xtp(data = nes, y = envjob_3, x = pid_3, w = wt)
```




```
##    Cell Contents
## |-------------------------|
## |                   Count |
## |          Column Percent |
## |-------------------------|
##
## ==========================================
##             pid_3
## envjob_3       Dem      Ind      Rep   Total
## ------------------------------------------
## Envir         1005      721      212    1938
##              59.82%   38.25%   15.30%
## ------------------------------------------
## Mid            508      749      525    1782
##              30.24%   39.73%   37.88%
## ------------------------------------------
## Jobs           167      415      649    1231
##               9.94%   22.02%   46.83%
## ------------------------------------------
## Total         1680     1885     1386    4951
##              33.93%   38.07%   27.99%
## ==========================================
```

4.1.3 Troubleshooting

- Make sure that the `poliscidata` package is loaded. Use `library(poliscidata)` to load.
- The arguments for the independent and dependent variables are just the variable names. They do not follow the template of `data$variable`.
- Crosstabs are used when both the independent and dependent variables are categorical. Avoid making a crosstab with numeric data.

4.2 Comparison of Means

4.2.1 Problem

You want to make a comparison of means table.

4.2.2 Solution

1. Load the `poliscidata` package.
2. In order to create a comparison of means table in R, we use the function called `compmeans()`.

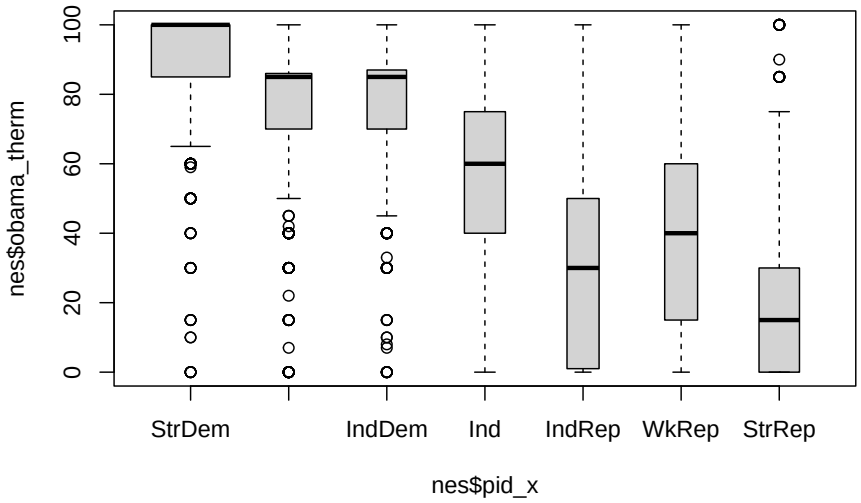
3. The function follows the following template:

```
compmeans(x = data$dependent, f = data$independent)
```

Here is a comparison of means table for feelings towards Obama `obama_therm` by party identification `pid_x`.

```
# create the comparison of means table
compmeans(x = nes$obama_therm, f = nes$pid_x,
          weights = nes$wt)
```

```
## Warning in descr::compmeans(...): 442 rows with missing values dropped
```




```
## Mean value of "nes$obama_therm" according to "nes$pid_x"
##            Mean    N Std. Dev.
## StrDem 91.12934 1384  14.19526
## WkDem  75.39237  813  22.08436
## IndDem 76.11223  695  20.22679
## Ind    54.56354  724  29.85396
## IndRep 32.15929  565  27.71165
## WkRep  36.67241  580  27.85999
## StrRep 18.44039  713  22.65842
## Total  60.72470 5474  34.64432
```

4.2.3 Troubleshooting

- Make sure `poliscidata` is loaded.
- Keep in mind that the variables follow the pattern of `data$variable`.
- You only need to specify the weights argument if you have survey data with survey weights.

4.3 Make a Bar Chart

4.3.1 Problem

Make a barchart that plots the mean of a dependent variable (Y) by an independent variable (X).

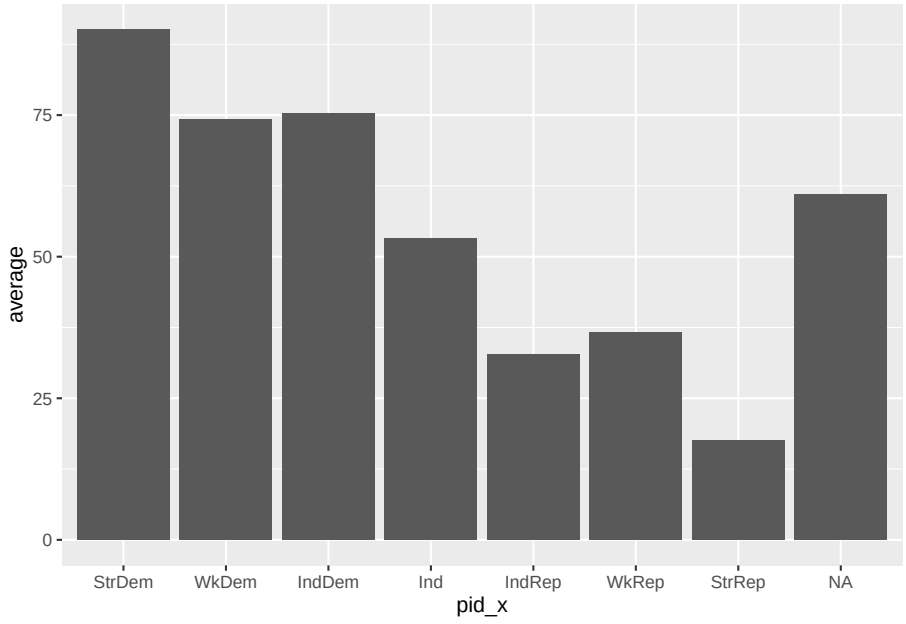
4.3.2 Solution

1. Create a summary dataset to plot (see 3.8).
2. Use `ggplot()` for the data and the mapping.
3. The data is the summary dataset, `data = sum_data`
4. The mapping is `mapping = aes(x = independent, y = dependent)`
5. If you want to color the bars, add `fill = independent` to the mapping.
6. Use a `+` at the end of the line of code.
7. Use `geom_col()` to make the bar shapes.

```
p1 <- ggplot(data = sum_data,
             mapping = aes(x = independent, y = dependent)) +
  geom_col()
```

Plot the mean feelings towards Obama `obama_therm` by party identification `pid_x`.

```
# create the summary dataset
partymeans <- nes %>%
  group_by(pid_x) %>%
  summarise(average = wtd.mean(obama_therm, na.rm = T,
                               weights = wt))

# specify the data and mapping
p1 <- ggplot(data = partymeans,
             mapping = aes(x = pid_x, y = average)) +

  # add the bar shape
  geom_col()

# print the plot
p1
```



4.3.3 Troubleshooting

- Make sure that you use the summary dataset instead of the larger dataset.
- If there are errors in making the summary dataset, then there will be problems in the plot.
- You need to load the `tidyverse` package.