PSC 253 Minimal Manual

Matthew B. Platt

2024-10-09

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.

```r
# addition
43 + 5
```

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

```r
# subtraction
43 - 5
```

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

```r
# multiplication
43 * 5
```

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

```r
# 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
```

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

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

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

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

1.5 Creating a Project

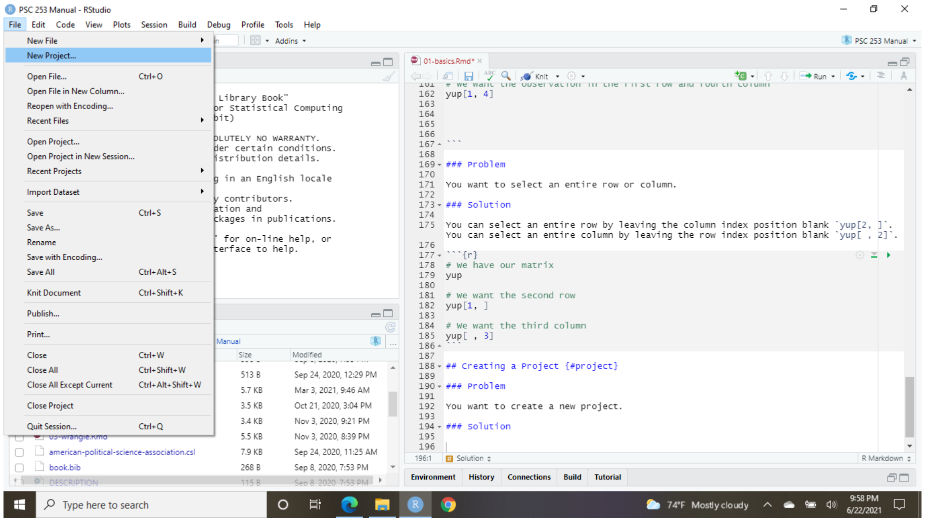
Creating a project creates a folder on your computer that Rstudio and R will treat as the default directory for your code. That is, whenever you tell R to look for something on your computer, it will begin by looking in the project folder.

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":



```
 Library Book"
or Statistical Computing
bit)

OLUTELY NO WARRANTY.
der certain conditions.
istribution details.

g in an English locale

y contributors.
ation and
ckages in publications.

' for on-line help, or
terface to help.
```

```
yup[1, 4]



```

Problem

You want to select an entire row or column.

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]`.

```{r}
# We have our matrix
yup

# We want the second row
yup[1, ]

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

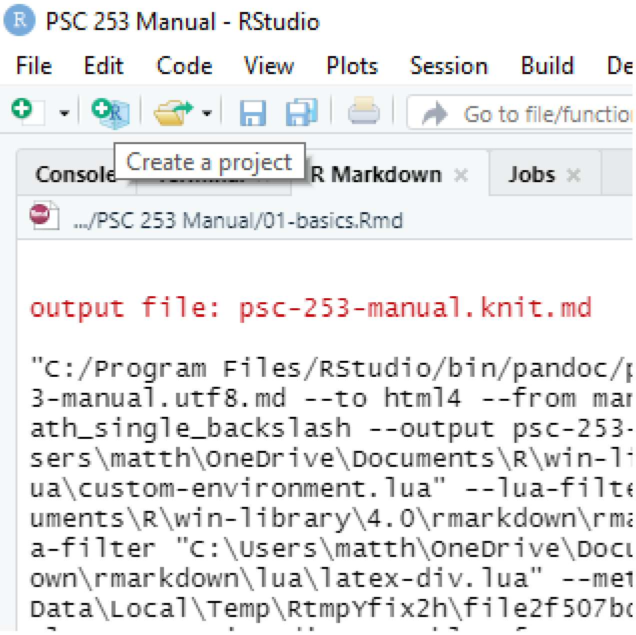
Creating a Project {#project}

Problem

You want to create a new project.

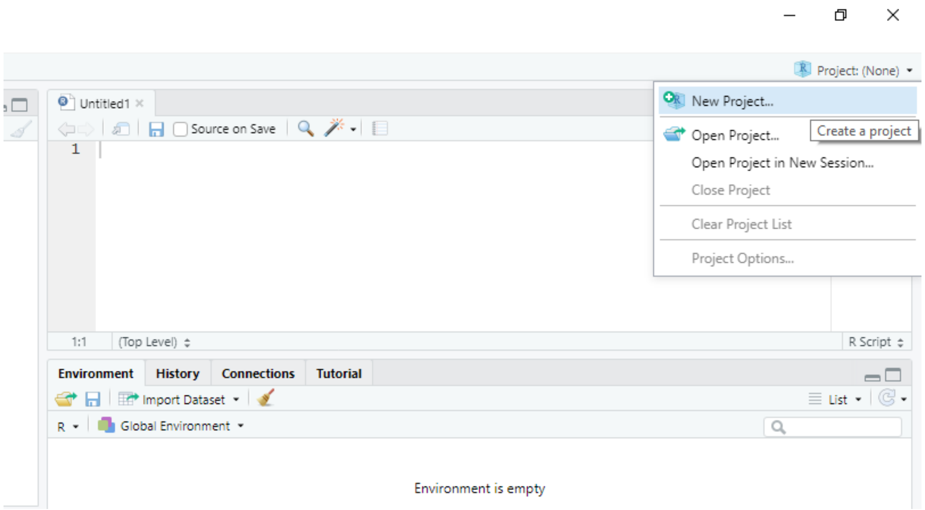
Solution

```

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

```

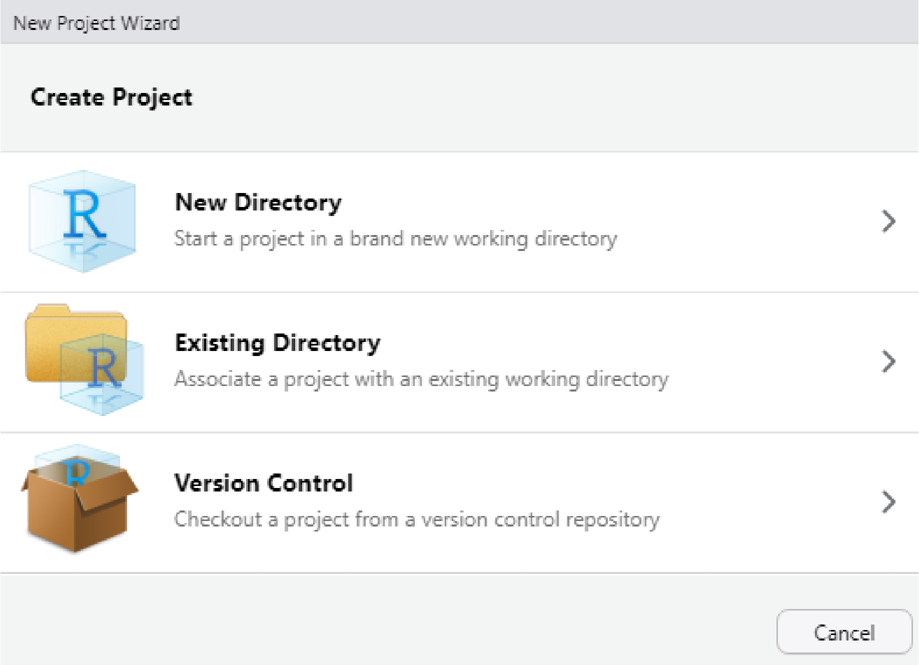


```
output file: psc-253-manual.knit.md

"C:/Program Files/RStudio/bin/pandoc/p
3-manual.utf8.md --to html4 --from mar
ath_single_backslash --output psc-253-
sers\matth\OneDrive\Documents\R\win-li
ua\custom-environment.lua" --lua-filte
uments\R\win-library\4.0\rmarkdown\rma
a-filter "C:\Users\matth\OneDrive\Docu
own\rmarkdown\lua\latex-div.lua" --met
Data\Local\Temp\RtmpYfix2h\file2f507bc
```

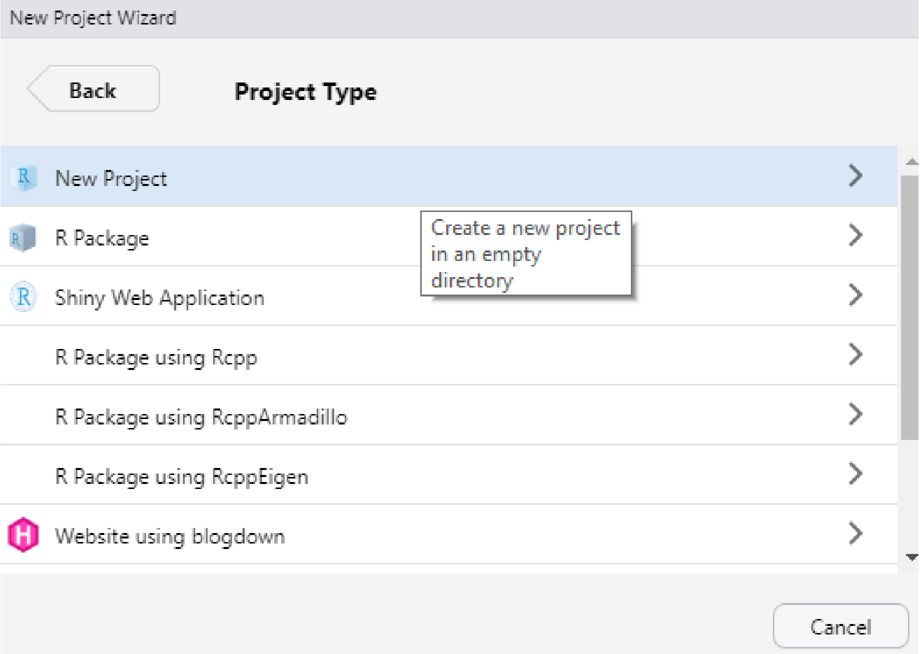
Or you can select "New Project" from the project menu at the top left of Rstudio:



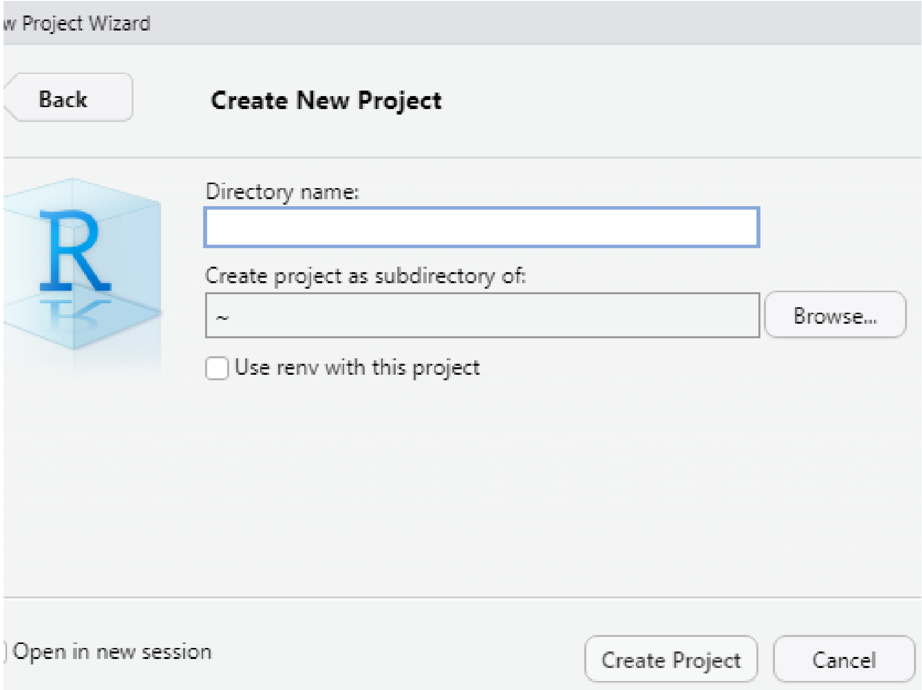
When you select "New Project", the below dialogue box appears. Select "New Directory":


New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

Then select “New project”,



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



w Project Wizard

Create New Project

Directory name:

Create project as subdirectory of: ~

☐ Use renv with this project

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 RCPA3 package

install.packages("RCPA3")
```

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 RCPA3 package

library(RCPA3)
```

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()`.

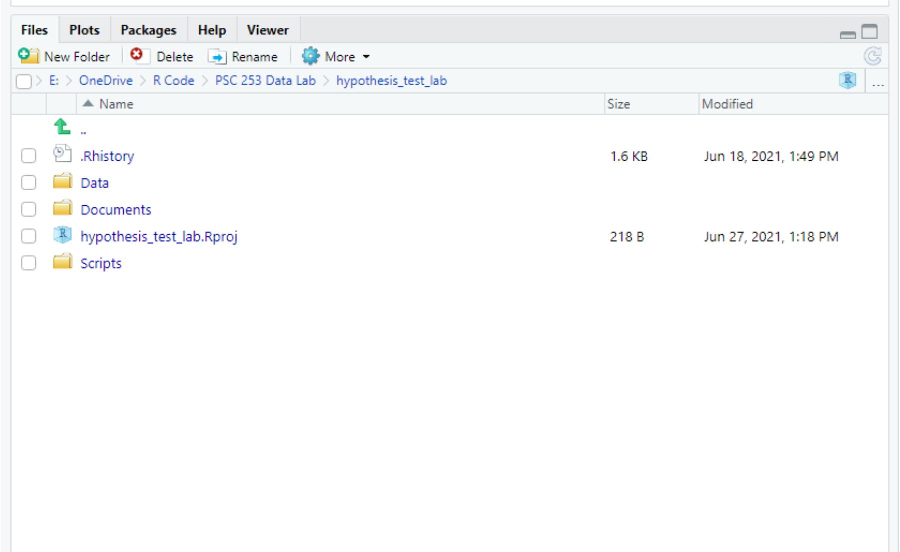
```r
# Creating a path to the "images" folder in this project.

library(here)
```

```
## here() starts at /Users/matthewplatt/Library/CloudStorage/OneDrive-Personal/R Code/PSC 253 Mar
```

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

```
## [1] "/Users/matthewplatt/Library/CloudStorage/OneDrive-Personal/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`.



E: > OneDrive > R Code > PSC 253 Data Lab > hypothesis_test_lab

Name	Size	Modified
..		
.Rhistory	1.6 KB	Jun 18, 2021, 1:49 PM
Data		
Documents		
hypothesis_test_lab.Rproj	218 B	Jun 27, 2021, 1:18 PM
Scripts		

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

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

```r
# 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)
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 dta file "congbills.dta" 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

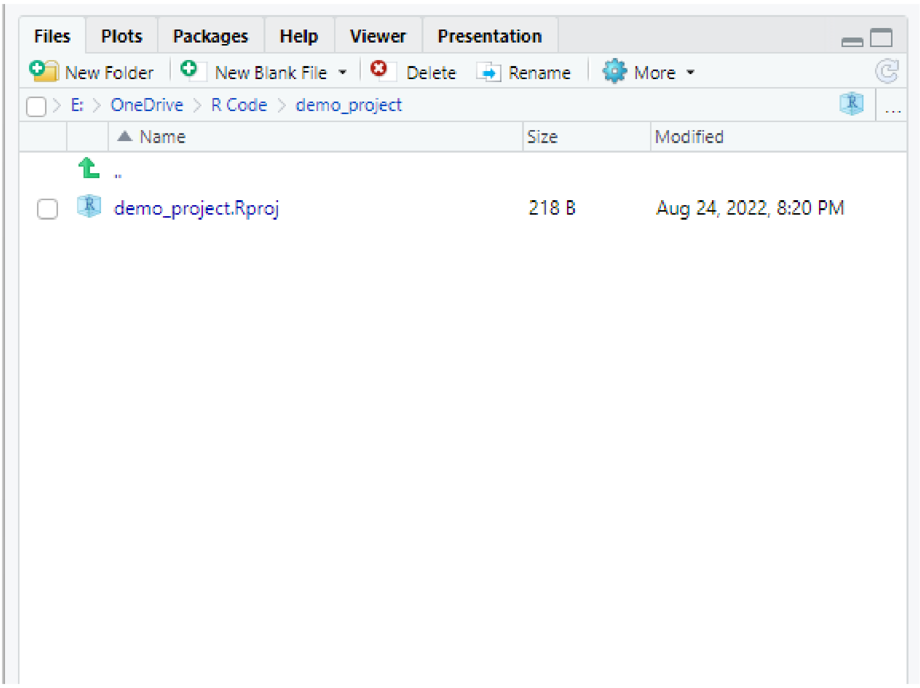
2.1 Create TIER Folders

2.1.1 Problem

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

2.1.2 Solution

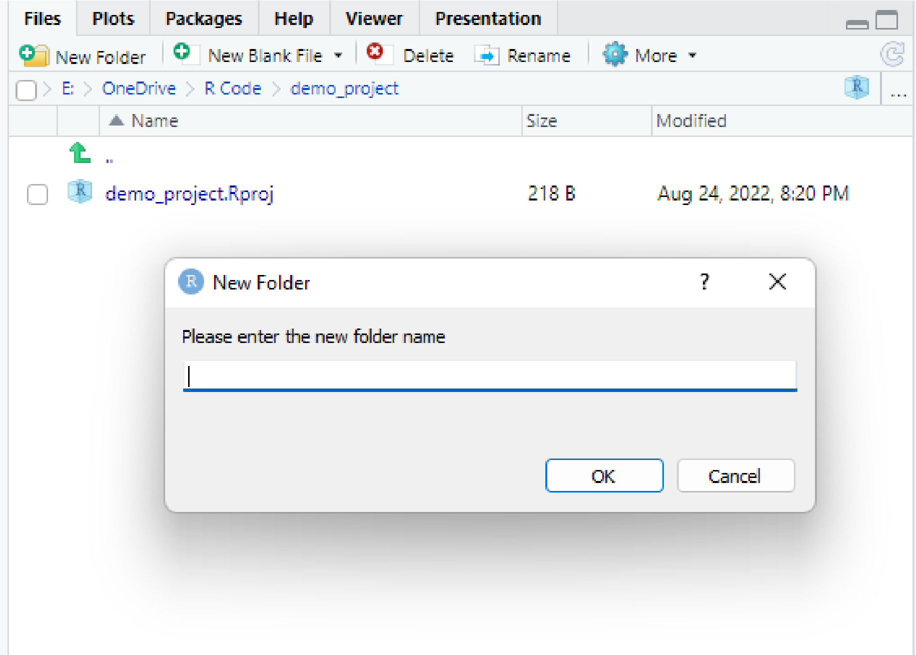
1. You should have already created a project.



E: > OneDrive > R Code > demo_project

Name	Size	Modified
..		
demo_project.Rproj	218 B	Aug 24, 2022, 8:20 PM

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

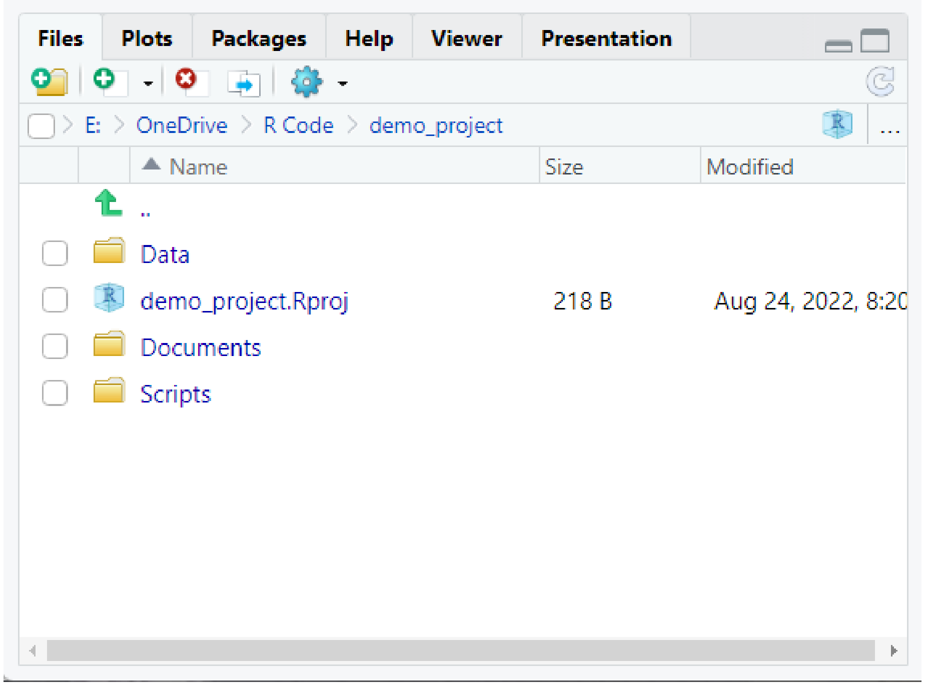


E: > OneDrive > R Code > demo_project

Name	Size	Modified
..		
demo_project.Rproj	218 B	Aug 24, 2022, 8:20 PM

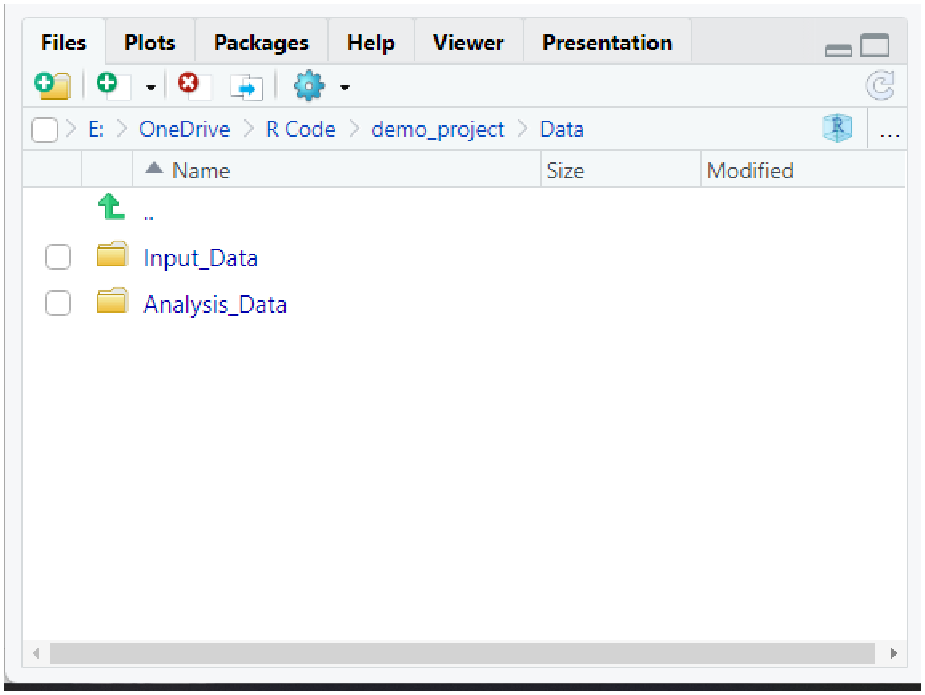
Please enter the new folder name

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

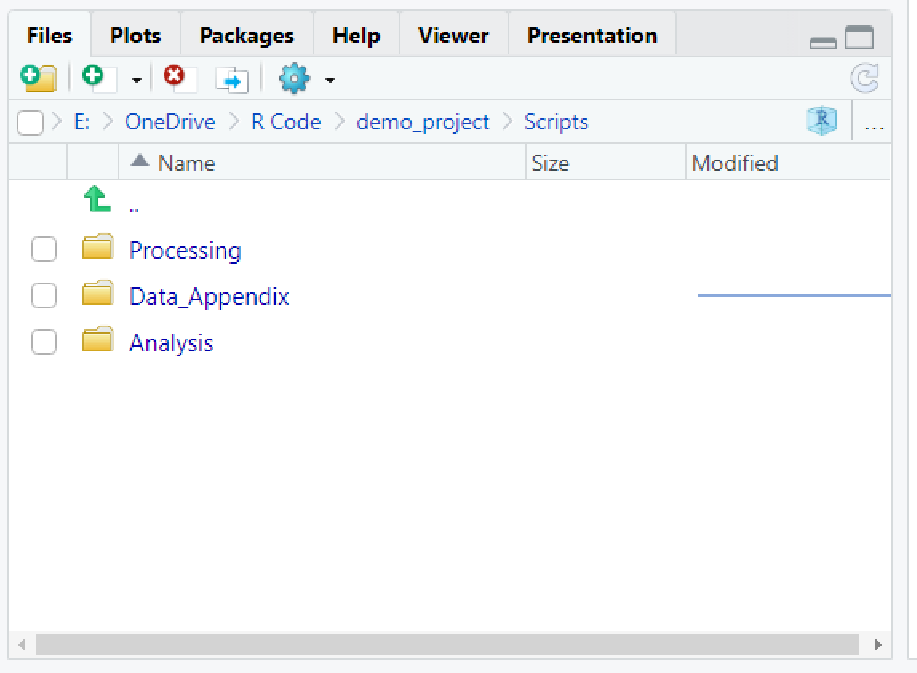


Name	Size	Modified
..		
Data		
demo_project.Rproj	218 B	Aug 24, 2022, 8:20
Documents		
Scripts		

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

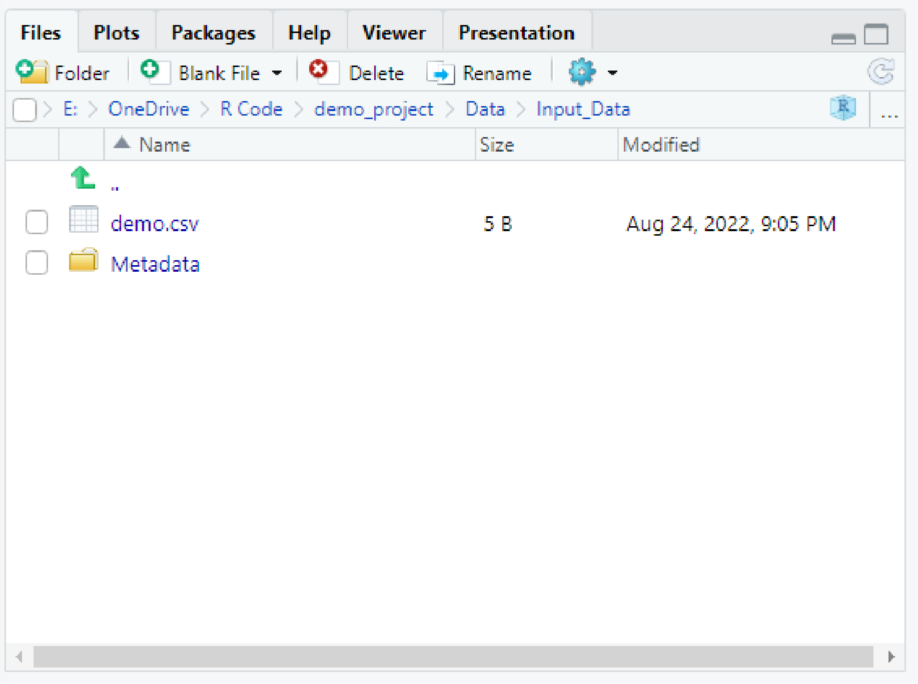
2.2.1 Problem

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.

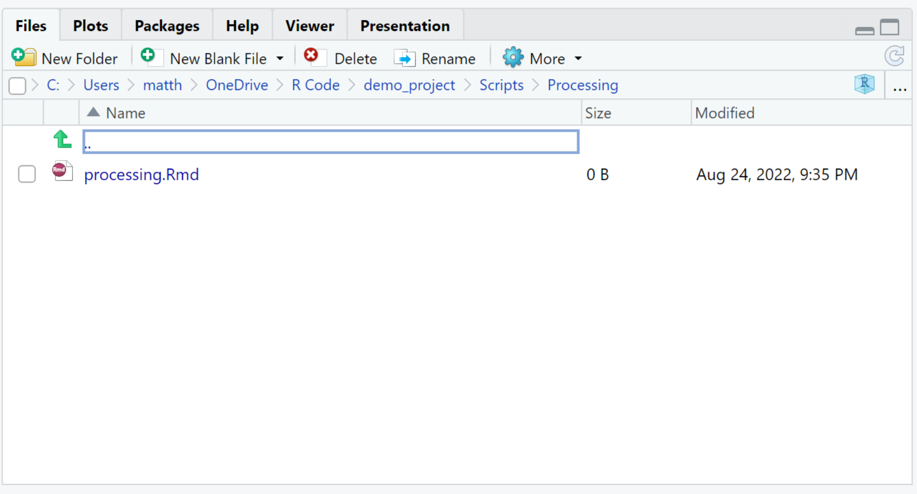


E: > OneDrive > R Code > demo_project > Data > Input_Data

Name	Size	Modified
..		
demo.csv	5 B	Aug 24, 2022, 9:05 PM
Metadata		

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

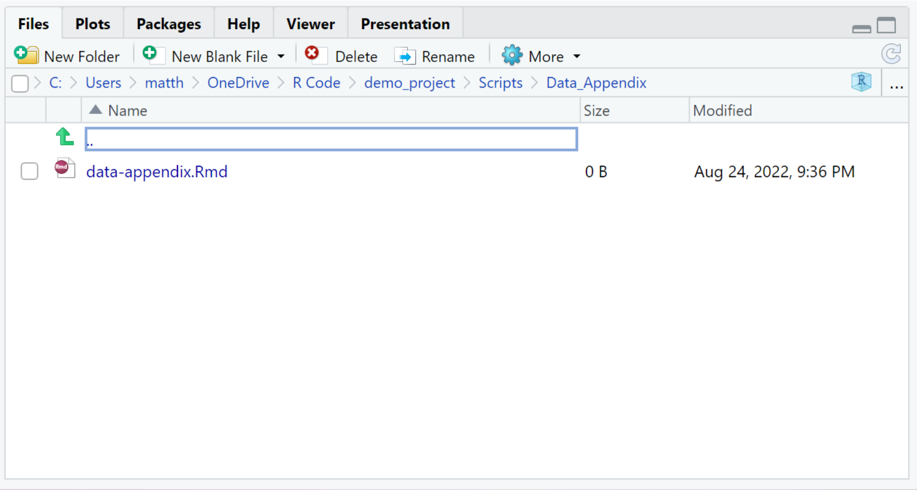
4. Inside of the “Scripts” folder there are three subfolders – “Processing”, “Analysis”, and “Data_Appendix”. The “Processing” folder should have one file named `processing.rmd`. This file is used to transform the input data into the form that you will use for the data analysis.



C: > Users > matth > OneDrive > R Code > demo_project > Scripts > Processing

Name	Size	Modified
..		
processing.Rmd	0 B	Aug 24, 2022, 9:35 PM

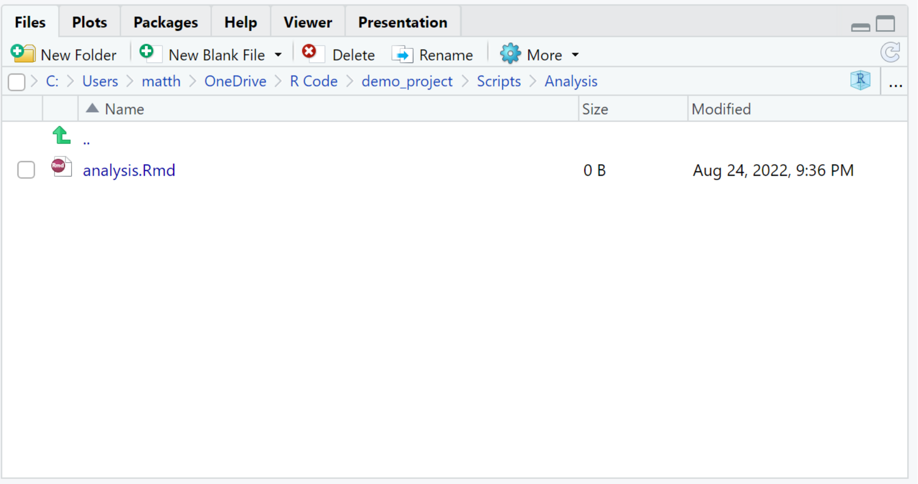
The “Data_Appendix” folder should have one filed named `data-appendix.rmd`. This file creates the Data Appendix, which serves as a codebook and provides descriptive statistics for the variables that are used in the actual data analysis.



C: > Users > matth > OneDrive > R Code > demo_project > Scripts > Data_Appendix

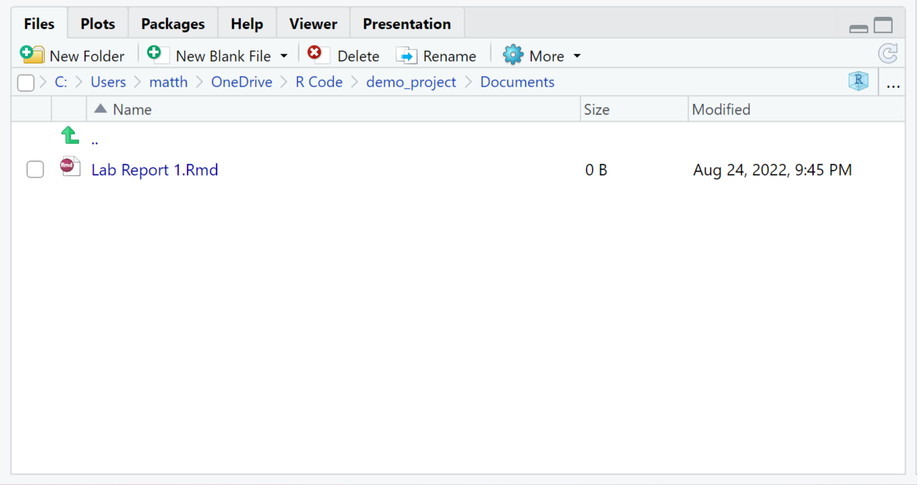
Name	Size	Modified
..		
data-appendix.Rmd	0 B	Aug 24, 2022, 9:36 PM

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.



Name	Size	Modified
..		
analysis.Rmd	0 B	Aug 24, 2022, 9:36 PM

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.



Name	Size	Modified
..		
Lab Report 1.Rmd	0 B	Aug 24, 2022, 9:45 PM

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.dem` to a new variable called `dem_therm`. The dataset is named “nes”.

```
nes$dem_therm <- nes$ft.dem
```

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.dem` to a new variable called `dem_therm`. The dataset is named “nes”.

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

  # use mutate() to assign values to the new variable
  mutate(dem_therm = ft.dem)
```

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 `ft.obama`, `ft.dem`, and `ft.biden.pre`.

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

  # use mutate() to add the three variables and divide to get the average
  mutate(avg_demfeel = (ft.obama + ft.dem + ft.biden.pre)/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 `battleground2020` in the `states` from a numeric dummy variable into a factor.

```
# the frequency table for the old variable
freqC(states$battleground2020, plot = F)
```

```
## ===========================================================================
##           Describing Distribution of Values with Frequency Table
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-44}Frequency Distribution of states$battleground2020
##
##          Frequency   Percent
## ------  ----------  --------
## 0                  37     74.00
## 1                  13     26.00
## Total              50    100.00
```

```
# call the new variable `battlefact`
states$battlefact <- as.factor(states$battleground2020)

# assign the levels
levels(states$battlefact) <- c("not a battleground", "battleground")

# the frequency table for the new variable
freqC(states$battlefact, plot = F)
```

```
## ===========================================================================
##           Describing Distribution of Values with Frequency Table
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-44}Frequency Distribution of states$battlefact
##
##                         Frequency   Percent
## -------------------  ----------  --------
## not a battleground           37     74.00
## battleground                 13     26.00
## Total                        50    100.00
```

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 `partyid7`
freqC(nes$partyid7, plot = F)
```

```
## ================================================================================
##            Describing Distribution of Values with Frequency Table
## ================================================================================
##
##
## Table: \label{tab:unnamed-chunk-46}Frequency Distribution of nes$partyid7 (PRE: SUM
##
##                                   Frequency   Percent   Cum.Percent
## ------------------------------  ----------  --------  ------------
## 1. Strong Democrat                    1961     23.78         23.78
## 2. Not very strong Democrat            900     10.92         34.70
## 3. Independent-Democrat                975     11.83         46.53
## 4. Independent                         968     11.74         58.27
## 5. Independent-Republican              879     10.66         68.93
## 6. Not very strong Republican          832     10.09         79.02
## 7. Strong Republican                  1730     20.98        100.00
## Total                                 8245    100.00            NA
```

```
# make `partyid7` numeric
nes$pid7 <- as.numeric(nes$partyid7)

# frequency table of the new variable `pid7`
freqC(nes$pid7, plot = F)
```

```
## ================================================================================
##            Describing Distribution of Values with Frequency Table
## ================================================================================
##
##
## Table: \label{tab:unnamed-chunk-46}Frequency Distribution of nes$pid7
##
##          Frequency   Percent
## ------  ----------  --------
## 1             1961     23.78
## 2              900     10.92
## 3              975     11.83
## 4              968     11.74
## 5               879     10.66
## 6               832     10.09
## 7              1730     20.98
## Total          8245    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
freqC(nes$abort4, plot = F)
```

```
## ===========================================================================
##           Describing Distribution of Values with Frequency Table
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-48}Frequency Distribution of nes$abort4
##
##                                 Frequency   Percent
## -----------------------------  ----------  --------
## never                                 870     11.01
## rape, incest, life of mother         1916     24.25
## yes with limits                      1097     13.89
## always                               4017     50.85
## Total                                7900    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
freqC(nes$abort4, plot = F)
```

```
## ===========================================================================
##           Describing Distribution of Values with Frequency Table
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-48}Frequency Distribution of nes$abort4
##
##                     Frequency   Percent   Cum.Percent
## ----------------  ----------  --------  ------------
## never                    870     11.01         11.01
## some conditions         1916     24.25         35.27
## more conditions         1097     13.89         49.15
## always                  4017     50.85        100.00
## Total                   7900    100.00            NA
```

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 an Indicator/Dummy Variable

An indicator variable (also known as a dummy variable) is a conversion of an existing variable such that it takes the value of 1 when the existing variable meets some criteria and the value of 0 otherwise. For example, if we want to plot the proportion of observations who voted for Joe Biden in 2020, then we would first make an indicator variable out of the nominal variable `presvote2020`.

3.7.1 Problem

You want to create an indicator/dummy variable.

3.7.2 Solution

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. Create a name for the indicator variable.
5. Set the indicator variable as being equal to a numeric version of whether the old variable satisfies some criteria.

```
# assign the dataset to itself
dataset <- dataset |>

  # use mutate() to create the indicator variable
  mutate(indicator_variable = as.numeric(some_criteria))
```

Here we create an indicator variable called `biden_vote` that takes a value of 1 for all observations in which `presvote2020` was equal to Joe Biden.

```
# assign the dataset to itself
nes <- nes |>

  #use mutate() to create the indicator variable
  mutate(biden_vote = as.numeric(presvote2020 == "1. Joe Biden"))

# check the new variable
freqC(nes$biden_vote, plot = F)
```

```
## ================================================================================
##           Describing Distribution of Values with Frequency Table
## ================================================================================
##
##
## Table: \label{tab:unnamed-chunk-50}Frequency Distribution of nes$biden_vote
##
##          Frequency   Percent
## ------  ----------  --------
## 0             2740     43.65
## 1             3537     56.35
## Total         6277    100.00
```

3.7.3 Troubleshooting

- The criteria is written as variable, logical operator, and value of the variable.

- The values of categorical variables must be in quotation marks.
- The most common error is that the value of the old variable is not written correctly. If the above example did not include the "1." in front of "Joe Biden", then the indicator variable would have returned values of 0 for all of the observations.

3.8 Filter your Data

3.8.1 Problem

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

3.8.2 Solution

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:

- `==` equal to
- `>` greater than
- `<` less than
- `>=` greater than or equal to
- `<=` less than or equal to
- `!=` not equal to
- `|` or, if there are multiple criteria
- `&` and, if there are multiple criteria

1. Follow the template below.

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

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

```
# frequency of `biden_vote` in full dataset
freqC(nes$biden_vote, plot = F)
```

```
## ===========================================================================
##           Describing Distribution of Values with Frequency Table
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-52}Frequency Distribution of nes$biden_vote
##
##          Frequency   Percent
## ------  ----------  --------
## 0             2740     43.65
## 1             3537     56.35
## Total         6277    100.00
```

```
# create a subset of `nes` data that only includes Biden voters
bidenites <- nes %>%
  filter(biden_vote == 1)

# frequency of a `biden_vote` in subset
freqC(bidenites$biden_vote, plot = F)
```

```
## ============================================================================
##          Describing Distribution of Values with Frequency Table
## ============================================================================
##
##
## Table: \label{tab:unnamed-chunk-52}Frequency Distribution of bidenites$biden_vote
##
##          Frequency   Percent
## ------  ----------  --------
## 1             3537    100.00
## Total         3537    100.00
```

3.8.3 Troubleshooting

- The criteria is written as variable, logical operator, and value of the variable.
- The values of categorical variables must be in quotation marks.

3.9 Summarize Data Using Means

3.9.1 Problem

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

3.9.2 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, `ft.obama`, by party identification, `partyid7`.

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

  # use group_by() to group the data by partyid7
  group_by(partyid7) %>%

  # calculate the means of ft.obama
  summarise(average = wtd.mean(ft.obama, na.rm = T,
                               w = wt))

# show the summary dataset
partymeans
```

```
## # A tibble: 8 x 2
##   partyid7                      average
##   <ord>                           <dbl>
## 1 1. Strong Democrat               93.3
## 2 2. Not very strong Democrat      83.0
## 3 3. Independent-Democrat          82.0
## 4 4. Independent                   63.0
## 5 5. Independent-Republican        37.7
## 6 6. Not very strong Republican    44.2
## 7 7. Strong Republican             18.4
## 8 <NA>                             77.9
```

3.9.3 Troubleshooting

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

3.10 Transform an interval variable into a nominal variable

Many of the tools we use to analyze the relationship between two variables require that the independent variable is nominal or ordinal. In those cases, it can be useful to transform an interval independent variable into a nominal or ordinal version.

3.10.1 Problem

You want to transform an interval variable into a nominal variable.

3.10.2 Solution

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. Create a name for the new nominal variable.
5. Set the nominal variable as being equal to a `factor()` of whether the old variable is greater than some value. The choice of value is up to the researcher, but it should make sense as a way to divide the original variable between "high" and "low".
6. Place a comma at the end of your criteria, and press return.
7. Use the argument `labels =` to assign levels to the values of the variable.
8. The levels should be provided as a list in `c()` with the names of the levels in quotation marks.

```r
# assign the dataset to itself
dataset <- dataset |>

  # use mutate to create the nominal variable
  mutate(new_nominal = factor(

    # the criteria for determining high vs. low values
    old_interval > some_value,

    # assign the levels to the new variable
    labels = c("low", "high")))
```

In the example below we transform the Obama feeling thermometer, `ft.obama`, into a nominal variable, `high_obama`. The interval variable `ft.obama` ranges from 0 to 100, so we set 60 and above as the value for "high".

```r
nes <- nes |>
  mutate(high_obama = factor(ft.obama > 60,
                             labels = c("cool Obama feelings",
                                        "warm Obama feelings")))
freqC(nes$high_obama, plot = F)
```

```
## ================================================================================
##              Describing Distribution of Values with Frequency Table
## ================================================================================
##
##
## Table: \label{tab:unnamed-chunk-56}Frequency Distribution of nes$high_obama
##
```

```
##                        Frequency   Percent
## --------------------  ----------  --------
## cool Obama feelings         3653     44.74
## warm Obama feelings         4512     55.26
## Total                       8165    100.00
```

3.10.3 Troubleshooting

- Make sure that you have the correct number of closing parentheses. Following the template above, there should be three parentheses at the end.
- Remember that we are converting an interval variable, so the value of that variable should be a number without quotation marks.

3.11 Transform an interval variable into an ordinal variable

Many of the tools we use to analyze the relationship between two variables require that the independent variable is ordinal. In those cases, it can be useful to transform an interval independent variable into an ordinal version.

3.11.1 Problem

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

3.11.2 Solution

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. Create a name for the new ordinal variable.
5. Set the ordinal variable as being equal to a factor
6. Inside the `factor()` function, use the function `transformC()`. This function will take four arguments.Each argument is separated by a comma.
7. The first argument for `transformC()` is set type equal to 'cut'.
8. The second argument is set x equal to the name of interval variable.
9. The third argument is set cutpoints equal to a list of the values you want to use to split the interval variable into ordinal categories.
10. Alternatively, the third argument is set groups equal to the number of ordinal categories you want.
11. The fourth argument is set confirm equal to F.
12. Close the parentheses to indicate that this is the end of the `transformC()` function, type a comma, and press return.
13. Use the argument `labels =` to assign levels to the values of the variable.
14. The levels should be provided as a list in `c()` with the names of the levels in quotation marks. Some variation of "low", "medium", "high" would

make sense for an ordinal variable with three categories.

```
# assign dataset to itself
dataset <- dataset |>

  # use `mutate()` to create a new variable as a factor
  mutate(new_ordinal = factor(

    # Use transformC() to create the ordinal variable
    transformC(type = 'cut',

               # second argument x = interval_variable
               x = old_interval,

               # third argument set cutpoints
               cutpoints = c(value1, value2),

               # fourth argument confirm = F
               confirm = F),

    # assign levels
    labels = c("low", "medium", "high")))
```

In the example below we transform the Obama feeling thermometer, `ft.obama`, into a ordinal variable, `obama_ord`. The interval variable `ft.obama` ranges from 0 to 100 and is supposed to act as thermometer, so we set the cutpoints at 40 and 60. Values below 40 are "cold", values between 40 and 60 are "mid", and values above 60 are "warm".

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

  # use `mutate()` to create a new variable as a factor
  mutate(obama_ord = factor(

    # Use transformC() to create the ordinal variable
    transformC(type = 'cut',

               # second argument x = interval_variable
               x = ft.obama,

               # third argument set cutpoints
               cutpoints = c(40, 60),

               # fourth argument confirm = F
               confirm = F),
    # assign levels
    labels = c("cold", "mid", "warm")))

# check our work
freqC(nes$obama_ord, plot = F)
```

```
## ============================================================================
##           Describing Distribution of Values with Frequency Table
## ============================================================================
##
##
## Table: \label{tab:unnamed-chunk-58}Frequency Distribution of nes$obama_ord
##
##          Frequency   Percent   Cum.Percent
## ------  ----------  --------  ------------
## cold          2290     28.05         28.05
## mid            891     10.91         38.96
## warm          4984     61.04        100.00
## Total         8165    100.00            NA
```

3.11.3 Troubleshooting

- One reason to use cutpoints instead of groups is that it is not clear how to discover where the cutpoints are are when you only use groups. With the 'groups' argument, `transformC()` will try to make the categories have roughly equivalent numbers of observations, and that may not always make sense for the actual data.

- Be careful about syntax errors like missing commas and parentheses. This code uses functions inside of functions, so make sure that you have the right parentheses in the right places.

3.12 Pivot a Dataframe Wide

Data that we collect out in the wild is not always organized around our chosen unit of analysis. Look at the data below:

```
knitr::kable(urban)
```

city	type	value
Atlanta	voting	43
Atlanta	protests	12
Dover	voting	78
Dover	protests	29
Rochester	voting	37
Rochester	protests	52

Here, each row represents a city and a type of political participation. This is not what we want. We want the unit of analysis to be cities, so each row should just represent one city, with different columns for the different types of participation. Pivoting data is one way to reorganize a dataset so that the rows of the data match the unit of analysis. A "wide" pivot is when we turn the values of a variable into columns of the dataset.

3.12.1 Problem

You want to do a "wide" pivot that reorganizes a dataset by making separate columns for the values of a variable.

3.12.2 Solution

1. Create a new data object.
2. Assign the old dataset to the new object.
3. Use the pipe operator `|>` at the end of that line of code.
4. Use the `pivot_wider()` function, specifying at least two arguments.
5. The first argument, `names_from`, is to provide the name of the variable whose values are being converted into columns.
6. The second argument, `values_from`, provides the values that will be used to populate the new columns.

```
# assign the old dataset to a new object
new_data <- old_data |>

  # use pivot_wider()
  pivot_wider(

    # use `names_from` to specify the variable that is being turned into columns
    names_from = some_variable,

    # use `values_from` to specify the variable that provides values for the columns
    values_from = some_other_variable
  )
```

In the example below we perform a wide pivot on the `urban` dataset to make cities into the unit of analysis. Specifically, we will create separate columns for each type of political participation – voting and protests.

```
# assign the old dataset to a new object
city_part <- urban |>

  # use pivot_wider()
  pivot_wider(

    # use `names_from` to specify the variable that is being turned into columns
    names_from = type,

    # use `values_from` to specify the variable that provides values for the columns
    values_from = value
  )

# look at the data
knitr::kable(city_part)
```

city	voting	protests
Atlanta	43	12
Dover	78	29
Rochester	37	52

3.12.3 Troubleshooting

- In more complicated datasets, it will probably be necessary to use the `id_cols` argument in `pivot_wider()` to specify the variable that identifies all of the unique observations in a dataset.

3.13 Pivot a Dataframe Long

Data that we collect out in the wild is not always organized around our chosen unit of analysis. Look at the data below:

```
knitr::kable(urban2)
```

city	1981	1999	2003
Atlanta	43	45	47
Dover	78	81	84
Rochester	37	47	57

Here we have the voter turnout data for three cities in 1981, 1999, and 2003. The unit of analysis cities. In this case, we are interested in how cities perform over time, so we want the unit of analysis to be city-year instead of cities. Pivoting data is one way to reorganize a dataset so that the rows of the data match the unit of analysis. A “long” pivot is when we collapse the columns of a dataset into the values of a variable.

3.13.1 Problem

You want to do a “long” pivot that reorganizes your data such that a number of columns are collapsed into the values of two variables.

3.13.2 Solution

1. Create a new data object.

2. Assign the old dataset to the new object.
3. Use the pipe operator `|>` at the end of that line of code.
4. Use the `pivot_longer()` function, specifying at least three arguments.
5. The first argument, `cols`, is to provide the name of the columns that are being collapsed.
6. The second argument, `names_to`, provides the new name of the variable that will whose values will now be the names of the collapsed columns
7. The third argument, `values_to`, provides the name of the new variable that inherits the values from the collapsed columns.

```r
# assign the old dataset to a new object
new_data <- old_data |>

  # use pivot_longer()
  pivot_longer(

    # use `cols` to specify the variables that are being collapsed
    cols = first_column:last_column,

    # use `names_to` to specify the new variable that holds the column names
    names_to = variable_name,

    # use `values_to` to specify the variable that holds the new values
    values_to = other_variable
  )
```

In the example below we perform a long pivot on the `urban2` dataset to make city-year into the unit of analysis.Specifically, we will collapse the variable `1981`, `1999`, and `2003` into one new variable called `year`. The values will be placed in a variable called `turnout`.

```r
# assign the old dataset to a new object
city_year <- urban2 |>

  # use pivot_longer()
  pivot_longer(

    # use `cols` to specify the variables that are being collapsed
    cols = `1981`:`2003`,

    # use `names_to` to specify the new variable that holds the column names
    names_to = "year",

    # use `values_to` to specify the variable that holds the new values
    values_to = "turnout"
  )
```

```
# look at the data
knitr::kable(city_year)
```

city	year	turnout
Atlanta	1981	43
Atlanta	1999	45
Atlanta	2003	47
Dover	1981	78
Dover	1999	81
Dover	2003	84
Rochester	1981	37
Rochester	1999	47
Rochester	2003	57

3.13.3 Troubleshooting

- The names of the columns can also be provided as a list using `c()`.

3.14 Creating a 'Key' Variable

It is often helpful to have a variable that uniquely identifies each of your observations. This kind of 'id' or 'key' variable is particularly useful when pivoting or merging datasets.

3.14.1 Problem

You want to create a 'key' variable that uniquely identifies all of your observations.

3.14.2 Solution

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. Create a name for the new variable. Something simple like `key` or `id` works.
5. Set the new variable equal to the function `paste()`.
6. Inside the `paste()` list the variables that will be used to create the unique identifier. Each variable should be separated by a comma.
7. Add a comma after the last variable listed and then press return.
8. Set the argument `sep` equal to "-".

```
# assign dataset to itself
dataset <- dataset |>

  # use mutate to create `key` set to `paste()`
```

```r
  mutate(key = paste(

    # list the variables that uniquely identify observations
    first_variable,
    second_variable,

    # set a value for the `sep` argument
    sep = "-"
  ))
```

Look at the `city_year` data we created in the long pivot example.

```r
knitr::kable(city_year)
```

city	year	turnout
Atlanta	1981	43
Atlanta	1999	45
Atlanta	2003	47
Dover	1981	78
Dover	1999	81
Dover	2003	84
Rochester	1981	37
Rochester	1999	47
Rochester	2003	57

We can see that the combination of `city` and `year` are what would create a unique identifier for each observation in the dataset. The code below creates that key variable.

```r
# assign dataset to itself
city_year <- city_year |>

  # use mutate to create `key` set to `paste()`
  mutate(key = paste(

    # list the variables that uniquely identify observations
    city,
    year,

    # set a value for the `sep` argument
    sep = "-"
  ))

# check our work -- the number of unique values in `key` should equal the
# total number of observations
length(unique(city_year$key)) == nrow(city_year)
```

```
## [1] TRUE
```

3.14.3 Troubleshooting

- There are times when the length of a ‘key’ variable will not be equal to the total number of observations. For example, this will not be true for data that requires a wide pivot. In those circumstances you may want to create the key variable first, and then use the key variable for the `id_cols` argument in `pivot_wider()`.
- There is nothing special about using the dash, “-”, as the separator. You can substitute another symbol if you prefer. Just make sure that the symbol is in quotation marks.

3.15 Merge Multiple Datasets Together

Data collection is rarely perfect. We often find that one dataset has some good potential dependent variables and a different dataset has good potential independent variables. In these circumstances, we need to merge the data together.

3.15.1 Problem

You want to merge two datasets.

3.15.2 Solution

1. Identify the ‘key’ variable that identifies the unique observations in both datasets. That is, each dataset should have the same ‘key’ variable.
2. If a ‘key’ variable does not already exist, then create a key variable.
3. For the secondary dataset, use `select()` to include only the variables that you need to merge. Remember that the ‘key’ variable will be one of the variables that you select.
4. Create a new data object.
5. Assign the dataset that you want to keep all of the observations of to the new object.
6. Use the pipe operator `|>` at the end of that line of code.
7. Use the `left_join()` function to combine the datasets.
8. The first argument is the name of the dataset that you are merging with your new data object.
9. Set the `by` argument equal to the name of your ‘key’ variable in quotation marks.

```r
# Assign the dataset to a new data object
new_data <- old_data |>

  # Merge using the `left_join()` function
  left_join(other_data, by = 'key')
```

Below we will merge data on voter turnout for three cities across three years, `city_year`, with data on protest participation for three cities across three years, `urban3`. The 'key' to merge these data should be a combination of the name of the city and the year for the data.

```
# long pivot of `urban3`
# assign the old dataset to a new object
protest_year <- urban3 |>
  # use pivot_longer()
  pivot_longer(
    # use `cols` to specify the variables that are being collapsed
    cols = `1981`:`2003`,
    # use `names_to` to specify the new variable that holds the column names
    names_to = "year",
    # use `values_to` to specify the variable that holds the new values
    values_to = "protest"
  ) |>

  # create the 'key' variable
 mutate(key = paste(
    # list the variables that uniquely identify observations
    city,
    year,
    # set a value for the `sep` argument
    sep = "-"
  )) |>

  # select only the columns you want to merge
  select(key, protest)

# city-year and protest_year
# Assign the dataset to a new data object
participation <- city_year |>

  # Merge using the `left_join()` function
  left_join(protest_year, by = 'key') |>

  # relocate the 'key' variable to the front
  relocate(key)

# check our work
knitr::kable(participation)
```

key	city	year	turnout	protest
Atlanta-1981	Atlanta	1981	43	12
Atlanta-1999	Atlanta	1999	45	14
Atlanta-2003	Atlanta	2003	47	16
Dover-1981	Dover	1981	78	29
Dover-1999	Dover	1999	81	32
Dover-2003	Dover	2003	84	35
Rochester-1981	Rochester	1981	37	52
Rochester-1999	Rochester	1999	47	62
Rochester-2003	Rochester	2003	57	72

Chapter 4

Descriptive Statistics

We use descriptive statistics to learn about an individual variable. More specifically, we use central tendency and dispersion to describe our data.

4.1 Making Subsets of Data

4.1.1 Problem

You want to create a smaller version of your dataset - a subset - using some criteria based on your variables.

4.1.2 Solution

1. Decide on the variable(s) you want to base the subset on.
2. Create an object that you will assign the subset to
3. Use the `subset()` function.
4. The main arguments for the function are the data that is being subsetted, the criteria being used for the subset, and the columns of the original dataset that you want included in your subset.
5. The criteria for the subset are specified using logical operators:
    - `==` equal to
    - `>` greater than
    - `<` less than
    - `>=` greater than or equal to
    - `<=` less than or equal to
    - `!=` not equal to
    - `|` or, if there are multiple criteria
    - `&` and, if there are multiple criteria
6. Generically, we would write something like the following:

```r
# subset where variable is equal to some value
subset_name <- subset(dataset, variable == some value)

# subset where variable is greater than some value
subset_name <- subset(dataset, variable > some value)

# subset with multiple criteria
subset_name <- subset(dataset, variable < some value |
                        othervariable != some value)

# subset with selected columns
subset_name <- subset(dataset, variable <= some value,
                      select = c(variable1, variable2, variable3, variable))
```

In this example we will create a subset of the `nes` dataset based on the variable `black`.

```r
# making subset called 'black.nes' from 'nes'
black.nes <- subset(nes, black == "Yes")
```

4.1.3 Troubleshooting

- If the values of a variable are not numeric, then they will need to be in quotation marks: `variable == "value"`.
- Remember that the dataset has already been specified as an argument, so you should not use `data$variable` when writing the subset's criteria.
- This function requires the `RCPA3` package.

4.2 Making a Frequency Table

A frequency table shows the the number of observations that take on each value of a variable.

4.2.1 Problem

You want to create a frequency table of a variable.

4.2.2 Solution

1. Use `freqC()` to make the table.
2. Set the `data` argument to the name of your dataset.
3. Set the argument `x` to the name of the variable that you want to make a frequency table of.
4. If you have survey data, then set the `w` argument to the name of the weights variable.

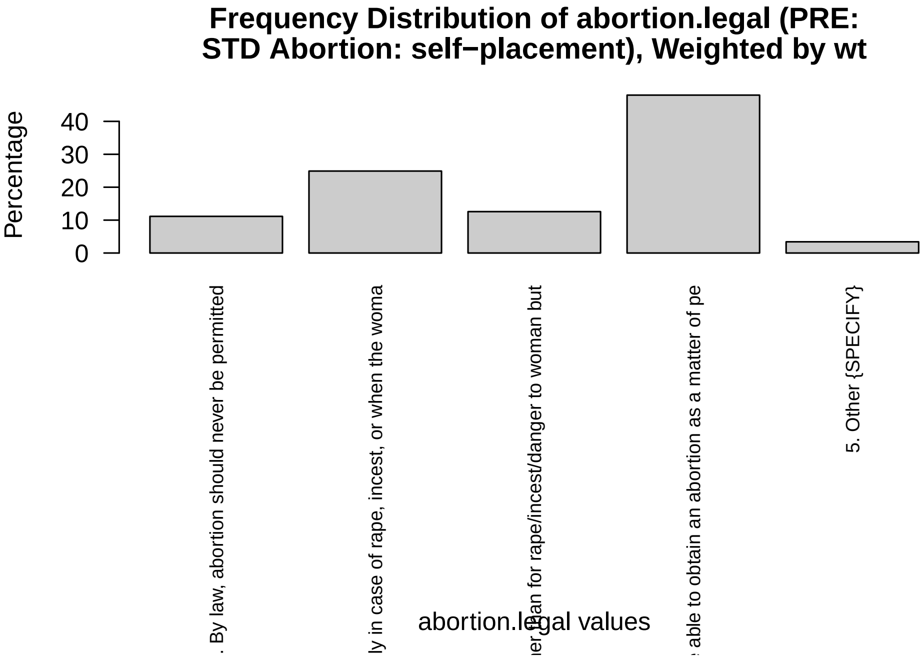
```
# Use `freqC()`
freqC(

  # set data to the name of the dataset
  data = dataset,

  # set x to the name of the variable
  x = variable_name,

  # set w to the weights (only for survey data)
  w = weights_name
)
```

In the example below we create a frequency table of the variable `abortion.legal` found in the `nes` dataset. The weights for this data are named `wt`.

```
# Use `freqC()`
freqC(

  # set data to the name of the dataset
  data = nes,

  # set x to the name of the variable
  x = abortion.legal,

  # set w to the weights (only for survey data)
  w = wt
)
```

```
## ===========================================================================
##    Describing Distribution of Values with Frequency Table and Bar Chart
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-72}Frequency Distribution of abortion.legal (PRE: STD Abortion
##
##                                                                                     Frequency
## --------------------------------------------------------------------------------  ----------
## 1. By law, abortion should never be permitted                                         915.64
## 2. The law should permit abortion only in case of rape, incest, or when the woma     2047.48
## 3. The law should permit abortion other than for rape/incest/danger to woman but     1034.15
## 4. By law, a woman should always be able to obtain an abortion as a matter of pe     3944.82
## 5. Other {SPECIFY}                                                                    280.10
## Total                                                                                8222.20
```

```
## One or more text labels for bars is long. Consider using rowlabs argument to abbreviate.
```



4.2.3 Troubleshooting

- Alternatively, you can ignore the `data` argument and specify `x` and `w` using the `data$variable` notation.
- Remember to use the weights for survey data. The frequency table will be incorrect otherwise.
- There is also a function called `freq()` that uses different arguments. It is easy to inadvertently use that function. Make sure that you are using `freqC()`.
- This function requires the `RCPA3` package.

4.3 Calculate the Mode

The mode tells us the value of the variable that has the largest number of observations. We could glean this information from a frequency table, but it is more straightforward to have R calculate the mode directly.

4.3.1 Problem

You want to calculate the mode of a variable.

4.3.2 Solution

1. Use the function `wtd.mode()`.

2. Set the `data` argument to the name of your dataset.
3. Set the argument `x` to the name of the variable for which a mode is being calculated.
4. If you have survey data, then set the `w` argument to the name of the weights variable.

```
# Use `wtd.mode()`
wtd.mode(

  # set data to the name of the dataset
  data = dataset,

  # set x to the name of the variable
  x = variable_name,

  # set w to the weights (only for survey data)
  w = weights_name
)
```

In the example below, we calculate the mode of `abortion.legal` in the `nes` dataset. The name of the weight is `wt`.

```
# Use `wtd.mode()`
wtd.mode(

  # set data to the name of the dataset
  data = nes,

  # set x to the name of the variable
  x = abortion.legal,

  # set w to the weights (only for survey data)
  w = wt
)
```

```
## [1] "4. By law, a woman should always be able to obtain an abortion as a matter of pe"
```

4.3.3 Troubleshooting

- Alternatively, you can ignore the `data` argument and specify `x` and `w` using the `data$variable` notation.
- Remember to use the weights for survey data. The mode could be incorrect otherwise.
- This function requires the `RCPA3` package.

4.4 Calculate the Median

The median tells us the value of the variable that splits the data in half.

4.4.1 Problem

You want to calculate the median of a variable.

4.4.2 Solution

1. Use the function `wtd.median()`.
2. Set the `data` argument to the name of your dataset.
3. Set the argument `x` to the name of the variable for which a mode is being calculated.
4. If you have survey data, then set the `w` argument to the name of the weights variable.

```
# Use `wtd.median()`
wtd.median(

  # set data to the name of the dataset
  data = dataset,

  # set x to the name of the variable
  x = variable_name,

  # set w to the weights (only for survey data)
  w = weights_name
)
```

In the example below, we calculate the median of `abortion.legal` in the `nes` dataset. The name of the weight is `wt`.

```
# Use `wtd.median()`
wtd.median(

  # set data to the name of the dataset
  data = nes,

  # set x to the name of the variable
  x = abortion.legal,

  # set w to the weights (only for survey data)
  w = wt
)
```

```
## [1] "4. By law, a woman should always be able to obtain an abortion as a matter of
```

4.4.3 Troubleshooting

- Alternatively, you can ignore the `data` argument and specify `x` and `w` using the `data$variable` notation.
- Remember to use the weights for survey data. The mode could be incorrect otherwise.
- You can only calculate the median if the variable is ordinal or interval.
- This function requires the `RCPA3` package.

4.5 Calculate the Mean

The mean is total sum of values of a variable divided by the number of observations. This is what we typically think of when we use the term 'average'.

4.5.1 Problem

You want to calculate the mean of a variable.

4.5.2 Solution

1. Use the function `wtd.mean()`.
2. Set the `data` argument to the name of your dataset.
3. Set the argument `x` to the name of the variable for which a mode is being calculated.
4. If you have survey data, then set the `w` argument to the name of the weights variable.

```
# Use `wtd.mean()`
wtd.mean(

  # set data to the name of the dataset
  data = dataset,

  # set x to the name of the variable
  x = variable_name,

  # set w to the weights (only for survey data)
  w = weights_name
)
```

In the example below, we calculate the mean of `ft.biden.post` in the `nes` dataset. The name of the weight is `wt`.

```
# Use `wtd.mean()`
wtd.mean(

  # set data to the name of the dataset
  data = nes,

  # set x to the name of the variable
  x = ft.biden.post,

  # set w to the weights (only for survey data)
  w = wt
)
```

```
## [1] 52.307
```

4.5.3 Troubleshooting

- Alternatively, you can ignore the `data` argument and specify `x` and `w` using the `data$variable` notation.
- Remember to use the weights for survey data. The mode could be incorrect otherwise.
- You can only calculate the mean if the variable is interval.
- This function requires the `RCPA3` package.

4.6 Describe a Variable

4.6.1 Problem

You want a general description of a variable.

4.6.2 Solution

1. Use the function `describeC()`.
2. Set the `data` argument to the name of your dataset.
3. Set the argument `x` to the name of the variable for which a mode is being calculated.
4. If you have survey data, then set the `w` argument to the name of the weights variable.

```
# Use `describeC()`
describeC(

  # set data to the name of the dataset
  data = dataset,

  # set x to the name of the variable
  x = variable_name,

  # set w to the weights (only for survey data)
  w = weights_name
)
```

In the example below, we provide a description of `ft.biden.post` in the `nes` dataset. The name of the weight is `wt`.

```
# Use `describeC()`
describeC(

  # set data to the name of the dataset
  data = nes,

  # set x to the name of the variable
  x = ft.biden.post,

  # set w to the weights (only for survey data)
  w = wt
)
```

```
## ===========================================================================
##                            Descriptive Statistics
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-80}Descriptive Statistics for ft.biden.post, weighted by wt
##
##                                 ft.biden.post
## --------------------------  --------------
## Observed values                   7256.644
## Missing values                    1023.356
## Unique values                           64
## Class                              numeric
## Mean                                52.307
## Median                                  60
## Mode                                     0
## Variance                           1268.51
## Standard deviation                  35.616
## Minimum                                  0
## Maximum                                100
## Range                                  100
## First quartile (25%)                    15
## Third quartile (75%)                    85
## Interquartile range (IQR)               70
## Skewness                            -0.222
## Kurtosis                             1.553
```

4.6.3 Troubleshooting

- The function works with all levels of measurement, but it is most useful for interval variables.
- Alternatively, you can ignore the `data` argument and specify `x` and `w` using the `data$variable` notation.
- Remember to use the weights for survey data. The mode could be incorrect otherwise.
- This function requires the `RCPA3` package.

4.7 Calculate the Range

The range of a variable tells you the lowest and highest value. This can only be calculated for interval variables.

4.7.1 Problem

You want to calculate the range of an interval variable.

4.7.2 Solution

1. Use the `range()` function.
2. The argument is the name of the variable that you want the range of using the format `data$variable`.
3. Set the argument `na.rm` equal to True.

```r
# Use the range() function
range(

  # enter the name of the dataset and the name of the variable
  dataset$variable_name,

  # set na.rm=T
  na.rm = T
)
```

In the example below, we find the range of the variable `ft.biden.post`. The dataset is called `nes`.

```r
# Use the range() function
range(

  # enter the name of the dataset and the name of the variable
  nes$ft.biden.post,

  # set na.rm=T
  na.rm = T
)
```

```
## [1]   0 100
```

4.7.3 Troubleshooting

- No common mistakes identified.

4.8 Make a Bar Chart of One Variable for non-Survey Data

We can use a bar chart to visualize the dispersion for nominal or ordinal variables.

4.8.1 Problem

You want to make a bar chart to visualize the dispersion of one variable when we do not have survey data.

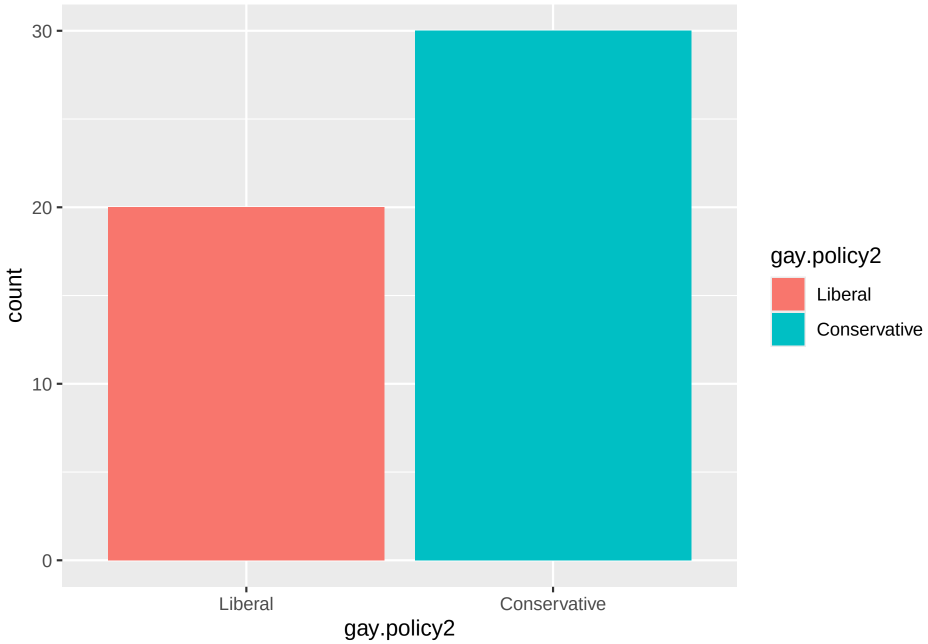
4.8.2 Solution

1. Create a plot object. I typically name my plots something simple like `p1`.
2. Assign the function `ggplot()` to the plot object.
3. Set the argument `data` to the name of your dataset.
4. Set the argument `mapping` to the function `aes()`.
    a. Inside the `aes()` function, set the argument `x` to the name of your variable.
    b. Set the argument `fill` to the name of your variable.
5. Make sure all the parentheses are closed and type a `+` sign.
6. Use the function `geom_bar()` to draw the bar. The function will not have any arguments.
7. Call the plot object by typing its name.

```
# assign ggplot() to a new plot object
p1 <- ggplot(

  # set data to the name of the dataset
  data = dataset,

  # set `mapping` to the function `aes()`
  mapping = aes(

    # set `x` to the name of the variable
    x = variable_name,
    # set `fill` to the name of the variable
    fill = variable_name
  )

  # close parentheses and type a + sign
) +

  # use the function `geom_bar()`
  geom_bar()

# call the plot
p1
```

In the example below, we create a bar chart for the variable `gay.policy2`. The name of the dataset is `states`.

```
# assign ggplot() to a new plot object
p1 <- ggplot(

  # set data to the name of the dataset
  data = states,

  # set `mapping` to the function `aes()`
  mapping = aes(

    # set `x` to the name of the variable
    x = gay.policy2,

    # set `fill` to the name of the variable
    fill = gay.policy2
  )

  # close parentheses and type a + sign
) +

  # use the function `geom_bar()`
  geom_bar()

# call the plot
p1
```



4.8.3 Troubleshooting

- Make sure that all arguments are separated by a comma, and that you use a `+` sign to connect the `ggplot()` function to the `geom_bar()` function.
- Remember to call the plot. If you do not type the name of the plot object (`p1` in the example above), then R will not show what the plot looks like.

4.9 Make a Histogram (for non survey data)

We use histograms to visualize the dispersion of an interval variable.

4.9.1 Problem

You want to make a histogram, and you do not have survey data.

4.9.2 Solution

1. Assign the function `ggplot()` to a new plot object. A simple name like `p1` is fine.
2. Set the argument `data` to the name of your dataset.
3. Set the argument `mapping` to the function `aes()`.
4. Inside the `aes()` function, set the argument `x` to the name of your variable.
5. Make sure all the parentheses are closed and type a `+` sign after the last parenthesis.

6. Use the function `geom_histogram()` to draw the histogram bars. The function will not have any arguments.
7. Call the plot object by typing its name.

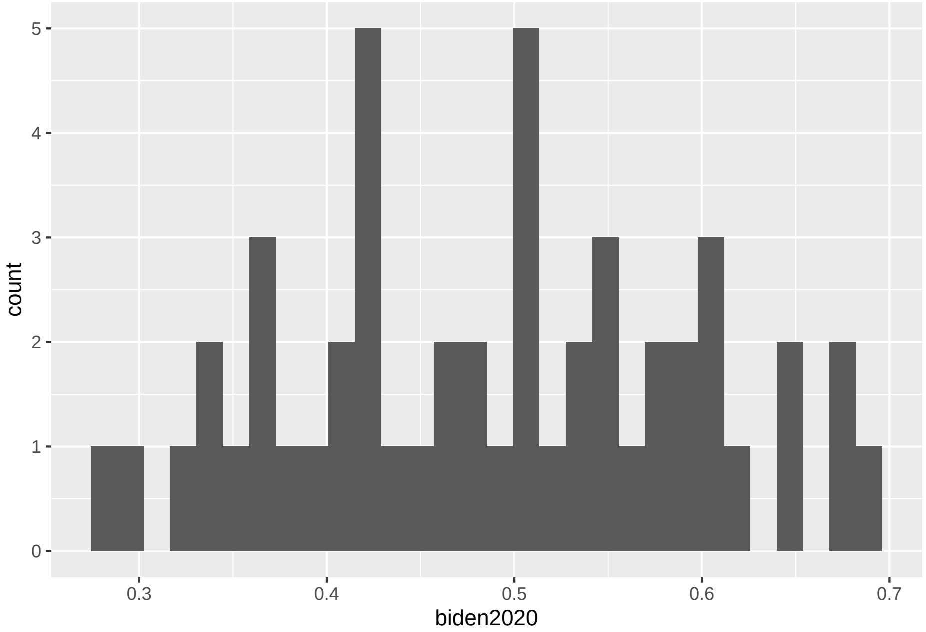
```
# assign ggplot() to a new plot object
p2 <- ggplot(

  # set data to the name of the dataset
  data = dataset,

  # set `mapping` to the function `aes()`
  mapping = aes(

    # set `x` to the name of the variable
    x = variable_name
  )

  # close parentheses and type a + sign
) +

  # use the function `geom_bar()`
  geom_histogram()

# call the plot
p2
```

In the example below we create a histogram of Biden's share of the vote in 2020, `biden2020`, which is found in the `states` dataset.

```
# assign ggplot() to a new plot object
p2 <- ggplot(

  # set data to the name of the dataset
  data = states,

  # set `mapping` to the function `aes()`
  mapping = aes(

    # set `x` to the name of the variable
    x = biden2020
  )

  # close parentheses and type a + sign
) +

  # use the function `geom_bar()`
  geom_histogram()
```

```
# call the plot
p2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



4.9.3 Troubleshooting

- Make sure that all arguments are separated by a comma, and that you use a `+` sign to connect the `ggplot()` function to the `geom_bar()` function.
- Remember to call the plot. If you do not type the name of the plot object (`p1` in the example above), then R will not show what the plot looks like.
- You can use the argument `binwidth` inside `geom_histogram()` to control the size and number of bars that are drawn.

Chapter 5

Simple Comparisons

5.1 Crosstab

5.1.1 Problem

You want to make a crosstab.

5.1.2 Solution

1. Load the `RCPA3` package.
2. In order to create a crosstab in R, we use the function called `crosstabC()`.

3. The function follows the following template:

```r
# use the function `crosstabC()
crosstabC(

  # set `data` equal to your dataset
  data = dataset,

  # `dv` equal to the name of your dependent variable
  dv = dependent_variable,

  # set `iv` equal to the name of your independent variable
  iv = independent_variable,

  # if you have survey data, set `w` equal to the name of your weights variable
  w = weights)
```

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

In this example, we are making a crosstab where the dataset is ‘nes’, the dependent variable is `abortion.imp`, the independent variable is `partyid3`, and the weights are called `wt`.

```
# use the function `crosstabC()
crosstabC(

  # set `data` equal to your dataset
  data = nes,

  # `dv` equal to the name of your dependent variable
  dv = abortion.imp,

  # set `iv` equal to the name of your independent variable
  iv = partyid3,

  # if you have survey data, set `w` equal to the name of your weights variable
  w = wt,
  # do not make the plot
  plot = F)
```

```
## ===========================================================================
##                           Cross-Tabulation Analysis
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-89}Cross-Tabulation of abortion.imp and partyid3, we
##
##                               1. Democrat   2. Independent   3. Republican    Totals
## --------------------------  ------------  ---------------  --------------  --------
## % 1. Not at all important           3.59             6.82            3.67      4.70
## __Count___                        102.67           188.80           95.39    386.86
## % 2. Not too important             10.31            15.16           14.41     13.23
## __Count___                        294.64           419.74          374.26   1088.64
## % 3. Somewhat important            27.28            29.30           25.65     27.44
## __Count___                        779.82           811.47          666.30   2257.59
## % 4. Very important                29.00            25.49           26.57     27.05
## __Count___                        829.02           705.82          690.36   2225.20
## % 5. Extremely important           29.81            23.24           29.71     27.57
## __Count___                        852.18           643.61          771.78   2267.58
## % Totals                          100.00           100.00          100.00    100.00
## __Count___                       2858.33          2769.44         2598.10   8225.87
```

5.1.3 Troubleshooting

- Make sure that the `RCPA3` package is loaded. Use `library(RCPA3)` to load.
- If you specify the data argument, then the arguments for the independent and dependent variables are just the variable names. They do not follow the template of `data$variable`.
- If you do not specify the data argument, then the arguments for the independent and dependent variables do follow the template of `data$variable`.
- Crosstabs are used when both the independent and dependent variables are categorical. Avoid making a crosstab with numeric data.

5.2 Comparison of Means

5.2.1 Problem

You want to make a comparison of means table.

5.2.2 Solution

1. Load the `RCPA3` package.
2. In order to create a comparison of means table in R, we use the function called `compmeansC()`.

3. The function follows the following template:

```r
# use the function `compmeansC()
compmeansC(

  # set `data` equal to your dataset
  data = dataset,

  # `dv` equal to the name of your dependent variable
  dv = dependent_variable,

  # set `iv` equal to the name of your independent variable
  iv = independent_variable,

  # if you have survey data, set `w` equal to the name of your weights variable
  w = weights)
```

Here is a comparison of means table for feelings towards Obama `ft.obama` by party identification `partyid3`.

```r
compmeansC(
  # set `data` equal to your dataset
  data = nes,

  # `dv` equal to the name of your dependent variable
  dv = ft.obama,

  # set `iv` equal to the name of your independent variable
  iv = partyid3,

  # if you have survey data, set `w` equal to the name of your weights variable
  w = wt,
  # do not make the plot
  plot = F)
```

```
## ===========================================================================
##                           Mean Comparison Analysis
## ===========================================================================
##
##
## Table: \label{tab:unnamed-chunk-91}Mean Values of ft.obama by partyid3, Weighted by
##
##                     Mean         N   St. Dev.
## ---------------  ------  --------  ---------
## 1. Democrat       89.90   2839.88      15.45
## 2. Independent    61.67   2748.29      32.33
## 3. Republican     26.98   2578.53      28.30
## Total             60.53   8166.71      36.65
```

5.2.3 Troubleshooting

- Make sure `RCPA3` is loaded.
- You only need to specify the weights argument if you have survey data with survey weights.

5.3 Make a Bar Chart for Two Variables

5.3.1 Problem

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

5.3.2 Solution

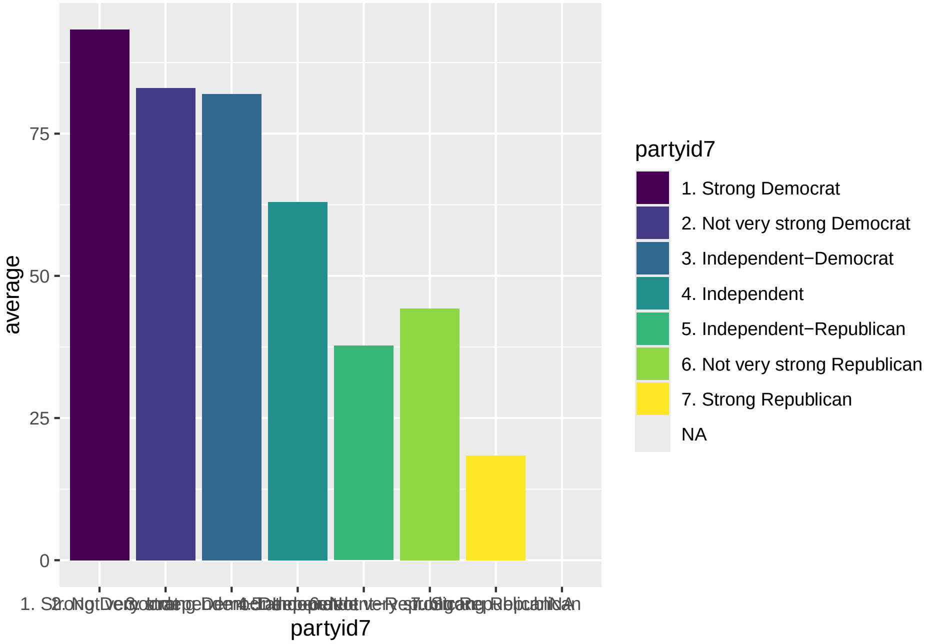
1. Create a summary dataset to plot.
2. Use `ggplot()` for the data and the mapping.

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.

```r
# create a plot object set to `ggplot()`
p1 <- ggplot(

  # specify the data as the name of your summary data
  data = sum_data,

  # specify the mapping
  mapping = aes(

    # set x to the name of the independent variable
    x = independent,

    # set y to the name of the dependent variable
    y = dependent,

    # set `fill` to the name of the independent variable
    fill = independent)) +

  # use `geom_col() to create the bars
  geom_col()

# call the plot using the name of your plot object
p1
```

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

```r
# create the summary dataset
# assign old data to a new data object
partymeans <- nes %>%

  # use group_by() to group the data by partyid7
  group_by(partyid7) %>%

  # calculate the means of ft.obama
  summarise(average = wtd.mean(ft.obama, na.rm = T,
                               w = wt))

# create a plot object set to `ggplot()`
p1 <- ggplot(
 # specify the data as the name of your summary data
 data = partymeans,

 # specify the mapping
 mapping = aes(

  # set x to the name of the independent variable
  x = partyid7,

  # set y to the name of the dependent variable
  y = average,

  # set `fill` to the name of the independent variable
  fill = partyid7)) +

 # use `geom_col() to create the bars
 geom_col()

# call the plot using the name of your plot object
p1
```



5.3.3 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.
- Remember that the name of the dependent variable in the plot is the name that you created in the `summarise()` when creating the summary data.
- You need to load the `tidyverse` package.