MLE 2023

2023-09-01

Section Contents

1 Maxmimum Likelihood Fall 2023

This document will include important links and course notes for fall 2023 Maximum Likelihood Estimation in Political Science.

- This webpage will be updated throughout the semester with new content.
- Sprinkled throughout the website are links to additional resources that might provide more in-depth explanations of a given topic. In particular, several of the sections have benefited from previous materials developed by Kosuke Imai and the book Quantitative Social Science, Chris Bail and SICSS, Marc Ratkovic, In Song Kim, Will Lowe, and others.
- This is a living web resource. If you spot errors or have questions or suggestions, please email me at k.mccabe@rutgers.edu.

2 R Overview

This course will primarily use R for analysis, though we will briefly discuss a few areas where Stata may be more efficient.

Learning to program in R is not a primary goal of this course, but in proceeding through the course, you will gain and/or get practice with a lot of R skills.

For those brand new to R, I strongly recommend you complete the following tutorials prior to or at the beginning of the course.

Goal

By the end of the first week of the course, you will want to have R and RStudio installed on your computer (both free) and feel comfortable using R as a calculator and loading datasets into R.

R and RStudio Installation

- This video from Christopher Bail explains the R and RStudio installation process. This involves

    1. Going to cran, select the link that matches your operating system, and then follow the installation instructions, and
    2. Visiting RStudio and follow the download and installation instructions. R is the statistical software and programming language used for analysis. RStudio provides a convenient user interface for running R code. You do not need RStudio to use R, but it is free and can make your life easier!

https://www.youtube.com/watch?v=ulIv0NiVTs4

- After installing R and RStudio, you can also follow along with Christopher Bail's R Basics and Data Wrangling videos to learn the basic functionality of R.

Supplemental Resources

To supplement the above resources, I would recommend playing around with one of the following:

- An additional great resource is Kosuke Imai's book *Quantitative Social Science*. The first chapter provides a written overview of installing R and the basic functions of R and R Studio, including loading data and packages into R. Data for the book is available at the bottom of this page. Alternate coding in tidyverse for the book is available here.
- If you were having difficulties following Chris Bail or Kosuke Imai's instructions for installation, you can try following the first couple of Rutgers Data Librarian Ryan Womack's videos, which similarly start from the point of installation. They are here. He goes at a slow pace and codes along in the videos. He also has a number of videos on more advanced data analysis topics.
- R for Data Science is another great resource and focuses on "tidyverse" code in R, which can be particularly helpful in data wrangling and visualization.

Note: Much of the code used in the course will rely on "base R" functions (functions that already exist in R). People have also developed tidyverse packages that can be easily installed in R, which supplements base R tools with alternative functions and a syntax based on a particular design philosophy, grammar, and data structure that they find preferable to base R. Using base R vs. tidyverse is often just a matter of personal taste. Either is fine to use in this course, and you will get exposure to code that relies on both.

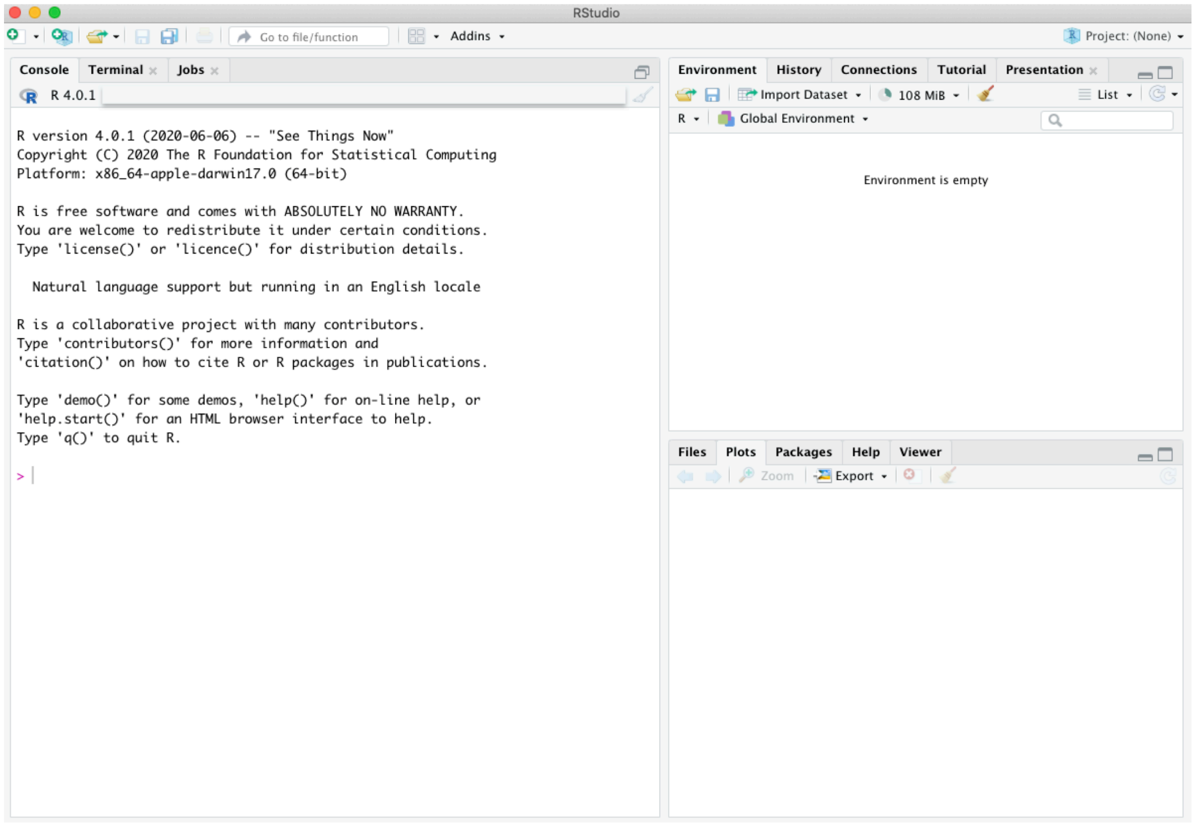
This is a lot of information to digest all at once. Don't worry. No one remembers everything. Plan on going back to these resources often throughout the course and beyond. We will have office hours the first week of the course to help troubleshoot issues.

2.1 First Time with R and RStudio

This next section provides a few notes on using R and RStudio now that you have installed it. This is mostly repetitive of the other resources. This includes only the bare essential information for opening an R script and digging into using R as a calculator. In this section, we cover the following materials:

- Using R as a calculator and assigning objects using `<-`
- Setting your working directory and the `setwd()` function.
- Creating and saving an R script

2.1.1 Open RStudio

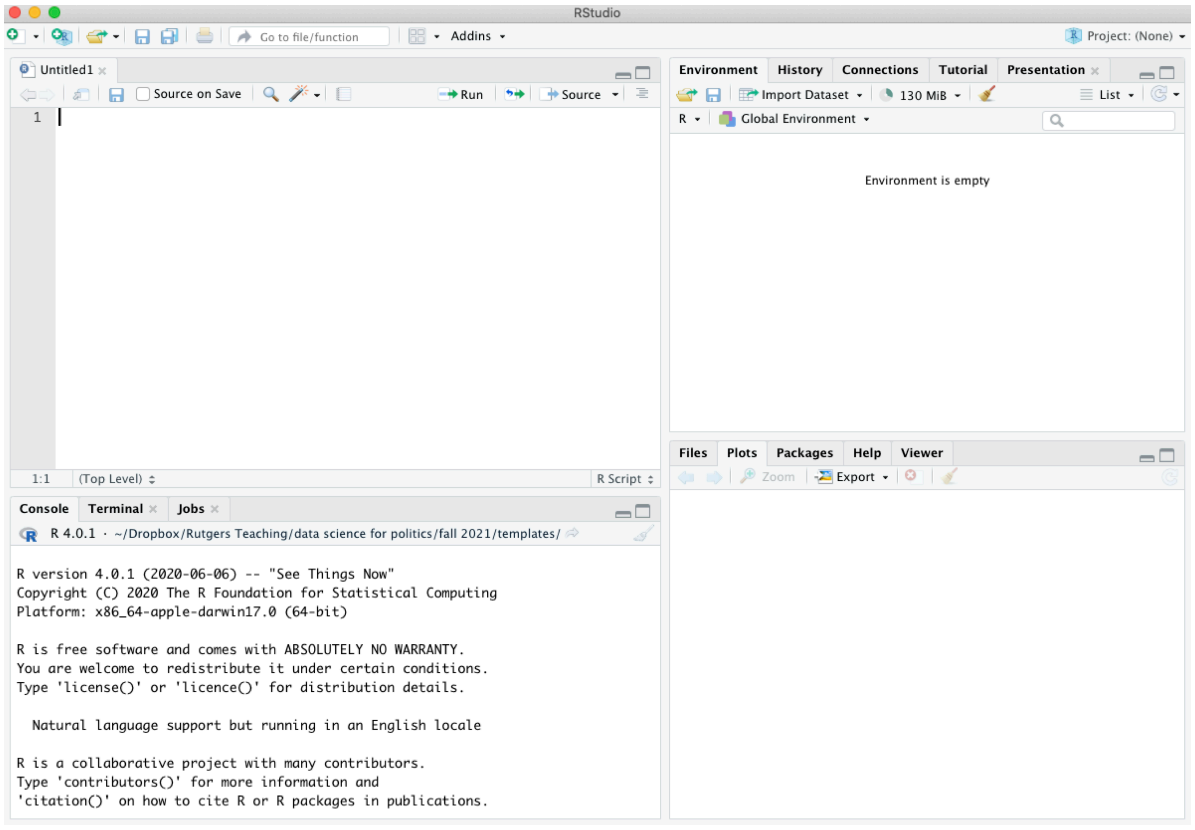


```
R version 4.0.1 (2020-06-06) -- "See Things Now"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Note: The first time you open RStudio, you likely only have the three windows above. We will want to create a fourth window by **opening an R script** to create the fourth window.

- To do this, in RStudio, click on File -> New -> R script in your computer's toolbar. This will open a blank document for text editing in the upper left of the RStudio window. We will return to this window in a moment.
  - You can alternatively click on the green + sign indicator in the top-left corner of the RStudio window, which should give you the option to create a new R script document.

Now you should have something that looks like this, similar to Figure 1.1. in QSS:



```
R version 4.0.1 (2020-06-06) -- "See Things Now"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
```

- The upper-left window has our .R script document that will contain code.
- The lower-left window is the console. This will show the output of the code we run. We will also be able to type directly in the console.
- The upper-right window shows the environment (and other tabs, such as the history of commands). When we load and store data in RStudio, we will see a summary of that in the environment.
- The lower-right window will enable us to view plots and search help files, among other things.

2.1.2 Using R as a Calculator

The *bottom left* window in your RStudio is the Console. You can type in this window to use R as a calculator or to try out commands. It will show the raw output of any commands you type. For example, we can try to use R as a calculator. Type the following in the Console (the bottom left window) and hit "enter" or "return" on your keyboard:

```
5 + 3
```

```
[1] 8
```

```r
5 - 3
```

```
[1] 2
```

```r
5^2
```

```
[1] 25
```

```r
5 * 3
```

```
[1] 15
```

```r
5/3
```

```
[1] 1.666667
```

```r
(5 + 3) * 2
```

```
[1] 16
```

In the other RStudio windows, the upper right will show a history of commands that you have sent from the text editor to the R console, along with other items. The lower right will show graphs, help documents and other features. These will be useful later in the course.

2.1.3 Working in an R Script

Earlier, I asked you to open an R script in the upper left window by doing File, then New File, then R Script. Let's go back to working in that window.

Set your working directory `setwd()`

(Almost) every time you work in RStudio, the first thing you will do is set your working directory. This is a designated folder in your computer where you will save your R scripts and datasets.

There are many ways to do this.

- An easy way is to go to Session → Set Working Directory → Choose Directory. I suggest choosing a folder in your computer that you can easily find and that you will routinely use for this class. Go ahead and create/select it.
- Note: when you selected your directory, code came out in the bottom left Console window. This is the `setwd()` command which can also be used directly to set your working directory in the future.
- If you aren't sure where your directory has been set, you can also type `getwd()` in your Console. Try it now

```
## Example of where my directory was
getwd()
```

If I want to change the working directory, I can go to the top toolbar of my computer and use Session → Set Working Directory → Choose Directory or just type my file pathway using the `setwd()` below:

```
## Example of setting the working directory using setwd().
## Your computer will have your own file path.
setwd("/Users/ktmccabe/Dropbox/Rutgers Teaching/")
```

Saving the R Script

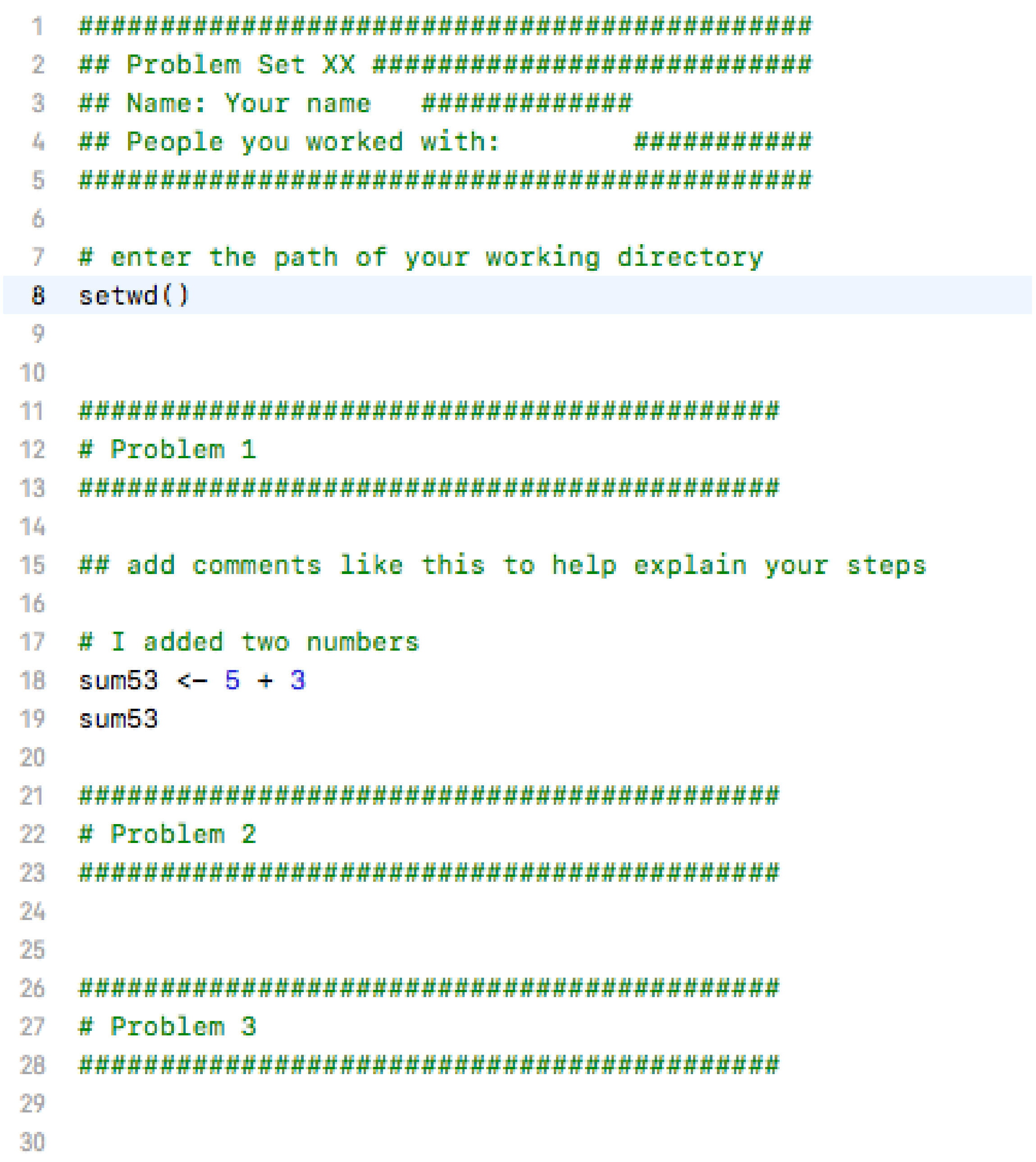
Let's now save our R script to our working directory and give it an informative name. To do so, go to File, then Save As, make sure you are in the same folder on your computer as the folder you chose for your working directory.

Give the file an informative name, such as: "McCabeWeek1.R". Note: all of your R scripts will have the .R extension.

2.1.4 Preparing your R script

Now that we have saved our R script, let's work inside of it. Remember, we are in the top-left RStudio window now.

- Just like the beginning of a paper, you will want to title your R script. In R, any line that you start with a # will not be treated as a programming command. You can use this to your advantage to write titles/comments. Below is a screenshot example of a template R script.
- You can specify your working directory at the top, too. Add your own filepath inside `setwd()`



```
#############################################
## Problem Set XX ###########################
## Name: Your name   #############
## People you worked with:        ###########
#############################################

# enter the path of your working directory
setwd()


###########################################
# Problem 1
###########################################

## add comments like this to help explain your steps

# I added two numbers
sum53 <- 5 + 3
sum53

###########################################
# Problem 2
###########################################


###########################################
# Problem 3
###########################################


```

- Then you can start answering problems in the rest of the script.
- Think of the R script as where you write the final draft of your paper. In the Console (the bottom-left window), you can mess around and try different things, like you might when you are taking notes or outlining an essay. Then, write the final programming steps that lead you to your answer in the R script. For example, if I wanted to add 5 + 3, I might try different ways of typing it in the Console, and then when I found out `5 +`

3 is the right approach, I would type that into my script.

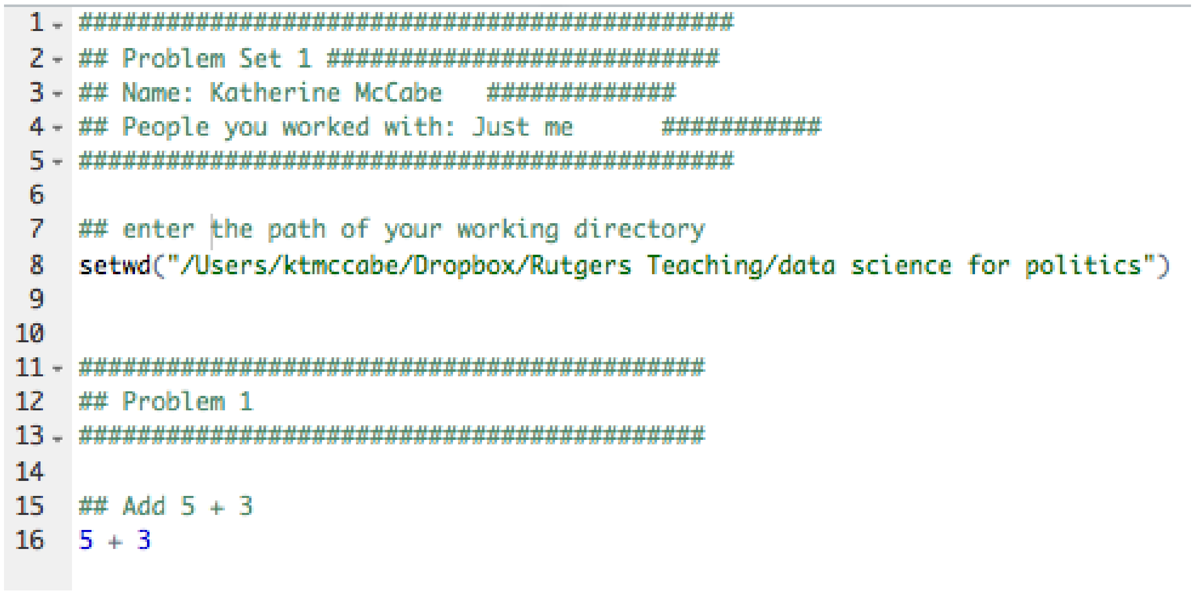
2.1.5 Running Commands in your R script

The last thing we will note in this initial handout is how to execute commands in your R script.

To run / execute a command in your R script (the upper left window), you can

1. Highlight the code you want to run, and then hold down "command + return" on a Mac or "control + enter" on Windows
2. Place your cursor at the end of the line of code (far right), and hit "command + return" on a Mac or "control + return" on Windows, or
3. Do 1 or 2, but instead of using the keyboard to execute the commands, click "Run" in the top right corner of the upper-left window.

Try it: Type `5 + 3` in the R script. Then, try to execute `5 + 3`. It should look something like this:

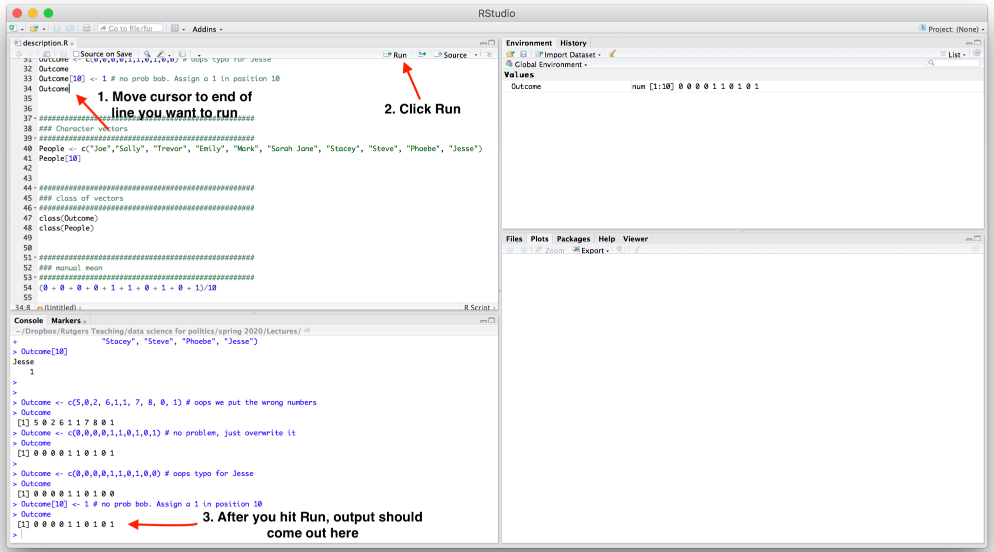


```
#############################################
## Problem Set 1 ###########################
## Name: Katherine McCabe   #############
## People you worked with: Just me      ###########
#############################################

## enter the path of your working directory
setwd("/Users/ktmccabe/Dropbox/Rutgers Teaching/data science for politics")


###########################################
## Problem 1
###########################################

## Add 5 + 3
5 + 3
```

After you executed the code, you should see it pop out in your Console:

```
5 + 3
```

```
[1] 8
```



```
Outcome <- c(0,0,0,0,1,1,0,1,0,0) # oops typo for Jesse
Outcome
Outcome[10] <- 1 # no prob bob. Assign a 1 in position 10
Outcome


##################################################
### Character vectors
##################################################
People <- c("Joe","Sally", "Trevor", "Emily", "Mark", "Sarah Jane", "Stacey", "Steve", "Phoebe", "Jesse")
People[10]


##################################################
### class of vectors
##################################################
class(Outcome)
class(People)


##################################################
### manual mean
##################################################
(0 + 0 + 0 + 0 + 1 + 1 + 0 + 1 + 0 + 1)/10
```

```
Outcome      num [1:10] 0 0 0 0 1 1 0 1 0 1
```

```
~/Dropbox/Rutgers Teaching/data science for politics/spring 2020/Lectures/
+                     "Stacey", "Steve", "Phoebe", "Jesse")
> Outcome[10]
Jesse
    1
>
>
> Outcome <- c(5,0,2, 6,1,1, 7, 8, 0, 1) # oops we put the wrong numbers
> Outcome
 [1] 5 0 2 6 1 1 7 8 0 1
> Outcome <- c(0,0,0,0,1,1,0,1,0,1) # no problem, just overwrite it
> Outcome
 [1] 0 0 0 0 1 1 0 1 0 1
>
> Outcome <- c(0,0,0,0,1,1,0,1,0,0) # oops typo for Jesse
> Outcome
 [1] 0 0 0 0 1 1 0 1 0 0
> Outcome[10] <- 1 # no prob bob. Assign a 1 in position 10
> Outcome
 [1] 0 0 0 0 1 1 0 1 0 1
>
```

Note: The symbol # also allows for annotation behind commands or on a separate line. Everything that follows # will be ignored by R. You can annotate your own code so that you and others can understand what each part of the code is designed to do.

```r
## Example
sum53 <- 5 + 3 # example of assigning an addition calculation
```

2.1.6 Objects

Sometimes we will want to store our calculations as "objects" in R. We use `<-` to assign objects by placing it to the left of what we want to store. For example, let's store the calculation `5 + 3` as an object named `sum53`:

```r
sum53 <- 5 + 3
```

After we execute this code, `sum53 now stores the calculation. This means, that if we execute a line of code that just has`sum53‘, it will output 8. Try it:

```r
sum53
```

```
[1] 8
```

Now we no longer have to type `5 + 3`, we can just type `sum53`. For example, let's say we wanted to subtract 2 from this calculation. We could do:

```r
sum53 - 2
```

```
[1] 6
```

Let's say we wanted to divide two stored calculations:

```r
ten <- 5 + 5
two <- 1 + 1
ten / two
```

```
[1] 5
```

The information stored does not have to be numeric. For example, it can be a word, or what we would call a character string, in which case you need to use quotation marks.

```r
mccabe <- "professor for this course"
mccabe
```

```
[1] "professor for this course"
```

Note: Object names cannot begin with numbers and no spacing is allowed. Avoid using special characters such as % and $, which have specific meanings in R. Finally, use concise and intuitive object names.}

- GOOD CODE: `practice.calc <- 5 + 3`
- BAD CODE: `meaningless.and.unnecessarily.long.name <- 5 + 3`

While these are simple examples, we will use objects all the time for more complicated things to store (e.g., like full datasets!) throughout the course.

We can also store an array or "vector" of information using `c()`

```r
somenumbers <- c(3, 6, 8, 9)
somenumbers
```

```
[1] 3 6 8 9
```

Importance of Clean Code

Ideally, when you are done with your R script, you should be able to highlight the entire script and execute it without generating any error messages. This means your code is clean. Code with typos in it may generate a red error message in the Console upon execution. This can happen when there are typos or commands are misused.

For example, R is case sensitive. Let's say we assigned our object like before:

```r
sum53 <- 5 + 3
```

However, when we went to execute `sum53`, we accidentally typed `Sum53`:

```r
Sum53
```

```
Error in eval(expr, envir, enclos): object 'Sum53' not found
```

Only certain types of objects can be used in mathematical calculations. Let's say we tried to divide `mccabe` by 2:

```r
mccabe / 2
```

```
Error in mccabe/2: non-numeric argument to binary operator
```

A big part of learning to use R will be learning how to troubleshoot and detect typos in your code that generate error messages.

Thomas J. Leeper @thosjleeper · Sep 17
Data science is 90% fixing punctuation and 10% maximum likelihood estimation.

2.1.7 Practice

Below is an exercise that will demonstrate you are able to use R as a calculator and create R scripts.

1. Create an R script saved as "LastnameSetup1.R" (use your last name). Within the R script, follow the example from this handout and title the script.
2. Set your working directory, and include the file pathway (within `setwd()`) at the top of your .R script.
3. Do the calculation 4 + 3 - 2 in R. Store it as an object with an informative name.

4. Do the calculation 5 × 4 in R. Store it as an object with an informative name.
5. Add these two calculations together. In R, try to do this by adding together the objects you created, not the underlying raw calculations.

2.2 Data Wrangling

So you have some data…. AND it's a mess!!!

A lot of the data we may encounter in courses has been simplified to allow students to focus on other concepts. We may have data that look like the following:

```r
nicedata <- data.frame(gender = c("Male", "Female", "Female", "Male"),
             age = c(16, 20, 66, 44),
             voterturnout = c(1, 0, 1, 0))
```

```
  gender age voterturnout
1   Male  16            1
2 Female  20            0
3 Female  66            1
4   Male  44            0
```

In the real world, our data may hit us like a ton of bricks, like the below:

```r
uglydata <- data.frame(VV160002 = c(2, NA, 1, 2),
             VV1400068 = c(16, 20, 66, 44),
             VV20000 = c(1, NA, 1, NA))
```

```
  VV160002 VV1400068 VV20000
1        2        16       1
2       NA        20      NA
3        1        66       1
4        2        44      NA
```

A lot of common datasets we use in the social sciences are messy, uninformative, sprawling, misshaped, and/or incomplete. What do I mean by this?

- The data might have a lot of missing values. For example, we may have `NA` values in R, or perhaps a research firm has used some other notation for missing data, such as a `99`.
- The variable names may be uninformative.

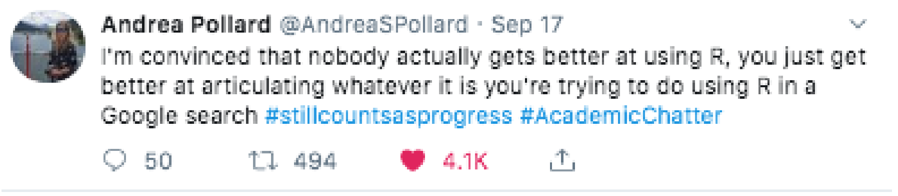
  - For example, there may be no way to know by looking at the data, which variable represents gender. We have to look at a codebook.

- Even if we can tell what a variable is, its categories may not be coded in a way that aligns with how we want to use the data for our research question.

  - For example, perhaps you are interested in the effect of a policy on people below vs. 65 and over in age. Well, your age variables might just be a numeric variable. You will have to create a new variable that aligns with your theoretical interest.

- Datasets are often sprawling. Some datasets may have more than 1000 variables. It is hard to sort through all of them. Likewise, datasets may have millions of observations. We cannot practically look through all the values of a column to know what is there.
- Sometimes we have data shaped into separate columns when we'd rather it be reshaped into different rows.
- Maybe you have encountered a beautiful dataset that provides many measures of your independent variables of interest, but there's one catch– it has no variable related to your outcome! You have to merge data from multiple sources to answer your research question.

Below are a few tips and resources. Ultimately, research is a constant debugging process. Loving R means seeing red error messages. The nice thing about R is that a lot of researchers constantly post coding tips and questions online. Google ends up being your friend, but it's entirely normal to have to devote several hours (days?) to cleaning data.


Andrea Pollard @AndreaSPollard · Sep 17
I'm convinced that nobody actually gets better at using R, you just get better at articulating whatever it is you're trying to do using R in a Google search #stillcountsasprogress #AcademicChatter
50 494 4.1K

2.2.1 Dealing with Uninformative Variable Names

Hopefully, there is an easy fix for dealing with uninformative variable names. I say "hopefully" because hopefully when you encounter a dataset with uninformative variable names, the place where you downloaded the data will also include a codebook telling you what each variable name means, and how the corresponding values are coded.

Unfortunately, this may not always be the case. One thing you can do as a researcher is when you create a dataset for your own work, keep a record (in written form, on a word document or in a pdf or code file) of what each variable means (e.g., the survey question it corresponds to or the exact economic measure), as well as how the values of the variables are coded. This good practice will help you in the short-term, as you pause and come back to working on a project over the course of a year, as well as benefit other researchers in the long run after you publish your research.

For examples of large codebooks, you can view the 2016 American National Election Study Survey and click on a codebook.

I recommend that once you locate the definition of a variable of interest, rename the variable in your dataset to be informative. You can do this by creating a new variable or overwriting the name of the existing variable. You might also comment a note for yourself of what the values mean.

```
## Option 1: create new variable
## gender 2=Male, 1=Female
uglydata$gender <- uglydata$VV160002

## Option 2: Overwrite
names(uglydata)[1] <- "gender2"
```

2.2.2 Dealing with Missing Data

When we have a column with missing data, it is best to do a few things:

- Try to quantify how much missing data there is and poke at the reason why data are missing.
  - Is it minor non-response data?
  - Or is it indicative of a more systematic issue? For example, maybe data from a whole group of people or countries is missing for certain variables.
- If the data are missing at a very minor rate and/or there is a logical explanation for the missing data that should not affect your research question, you may choose to "ignore" the missing data when performing common analyses, such as taking the mean or running a regression.
  - If missing data are a bigger problem, you may consider alternative solutions, such as "imputing" missing data or similarly using some type of auxilliary information to help fill in the missing values.

If we want to figure out how much missing data we have in a variable, we have a couple of approaches:

```
## Summarize this variable
summary(uglydata$gender)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  1.000   1.500   2.000   1.667   2.000   2.000       1 
```

```
## What is the length of the subset of the variable where the data are missing
length(uglydata$gender[is.na(uglydata$gender) == T])
```

```
[1] 1
```

If we choose to ignore missing data, this can often be easily accomplished in common operations. For example, when taking the `mean` we just add an argument `na.rm = T`:

```
mean(uglydata$VV1400068, na.rm=T)
```

```
[1] 36.5
```

If we do a regression using `lm` or `glm`, R will automatically "listwise" delete any observation that has missing data (`NA`) on any of the variables in our regression model.

We should always be careful with missing data to understand how R is treating it in a particular scenario.

For example if we were to run `table(uglydata$gender)`, we would have no idea there were missing data unless we knew that the total number of observations `nrow(uglydata)` was greater than 3. The `table()` command is omitting the missing values by default.

```
table(gender= uglydata$gender)
```

```
gender
1 2 
1 2 
```

2.2.3 Dealing with Variable Codings that Aren't Quite Right

Often times the ways that variables are coded in datasets we get off-the-shelf are not coded exactly as how we were dreaming up operationalizing our concepts. Instead, we are going to have to wrangle the data to get them into shape.

This may involve creating new variables that recode certain values, creating new variables that collapse some values into a smaller number of categories, combining multiple variables into a single variable (e.g., representing the average), or setting some of the variable values to be missing (`NA`). All of these scenarios may come up when you are dealing with real data.

Chapter 2 of Kosuke Imai's book *Quantitative Social Science* walks through some examples of how to summarize your data, subset the data (2.2.3), create new variables using conditional statements (Section 2.2.4, e.g., "If age is below 65, assign the new variable a value of"0", otherwise, assign it a value of "1"), and creating new factor variables (2.2.5, e.g., coding anyone who is Protestant, Catholic, or Lutheran in the data as "Christian").

Here is a short video working through the example from 2.2.4 using conditional statements to construct new variables. It uses the `resume` dataframe, which can be loaded below.

```
resume <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/resume.cs
```

https://www.youtube.com/watch?v=lMY2t4SQ0Pg

R Studio has its own set of primers on various topics, including summarizing and working with data. See the Work with Data primer, as well as the full list of other topics. These will often rely on tidyverse coding.

2.2.4 Dealing with Incomplete Data (Merging!)

Sometimes in order to answer our research questions, we need to combine data from multiple sources. If we have a large amount of data, this may be quite daunting. Fortunately, R has several commands that allow us to merge or append datasets.

Here is a video working through examples of merging and appending data based on the tutorial below.

https://www.youtube.com/watch?v=NwwlB6Lp1TY

Here are a few resources on merging and appending data:

- Using the `merge` command in R. See Explanation.
  - It will look for variable(s) held in common between datasets and join the datasets by the matching values on these variables.
- Appending data in R (e.g., Maybe you have one dataset from 2010 and another from 2012, and you want to stack them on top of each other). See Explanation.

Some merging problems are extremely difficult. For example, some researchers need to merge large datasets–like the voter file– with other administrative records. However, how someone's name is displayed in one dataset might not match at all with the other dataset. For these complex problems, we might need "fuzzy matching." Here is an R package that helps with this more complex case and related paper.

2.2.5 Dealing with Poorly Shaped Data

Data can come in a variety of shapes and sizes. It's a beautiful disaster.

Sometimes it's particularly useful to have data in wide formats, where every row relates to a particular unit of data– such as a country or a survey respondent. And perhaps each column represents information about that unit at a particular point in time. For example, perhaps you have a column with information on that subject for the past five years.

```r
countrywide <- data.frame(country = c("Canada", "USA"),
                          economy2016 = c(10, 12),
                          economy2017 = c(11, 11),
                          economy2018 = c(9, 5),
                          economy2019 = c(13, 8),
                          economy2020 = c(12, 6))
```

```
  country economy2016 economy2017 economy2018 economy2019 economy2020
1  Canada          10          11           9          13          12
2     USA          12          11           5           8           6
```

However, other times, it would be more useful to you, as you dig into your data analysis, to have this information arranged in "long" format, such that every row is now a unit-year combination. You have a row for Canada in 2020, a row for Canada in 2019, and so on. Countries are now represented in multiple rows of your data.

```r
countrylong <- data.frame(country = rep(c("Canada", "USA"),5),
                          year = 2016:2020,
                          economy= c(10, 12,11, 11,9, 5,13, 8,12, 6))
```

```
   country year economy
1   Canada 2016      10
2      USA 2017      12
3   Canada 2018      11
4      USA 2019      11
5   Canada 2020       9
6      USA 2016       5
7   Canada 2017      13
8      USA 2018       8
9   Canada 2019      12
10     USA 2020       6
```

Ultimately, different shapes of data are advantageous for different research questions. This means it is best if we have a way to (at least somewhat) easily shift between the two formats.

Here is a resource on how to "reshape" your data between wide and long from UCLA.

Here are a few additional resources:

- More on reshape – For the tidyverse fans. Using `gather()` and `spread()` in tidyverse from R for Data Science and explained by Chris Bail here.

2.2.6 Subsetting data by rows and columns

Sometimes we do not want to deal with our entire dataset for an analysis. Instead, we might want to only analyze certain rows (e.g., maybe if we are just studying Democrats, for example). Similarly, we might have a dataframe with 1000 columns, from which we are only using about 20. We might want to remove those extra columns to make it easier to work with our dataframes.

Below are a few examples of subsetting data and selecting columns. We will use the `resume` dataset from the Kosuke Imai QSS book for demonstration. This is a dataset from an experiment describing whether certain applicants, who varied in the gender (`sex`) and race (`race`) signaled by their name (`firstname`), received callbacks (`call`) for their employment applications.

Here is a short video working through these examples.

https://www.youtube.com/watch?v=6EdllKtaDHE

Let's load the data.

```
resume <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/resume.cs
```

Subset particular rows

To do this, put the row numbers you want to keep on the left side of the comma. Putting nothing on the right side means you want to keep all columns.

```
## numerically
resume[1,] # first row
```

```
  firstname    sex  race call
1   Allison female white    0
```

```
resume[1:4,] # first through 4th rows
```

```
  firstname    sex  race call
1   Allison female white    0
2   Kristen female white    0
3   Lakisha female black    0
4   Latonya female black    0
```

```r
resume[c(1, 3, 4),] # 1, 3, 4 rows
```

```
  firstname    sex  race call
1   Allison female white    0
3   Lakisha female black    0
4   Latonya female black    0
```

Using the `subset` command with logical expressions `> >= == < <= !=`

```r
## by logical expressions
women <- subset(resume, sex == "female")
women <- resume[resume$sex == "female", ] ## alternative

calledback <- subset(resume, call == 1)
calledback <- subset(resume, call > 0)
```

And or Or statements `&` or `|`

```r
blackwomen <- subset(resume, sex == "female" & race == "black")
bradbrendan <- subset(resume, firstname == "Brad" |
                        firstname == "Brendan")
```

The tidyverse also has commands for subsetting. Here is an example using `filter`.

```r
library(tidyverse)
blackwomen <- resume %>%
  filter(sex == "female" & race == "black")
```

Selecting particular columns

Note, now we are working on the right side of the comma.

```r
## numerically
first <- resume[, 1] # first column
firstsecond <- resume[, 1:2] # first and second column
notfourth <- resume[, -4] # all but the fourth column
```

```
## by labels
justthese <- resume[, c("firstname", "sex")]
```

Using the `select` command

```
## install.packages("dplyr")
library(dplyr)
subdata <- resume %>% dplyr::select(firstname, sex) ## just these
subdata2 <- resume %>% dplyr::select(-firstname, -sex) ## all but these two
```

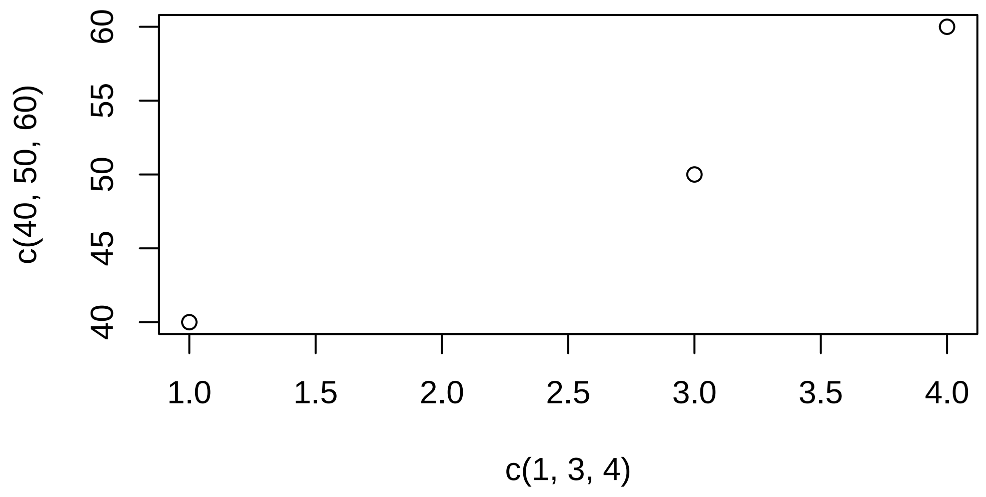
2.2.7 Visualizing Data

There are many commands for plotting your data in R. The most common functions in base R are `plot()`, `barplot()` and `hist()`. You will see many examples throughout the notes with each of these functions.

To get you started, the most simple thing to note about the `plot()` command is that it is based on a coordinate system. You specify the `x` and `y` coordinates for which you want to plot a series of points.

For example, here is a plot at points 1,40; 3,50; and 4,60.

```
plot(x = c(1,3,4), y=c(40, 50, 60))
```

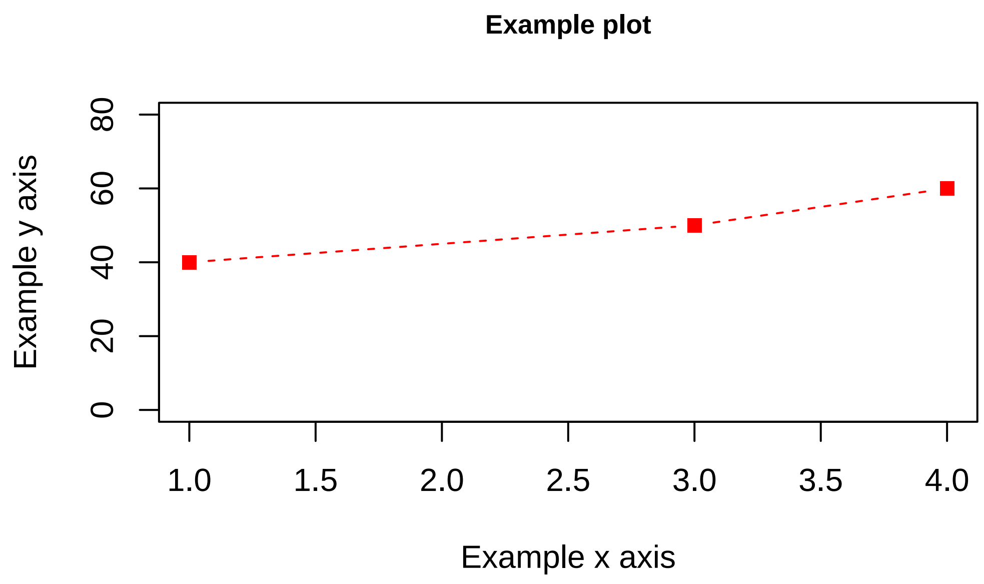


Instead of putting raw numbers as the coordinates, you can provide object names. E.g.,

```
xcoord <- c(1,3,4)
ycoord <- c(40, 50, 60)
plot(x = xcoord, y=ycoord)
```

Beyond that, you can play around with many aesthetics in R, such as the `type`, `pch`, `lty`, as well as labels `main`, `ylab`, `xlab`, font sizes `cex.main`, `cex.axis`, `cex.lab`, and axis limits `ylim`, `xlim`. Below is an example. Play around with changing some of the specifications, and see how the plot changes.

```
xcoord <- c(1,3,4)
ycoord <- c(40, 50, 60)
plot(x = xcoord, y=ycoord,
     main = "Example plot",
     ylab= "Example y axis",
     xlab = "Example x axis",
     cex.main = .8,
     ylim = c(0, 80),
     xlim = c(1,4),
     pch = 15,
     col="red",
     type = "b",
     lty=2)
```

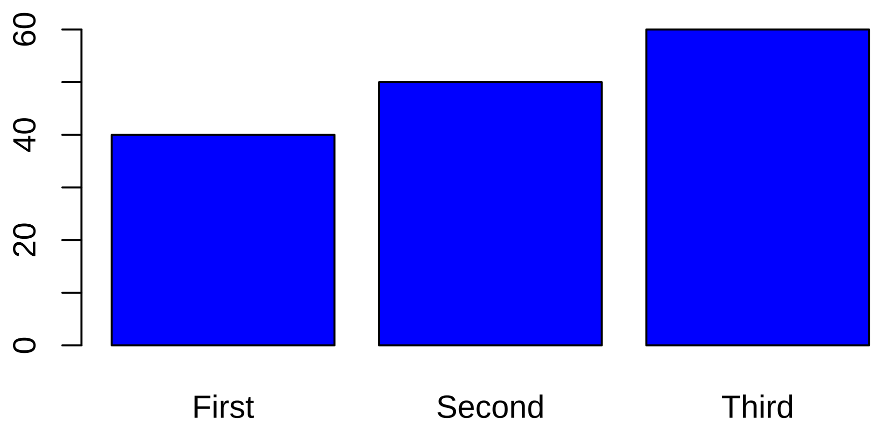


The function `barplot` takes a single vector of values. This can be a raw vector you have created or a `table` object or `tapply` object, for example, displaying the counts of different observations or means.

You can add a `names.arg` argument to specify particular names for each bar. Many of the other aesthetics are the same as `plot`. You can play around with adding aesthetics.

Example:

```
barplot(ycoord,
        names.arg = c("First", "Second", "Third"),
        col="blue")
```



For more on visualizing data, you can see the RStudio primers.

R also has a package called `ggplot2` which includes the function `ggplot` and many elaborate ways to plot your data. The `gg` stands for the grammar of graphics. For a video introduction to `ggplot` I recommend watching Ryan Womack's video from 27:30 on. It uses the data `diamonds` which can be loaded in R through the following command. See approximately minute 35 for an example with a bar plot.

```
library(ggplot2)
data(diamonds)
```

https://www.youtube.com/watch?v=c9DZ8lBwxZs

2.2.8 Reproducing your steps

It is really important to keep a record of all of the changes you have made to the original data. An R script or R markdown file is a useful way to do this, so long as you add comments that explain what you are doing.

You want to get your code to a place when a stranger can open your R file, load your data, and reproduce each step you took to get to the final results— all while never even needing to contact you with questions. That can be difficult, but it's good to aim for that high bar, even if sometimes, we fall short in how we are documenting each step.

This website provides a nice introduction to R Markdown, one tool for embedding R code inside textual documents. See here.

If you want to get advanced with reproducibility, you can watch Chris Bail's on the subject describing the uses of R Markdown and GitHub among other tools for communicating and collaboration. He also links to other resources.

2.3 Tools for writing up results

2.3.1 R Markdown

R Markdown is a free tool within RStudio that allows you to weave together text and code (along with images and tables) into the same document. It can compile these documents (written inside R Studio) into html, pdf, or Word doc output files. R Markdown can be incredibly helpful for doing things like generating replication files, writing up problem sets, or even writing papers.

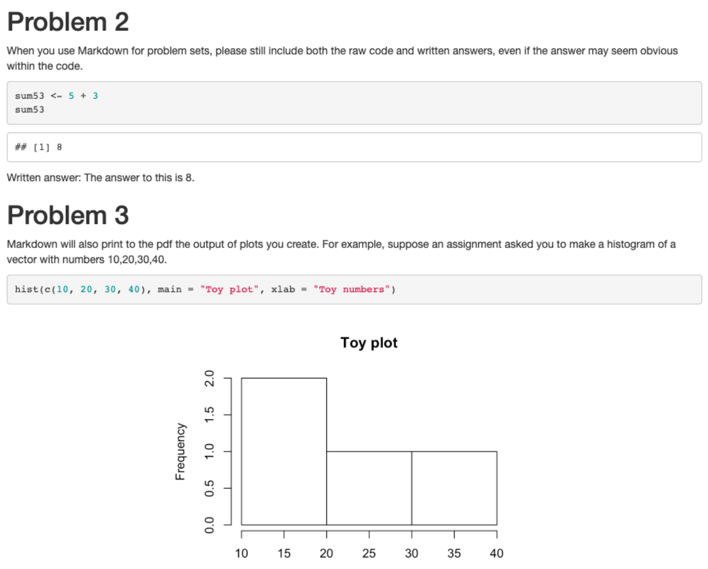
This site includes an introduction to R Markdown.

- See also here and here

R Markdown has its own syntax, which includes functionality for writing mathematical equations. The pdf output option in R Markdown requires LaTex, described in the next section. Below we work with just the html output.

R Markdown documents can be "compiled" into html, pdf, or docx documents by clicking the `Knit` button on top of the upper-left window. Below is an example of what a compiled html file looks like.

- Note that the image has both written text and a gray chunk, within which there is some R code, as well as the output of the R code (e.g., the number 8 and the image of the



Problem 2

When you use Markdown for problem sets, please still include both the raw code and written answers, even if the answer may seem obvious within the code.

```
sum53 <- 5 + 3
sum53
```

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

Written answer: The answer to this is 8.

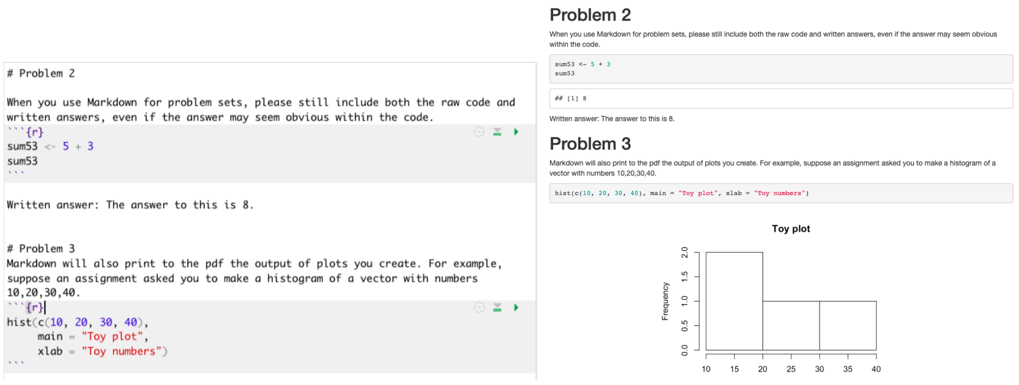
Problem 3

Markdown will also print to the pdf the output of plots you create. For example, suppose an assignment asked you to make a histogram of a vector with numbers 10,20,30,40.

```
hist(c(10, 20, 30, 40), main = "Toy plot", xlab = "Toy numbers")
```

histogram plot.

We say this is a "compiled" RMarkdown document because it differs from the raw version of the file, which is a .Rmd file format. Below is an example of what the raw .Rmd version looks like, compared to the compiled html version.



~~~
# Problem 2

When you use Markdown for problem sets, please still include both the raw code and
written answers, even if the answer may seem obvious within the code.
```{r}
sum53 <- 5 + 3
sum53
~~~

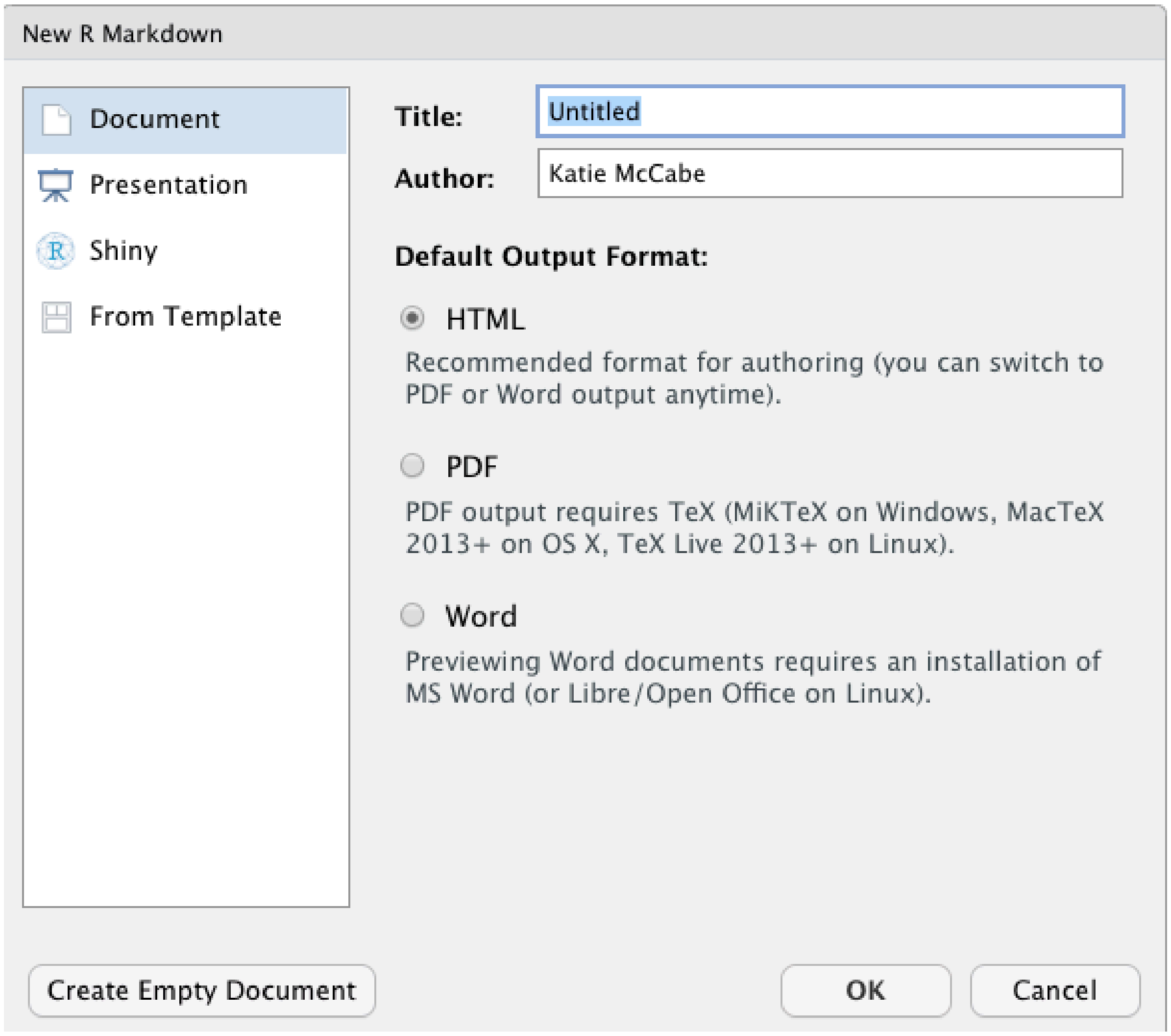
Written answer: The answer to this is 8.

Problem 3
Markdown will also print to the pdf the output of plots you create. For example,
suppose an assignment asked you to make a histogram of a vector with numbers
10,20,30,40.
```{r}
hist(c(10, 20, 30, 40),
     main = "Toy plot",
     xlab = "Toy numbers")
```
```

Problem 2

When you use Markdown for problem sets, please still include both the raw code and written answers, even if the answer may seem obvious within the code.

```
sum53 <- 5 + 3
sum53
```

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

Written answer: The answer to this is 8.

Problem 3

Markdown will also print to the pdf the output of plots you create. For example, suppose an assignment asked you to make a histogram of a vector with numbers 10,20,30,40.

```
hist(c(10, 20, 30, 40), main = "Toy plot", xlab = "Toy numbers")
```

Getting started with RMarkdown

Just like with a regular R script, to work in R Markdown, you will open up RStudio.

- For additional support beyond the notes below, you can also follow the materials provided by RStudio for getting started with R Markdown https://rmarkdown.rstudio.com/lesson-1.html.

The **first time** you will be working in R Markdown, you will want to install two packages: `rmarkdown` and `knitr`. You can do this in the Console window in RStudio (remember the lower-left window!).

Type the following into the Console window and hit enter/return.
```

```r
install.packages("rmarkdown")
install.packages("knitr")
```

Once you have those installed, now, each time you want to create an R Markdown document, you will open up a .Rmd R Markdown file and get to work.

1. Go to File -> New File -> R Markdown in RStudio
    - Alternatively, you can click the green + symbol at the top left of your RStudio window
2. This should open up a window with several options, similar to the image below
    - Create an informative title and change the author name to match your own
    - For now, we will keep the file type as html. In the future, you can create pdf or .doc documents. However, these require additional programs installed on your computer, which we will not cover in the course.


New R Markdown

Title: Untitled

Author: Katie McCabe

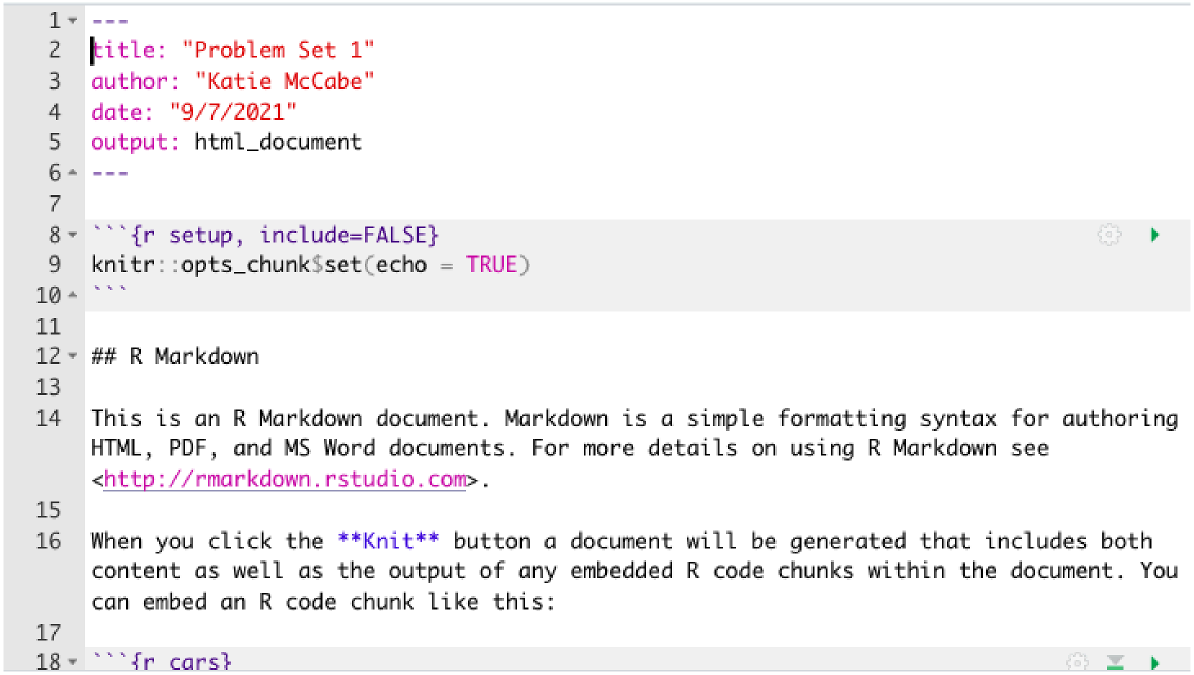
Default Output Format:

[✓ HTML]
Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

3. After you hit "OK" a new .Rmd script file will open in your top-left window with some template language and code chunks, similar to the image below. Alternatively, you can start from scratch by clicking "Create Empty Document" or open a template .Rmd file of your own saved on your computer.



~~~
---
title: "Problem Set 1"
author: "Katie McCabe"
date: "9/7/2021"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
~~~

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
```

4. ***Save as .Rmd file.*** Save the file by going to "File -> Save as" in RStudio

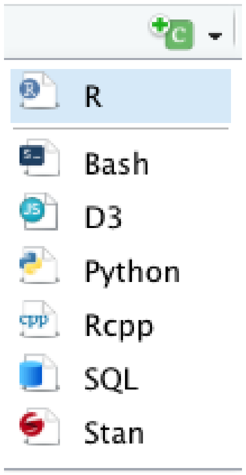
- Give the file an informative name like your LastnamePractice1.Rmd

5. ***Key Components.*** Now you are ready to work within the Rmd script file. We will point to four basic components of this file, and you can build your knowledge of RMarkdown from there.

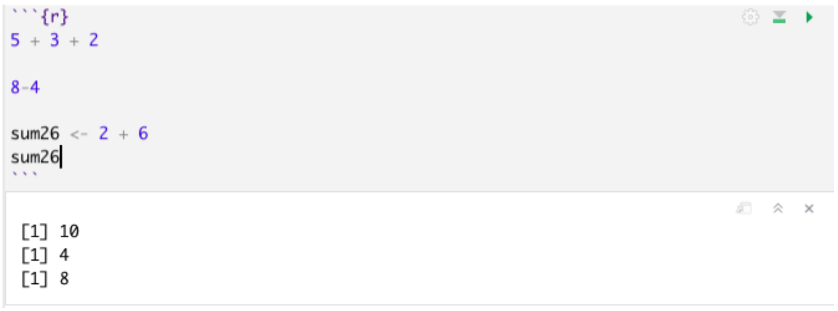
   1. The top part bracketed by `---` on top and bottom is the YAML component. This tells RStudio the pertinent information about how to "compile" the Rmd file.
      - Most of the time you can leave this alone, but you can always edit the title, author, or date as you wish.
   2. The next component are the global options for the document. It is conveniently labeled "setup." By default what this is saying is that the compiled version will "echo" (i.e., display all code chunks and output) unless you specifically specify otherwise. For example, note that it says `include = FALSE` for the setup chunk. That setting means that this code chunk will "run" but it will not appear in the nicely compiled .html file.
      - Most of the time you will not need to edit those settings.
   3. The third component I want to bring attention to is the body text. The # symbol in RMarkdown is used to indicate that you have a new section of the document. For example, in the compiled images at the beginning, this resulted in the text being larger and bolded when it said "Problem 2." In addition to just using a single #,

using ## or ### can indicate subsections or subsubsections. Other than that symbol, you can generally write text just as you would in any word processing program, with some exceptions, such as how to make text bold or italicized.

4. The final component I want to call attention to are the other main body code chunks. These are specific parts of the document where you want to create a mini R script. To create these, you can simply click the + C symbol toward the top of the top left window of RStudio and indicate you want an R chunk.



6. ***Writing R Code.*** Within a code chunk, you can type R code just like you would in any R script, as explained in the previous section. However, in RMarkdown, you also have the option of running an entire code chunk at once by hitting the green triangle at the top-right of a given code chunk.



~~~
```{r}
5 + 3 + 2

8-4

sum26 <- 2 + 6
sum26
```
~~~

```
[1] 10
[1] 4
[1] 8
```

7. ***Knitting the document.*** Once you have added a code chunk and/or some text, you are ready to compile or "Knit" the document. This is what generates the .html document.

   - To do so, click on the Knit button toward the top of the top-left window of Rstudio. After a few moments, this should open up a preview window displaying the compiled html file.
   - It will also save an actual .html file in your working directory (the same location on your computer where you have saved the .Rmd file)

- Try to locate this compiled .html file on your computer and open it. For most computers, .html files will open in your default web browser, such as Google Chrome or Safari.
- This step is a common place where errors are detected and generated. Sometimes the compiling process fails due to errors in the R code in your code chunks or an error in the Markdown syntax. If your document fails to knit, the next step is to try to troubleshoot the error messages the compiling process generates. The best way to reduce and more easily detect errors is to "knit as you go." Try to knit your document after each chunk of code you create.

2.3.2 LaTex

LaTex is a typesetting program for drafting documents, much like Microsoft Word. Some advantages of using LaTex in writing empirical research papers is, once you learn the basics of the program, it can become easier to add tables, figures, and equations into a paper with little effort. The downside is that it has its own syntax, which takes a little time to learn. LaTex also has a feature called "beamer" which uses LaTex to generate slides. You can use this for presentations. LaTex "compiles" documents into pdf files.

Here is one introduction for getting started with LaTex that starts from the installation stage. Here also is a link to a set of slides from Overleaf discussing the basics of LaTex: Slides

You can download the Tex distribution for your operating system here. In addition to this, there are many programs available for running LaTex on your computer, which range from free, very basic tools (e.g., TexWorks) to tools with fancy capabilities.

Overleaf is an online program for drafting LaTex documents. It has a nice feature where it allows you to share a document so that multiple people can work on the document simultaneously. This makes Overleaf a great program for projects where you have co-authors. The basic tools in Overleaf are available for free, but if you want to start sharing documents with a lot of co-authors, it requires a paid account.

LaTex also has the ability to integrate citations into your documents. The part 2 tutorial from Overleaf goes over this.

RStudio also has a program built-in called Sweave (.Rnw) documents that works with knitR, which weave together R code and LaTex syntax, allowing you to compile them into pdf documents and slide presentations. This is very similar to how R Markdown works, but with somewhat different syntax. See here for an overview. Your problem sets are generally Sweave/knitR documents.

2.3.3 Formatting and Exporting R Results

R has a number of tools, including the packages `texreg`, `xtable`, and `stargazer`, which can be used to export tables made in R to nicely formatted LaTex or html output.

Here is a link to the `texreg` package documentation. Section 5 has examples of the `texreg` and `htmlreg` functions within the texreg package. These can be integrated into R Markdown and Sweave documents, and their output can be pasted into LaTex or Microsoft Word.

Your choice of function will depend on where you ultimately want your results to be compiled. If you are generating results that will be compiled to pdf using LaTex, then `texreg` works well. If you are exporting results to Word, than you may wish to use the `htmlreg` function within the `texreg` package, which will generate output that can be pasted into Word.

A simple example using R Markdown html output. (Note, if you wanted to export the table to Word, you would add an argument specifying `file = "myfit.doc"` to the function. See the above link for examples:

```r
mydata <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/resume.cs
fit <- lm(call ~ race, data=mydata)
```

```r
## First time you use texreg, install it
install.packages("texreg")
```

```r
library(texreg)
htmlreg(list(fit),
        stars=c(0.001, 0.01, 0.05),
        caption = "Regression of Call Backs on Race")
```

Regression of Call Backs on Race

Model 1

(Intercept)

0.06***

(0.01)

racewhite

0.03***

(0.01)

R2

0.00

Adj. R2

0.00

Num. obs.

4870

***p < 0.001; **p < 0.01; *p < 0.05

You can add more arguments to the function to customize the name of the model and the coefficients. You can also add multiple models inside the list argument, for example, if you wanted to present a table with five regression models at once. Here is an example with two:

```r
fit2 <- lm(call ~ race + sex, data=mydata)

library(texreg)
htmlreg(list(fit, fit2),
        stars=c(0.001, 0.01, 0.05),
        caption = "Regression of Call Backs on Race and Sex")
```

Regression of Call Backs on Race and Sex

Model 1

Model 2

(Intercept)

0.06***

0.07***

(0.01)

(0.01)

racewhite

0.03***

0.03***

(0.01)

(0.01)

sexmale

-0.01

(0.01)

R2

0.00

0.00

Adj. R2

0.00

0.00

Num. obs.

4870

4870

***p < 0.001; **p < 0.01; *p < 0.05

2.3.4 Additional formatting examples

Here are some additional examples with different formats. You can run them on your own computer to see what the output looks like.

The package `texreg` has three primary formats

- `texreg()` for LATEX output;
- `htmlreg()` for HTML, Markdown-compatible and Microsoft Word-compatible output;
- `screenreg()` for text output to the R console.

If you are working with a LaTex document, I recommend using `texreg()`, which will output LaTex syntax in your R console, which you can copy and paste into your article document.

Note: this function allows you to customize model and coefficient names.

```r
library(texreg)
texreg(list(fit, fit2),
        stars=c(0.001, 0.01, 0.05),
        caption = "Regression of Call Backs on Race and Sex",
       custom.model.names = c("Bivariate", "Includes Sex"),
       custom.coef.names = c("Intercept",
                             "Race- White",
                             "Sex- Male"))
```

If you are working with a Microsoft Word document, I recommend using `htmlreg()` and specifying a file name for your output. This will export a file to your working directory, which you can copy and paste into your Word article document. Otherwise, the syntax is the same as above.

```r
library(texreg)
htmlreg(list(fit, fit2), file = "models.doc",
        stars=c(0.001, 0.01, 0.05),
        caption = "Regression of Call Backs on Race and Sex",
       custom.model.names = c("Bivariate", "Includes Sex"),
       custom.coef.names = c("Intercept",
                             "Race- White",
                             "Sex- Male"))
```

If you are trying to read the output in your R console, that's when I would use `screenreg()`. However, for professional manuscript submissions, I would recommend the other formats.

```r
library(texreg)
screenreg(list(fit, fit2),
        stars=c(0.001, 0.01, 0.05),
        caption = "Regression of Call Backs on Race and Sex",
       custom.model.names = c("Bivariate", "Includes Sex"),
       custom.coef.names = c("Intercept",
                             "Race- White",
                             "Sex- Male"))
```

The package `stargazer` allows similar options. I don't think there are particular advantages to either package. Whatever comes easiest to you. The default for stargazer will output LaTex code into your R console.

- Note that the syntax is similar but has slightly different argument names from the `texreg` package.
- Also, the intercept is at the bottom by default for `stargazer`. Be careful of the covariate ordering when you add labels.

```r
library(stargazer)
stargazer(list(fit, fit2),
         star.cutoffs=c(0.05,0.01, 0.001),
         title= "Regression of Call Backs on Race and Sex",
         dep.var.labels.include = F,
        column.labels = c("Call Back", "Call Back"),
        covariate.labels = c("Race- White",
                             "Sex- Male",
                             "Intercept"))
```

You can adjust the type of output in `stargazer` for other formats, similar to `texreg`. Here is an example of Microsoft Word output.

```r
library(stargazer)
stargazer(list(fit, fit2), out = "modelstar.doc", type="html",
         star.cutoffs=c(0.05,0.01, 0.001),
         dep.var.labels.include = F,
         title= "Regression of Call Backs on Race and Sex",
        column.labels = c("Call Back", "Call Back"),
        covariate.labels = c("Race- White",
                             "Sex- Male",
                             "Intercept"))
```

2.3.5 Additional Table Types

Sometimes you might want to create tables that are not from regression models, such as tables for descriptive statistics. R has other packages for tables of this type.

For example `xtable` can create simple html and latex tables. You just have to supply the function with a table object or matrix.

```r
library(xtable)
table1 <- table(race = mydata$race, sex = mydata$sex)
```

```r
## LaTeX
xtable(table1)
```

```
## Word
print(xtable(table1), type="html", file = "crosstab.doc")
```

```
## Html
print(xtable(table1), type="html")
```

female

male

black

1886

549

white

1860

575

2.4 Exploratory Data Analysis Tools

One of the first things you may want to do when you have a new dataset is to explore! Get a sense of the variables you have, their class, and how they are coded. There are many functions in base R that help with this, such as `summary()`, `table()`, and descriptive statistics like `mean` or `quantile`.

Let's try this with the built-in `mtcars` data.

```
data("mtcars")

summary(mtcars$cyl)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  4.000   4.000   6.000   6.188   8.000   8.000 
```

```
quantile(mtcars$wt)
```

```
     0%     25%     50%     75%    100% 
1.51300 2.58125 3.32500 3.61000 5.42400 
```

```r
mean(mtcars$mpg, na.rm=T)
```

```
[1] 20.09062
```

```r
sd(mtcars$mpg, na.rm=T)
```

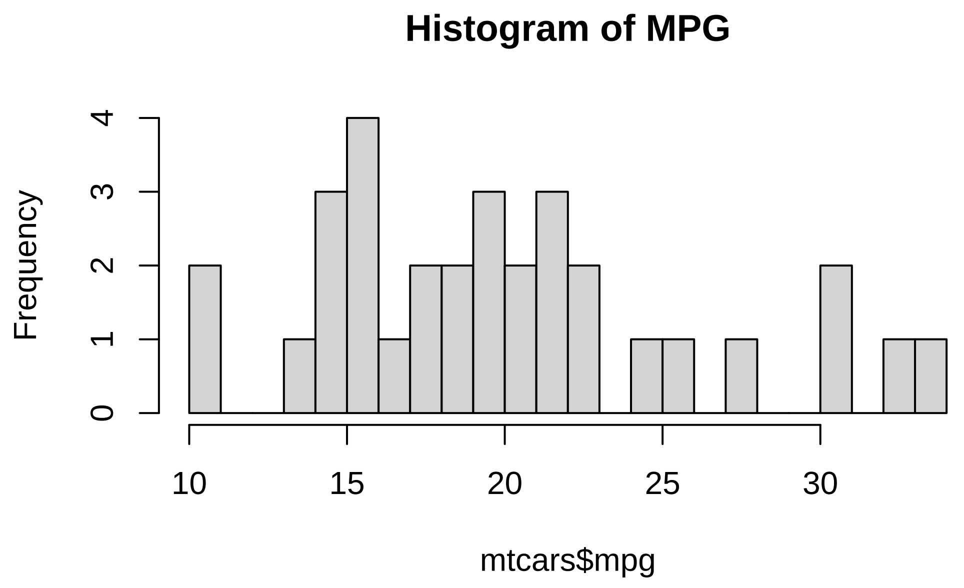
```
[1] 6.026948
```

```r
table(gear=mtcars$gear, carb=mtcars$carb)
```

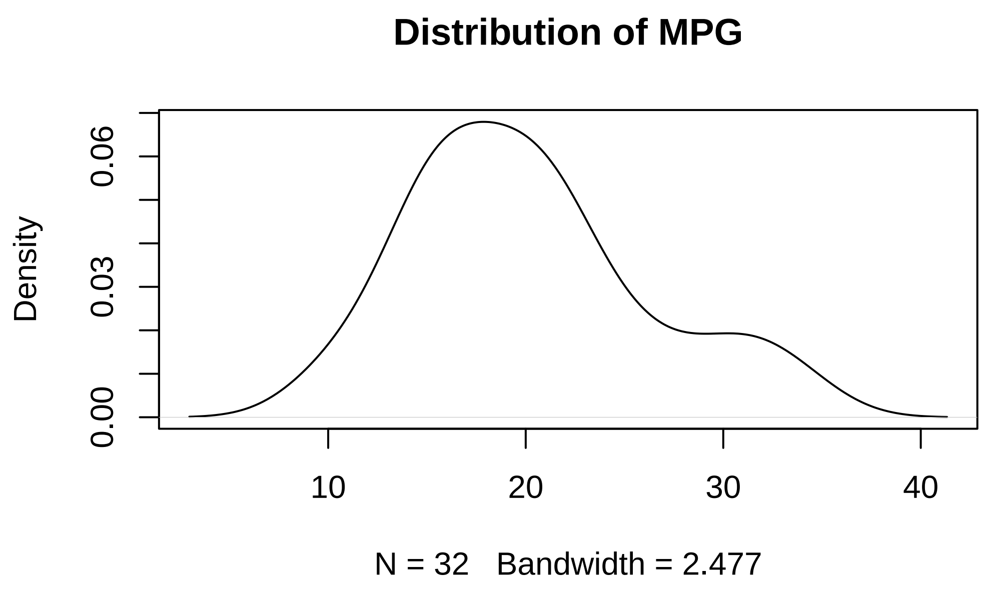
```
    carb
gear 1 2 3 4 6 8
   3 3 4 3 5 0 0
   4 4 4 0 4 0 0
   5 0 2 0 1 1 1
```

As discussed in the visualization, you can also quickly describe univariate data with histograms, barplots, or density plots using base R or `ggplot`.

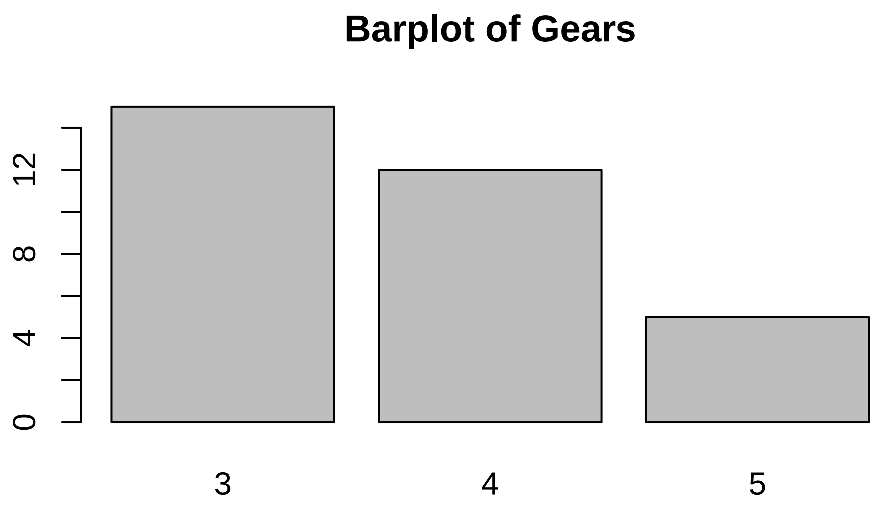
```r
hist(mtcars$mpg, breaks=20, main="Histogram of MPG")
```



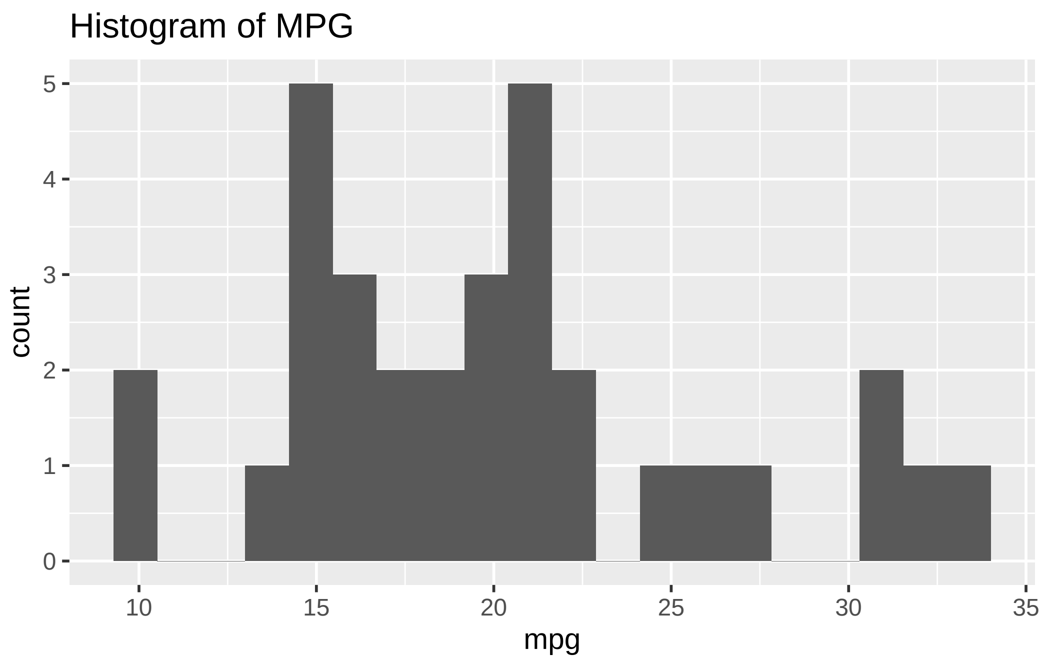
```r
plot(density(mtcars$mpg, na.rm=T),
     main="Distribution of MPG")
```



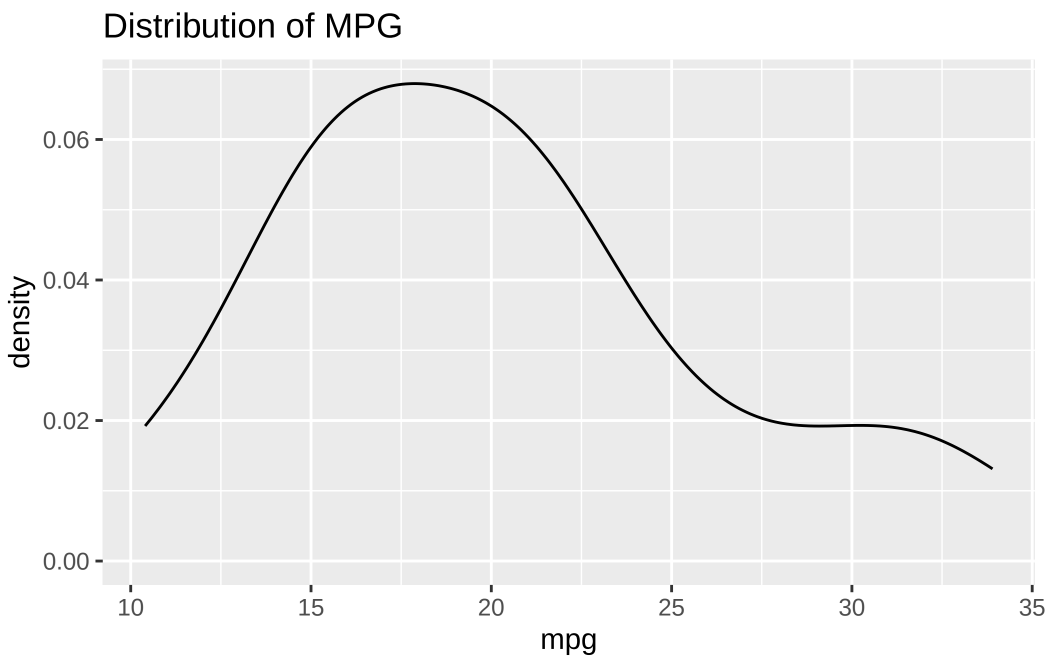
```r
barplot(table(mtcars$gear), main="Barplot of Gears")
```



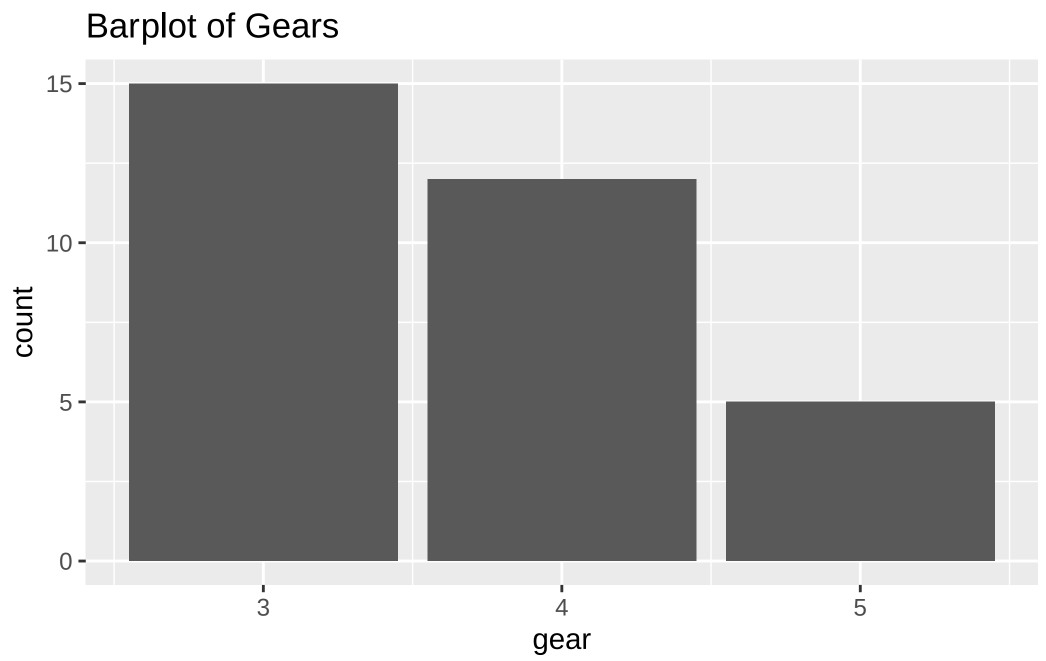
```r
library(ggplot2)
ggplot(mtcars, aes(mpg))+
  geom_histogram(bins=20)+
  ggtitle("Histogram of MPG")
```



```
ggplot(mtcars, aes(mpg))+
  geom_density()+
  ggtitle("Distribution of MPG")
```



```
ggplot(mtcars, aes(gear))+
  geom_bar(stat="count")+
  ggtitle("Barplot of Gears")
```


Barplot of Gears
count
15
10
5
0
3
4
5
gear

3 The Math

This section will provide an overview of a selection of mathematical skills that will be useful in the course. It will not go into everything, but at least provides a start.

MLE will include a range of concepts and skills from linear algebra, calculus, and probability and statistics. You do not need to be an expert in these to succeed in the course, but there are some fundamentals that will make your life easier.

Again, the end goal here is not the math. The end goal is to help your become great social scientists. Some popular methods in social science rely on these concepts. The more you can build an intuition for how, when, where, and why these methods work, the more tools you will have to carry out your research and the more credibility and command you will have over the research you conduct.

Think about restaurants you have visited. There is a difference between a server who says, "Well we have a chicken" and a server who can tell you exactly how the chicken was cooked, what ingredients were used in the sauce, and maybe even how the chicken was raised. Both dishes may be delicious, but you may gain more trust and understanding from the second.

This will not happen overnight nor by the end of the course, but the goal is to continue to progress in understanding. If you pursue a career in academia, this is only the start of years of continued learning and practice. It is a marathon, not a sprint.

Extra Resources

These are some additional resources that may help supplement the sections to follow. These provide far more detail than you will need in the course, but because of that, are much more comprehensive than the notes that will be outlined:

- A linear algebra web series. The notes on vector and matrices here will be brief, so this playlist 3Blue1Brown goes into more detail on specific. topics
- An alternative way of thinking about vectors and matrices is here which gives a more geometric interpretation, which some people find helpful.
- Will Moore and David Siegel. *A Mathematics Course for Political and Social Research.* You can get the book or visit the website, which includes some problem sets and solutions, as well as a video syllabus.

3.1 Mathematical Operations

In this first section, we will review mathematical operations that you have probably encountered before. In many cases, this will be a refresher on the rules and how to read notation.

3.1.1 Order of Operations

Many of you may have learned the phrase, "Please Excuse My Dear Aunt Sally" which stands for Parentheses (and other grouping symbols), followed by Exponents, followed by Multiplication and Division from left to right, followed by Addition and Subtraction from left to right.

There will be many equations in our future, and we must remember these rules.

- Example: $((1+2)^3)^2 = (3^3)^2 = 27^2 = 729$

To get the answer, we focused on respecting the parentheses first, identifying the inner-most expression $1+2=3$, we then moved out and conducted the exponents to get to $(3^3)^2 = 27^2$.

Note how this is different from the answer to $1 + (2^3)^2 = 1 + 8^2 = 65$, where the addition is no longer part of the parentheses.

3.1.2 Exponents

Here is a cheat sheet of some basic rules for exponents. These can be hard to remember if you haven't used them in a long time. Think of a in this case as a number, e.g., 4, and b, k, and l, as other numbers.

- $a^0 = 1$
- $a^1 = a$
- $a^k * a^l = a^{k+l}$
- $(a^k)^l = a^{kl}$
- $(\frac{a}{b})^k = (\frac{a^k}{b^k})$

These last two rules can be somewhat tricky. Note that a negative exponent can be re-written as a fraction. Likewise an exponent that is a fraction, the most common of which we will encounter is $\frac{1}{2}$ can be re-written as a root, in this case the square root (e.g., $\sqrt{a}$).

- $a^{-k} = \frac{1}{a^k}$
- $a^{1/2} = \sqrt{a}$

3.1.3 Summations and Products

The symbol $\sum$ can be read "take the sum of" whatever is to the right of the symbol. This is used to make the written computation of a sum much shorter than it might be otherwise. For example, instead of writing the addition operations separately in the example below, we can simplify it with the $\sum$ symbol. This is especially helpful if you would need to add together 100 or 1000 or more things. We will see these appear a lot in the course, for better or worse, so getting comfortable with the notation will be useful.

Usually, there is notation just below and just above the symbol (e.g., $\sum_{i=1}^{3}$). This can be read as "take the sum of the following from $i = 1$ to $i = 3$. We perform the operation in the expression, each time changing i to a different number, from 1 to 2 to 3. We then add each expression's output together.

- Example: $\sum_{i=1}^{3}(i+1)^2 = (1+1)^2 + (2+1)^2 + (3+1)^2 = 29$

We will also encounter the product symbol in this course: $\prod$. This is similar to the summation symbol, but this time we are multiplying instead of adding.

- Example: $\prod_{k=1}^{3} k^2 = 1^2 \times 2^2 \times 3^2 = 36$

3.1.4 Logarithms

In this class, we will generally assume that log takes the natural base e, which is a mathematical constant equal to 2.718…. In other books, the base might be 10 by default.

- If we have, $\log_{10} x = 2$, this is like saying 10^2 = 100.
- With base e, we have $\log_e x = 2$, which is $e^2 = 7.389$.
- We are just going to write $\log_e$ as log but know that the e is there.

A key part of maximum likelihood estimation is writing down the *log* of the likelihood equation, so this is a must-have for later in the course.

Here is a cheat sheet of common rules for working with logarithms.

- $\log x = 8 \rightarrow e^8 = x$
- $e^\pi = y \rightarrow \log y = \pi$
- $\log(a \times b) = \log a + \log b$
- $\log a^n = n \log a$
- $\log \frac{a}{b} = \log a - \log b$

Why logarithms? There are many different reasons why social scientists use logs.

- Some social phenomena grow exponentially, and logs make it is easier to visualize exponential growth, as is the case in visualizing the growth in COVID cases. See this example.
- Relatedly, taking the log of a distribution that is skewed, will make it look more normal or symmetrical, which has some nice properties.
- Sometimes the rules of logarithms are more convenient than non-logarithms. In MLE, we will take particular advantage of this rule: $\log(a \times b) = \log a + \log b$, which turns a multiplication problem into an addition problem.

3.2 Mathematical Operations in R

We will use R for this course, and these operations are all available with R code, allowing R to become a calculator for you. Here are some examples applying the tools above.

3.2.1 PEMDAS

```
((1+2)^3)^2
```

```
[1] 729
```

```
1 + (2^3)^2
```

```
[1] 65
```

3.2.2 Exponents

You can compare the computation below to match with the rules above. Note how the caret ^ symbol is used for exponents, and the asterisk * is used for multiplication. We also have a function `sqrt()` for taking the square root.

```
## Let's say a = 4 for our purposes
a <- 4
## And let's say k= 3 and l=5, b=2
k <- 3
l <- 5
b <- 2
```

- $a^0 = 1$

- $a^1 = a$

```
a^0
a^1
```

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

Note how we use parentheses in R to make it clear that the exponent includes not just `k` but `(k + l)`

- $a^k * a^l = a^{k+l}$

```
a^k * a^l
a^(k + l)
```

```
[1] 65536
[1] 65536
```

Note how we use the asterisk to make it clear we want to multiply `k*l`

- $(a^k)^l = a^{kl}$

```
(a^k)^l
a^(k*l)
```

```
[1] 1073741824
[1] 1073741824
```

- $(\frac{a}{b})^k = (\frac{a^k}{b^k})$

```
(a / b)^k
(a ^k)/(b^k)
```

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

- $a^{-k} = \frac{1}{a^k}$

```r
a^(-k)
1 / (a^k)
```

```
[1] 0.015625
[1] 0.015625
```

- $a^{1/2} = \sqrt{a}$

```r
a^(1/2)
sqrt(a)
```

```
[1] 2
[1] 2
```

3.2.3 Summations

Summations and products are a little more nuanced in R, depending on what you want to accomplish. But here is one example.

Let's take a vector (think a list of numbers) that goes from 1 to 4. We will call it `ourlist`.

```r
ourlist <- c(1,2,3,4)
ourlist
## An alternative is to write this: ourlist <- 1:4
```

```
[1] 1 2 3 4
```

Now let's do $\sum_{i=1}^{4}(i+1)^2 = (1+1)^2 + (1+2)^2 + (1+3)^2 + (1+4)^2 = 54$

In R, when you add a number to a vector, it will add that number to each entry in the vector. Example

```r
ourlist + 1
```

```
[1] 2 3 4 5
```

We can use that now to do the inside part of the summation. Note: the exponent works the same way, squaring each element of the inside expression.

```r
(ourlist + 1)^2
```

```
[1]  4  9 16 25
```

Now, we will embed this expression inside a function in R called `sum` standing for summation. It will add together each of the inside components.

```r
sum((ourlist + 1)^2)
```

```
[1] 54
```

If instead we wanted the product, multiplying each element of the inside together, we could use `prod()`. We won't use that function very much in this course.

3.2.4 Logarithms

R also has functions related to logarithms, called `log()` and `exp()` for the for the natural base e. By default, the natural exponential is the base in the `log()` R function.

- $\log x = 8 \rightarrow e^8 = x$

```r
exp(8)
```

```
[1] 2980.958
```

```r
log(2980.958)
```

```
[1] 8
```

Note that R also has a number of built-in constants, like `pi`.

- $e^\pi = y \rightarrow \log y = \pi$

```r
exp(pi)
```

```
[1] 23.14069
```

```r
log(23.14069)
```

```
[1] 3.141593
```

- $\log(a \times b) = \log a + \log b$

```r
log(a * b)
log(a) + log(b)
```

```
[1] 2.079442
[1] 2.079442
```

Let's treat $n = 3$ in this example and enter 3 directly where we see n below. Alternatively you could store 3 as n, as we did with the other letters above.

- $\log a^n = n \log a$

```r
log(a ^ 3)

3 * log(a)
```

```
[1] 4.158883
[1] 4.158883
```

- $\log \frac{a}{b} = \log a - \log b$

```r
log(a/b)
log(a) - log(b)
```

```
[1] 0.6931472
[1] 0.6931472
```

3.3 Derivatives

We will need to take some derivatives in the course. The reason is because a derivative gets us closer to understanding how to minimize and maximize certain functions, where a function is a relationship that maps elements of a set of inputs into a set of outputs, where each input is related to one output.

This is useful in social science, with methods such as linear regression and maximum likelihood estimation because it helps us estimate the values that we think will best describe the relationship between our independent variables and a dependent variable.

For example, in ordinary least squares (linear regression), we choose coefficients, which describe the relationship between the independent and dependent variables (*for every 1 unit change in x, we estimate* $\hat{\beta}$ amount of change in y), based on a method that tries to *minimize* the squared error between our estimated outcomes and the actual outcomes. In MLE, we will have a different quantity, which we will try to maximize.

To understand derivatives, we will briefly define limits.

Limits

A limit describes how a function behaves as it approaches (gets very close to) a certain value

- $\lim_{x \to a} f(x) = L$

Example: $\lim_{x \to 3} x^2 = 9$ The limit of this function as x approaches three, is 9. Limits will appear in the expression for calculating derivatives.

3.3.1 Derivatives

For intuition on a derivative, watch this video from The Math Sorcerer.

A derivative is the instantaneous rate at which the function is changing at x: *the slope of a function at a particular point.*

There are different notations for indicating something is a derivative. Below, we use $f'(x)$ because we write our functions as $f(x) = x$. Many times you might see a function equation like $y = 3x$ There, it will be common for the derivative to be written like $\frac{dy}{dx}$.

Let's break down the definition of a derivative by looking at its similarity to the simple definition of a slope, as the rise over the run:

- Slope (on average): rise over run: change in $f(x)$ over an interval ($[c, b]$ where $b - c = h$): $\frac{f(b)-f(c)}{b-c}$

For slope at a specific point x (the derivative of f(x) at x), we just make the interval h very small:

- $f'(x) = \lim_{h \to 0} \frac{f(a+h)-f(a)}{h}$

Example $f(x) = 2x + 3$.

- $f'(x) = \lim_{h \to 0} \frac{2(x+h)+3-(2x+3)}{h} = \lim_{h \to 0} \frac{2x+2h-2x}{h} = \lim_{h \to 0} 2 = 2$

This twitter thread by the brilliant teacher and statistician Allison Horst, provides a nice cartoon-based example of the derivative. (Note that sometimes the interval h is written as Δx, the change in x).

3.3.2 Critical Points for Minima or Maxima

In both OLS and MLE, we reach points where take what are called "first order conditions." This means we take the derivative with respect to a parameter of interest and then set the derivative = 0 and solve for the parameter to get an expression for our estimator. (E.g., In OLS, we take the derivative of the sum of squared residuals, set it equal to zero, and solve to get an expression for $\hat{\beta}$).

The reason we are interested in when the derivative is zero, is because this is when the instanaeous rate of change is zero, i.e., the slope at a particular point is zero. When does this happen? At a critical point- maximum or minimum. Think about it– at the top of a mountain, there is no more rise (and no decline). You are completing level on the mountaintop. The slope at that point is zero.

Let's take an example. The function $f(x) = x^2 + 1$ has the derivative $f'(x) = 2x$. This is zero when $x = 0$.

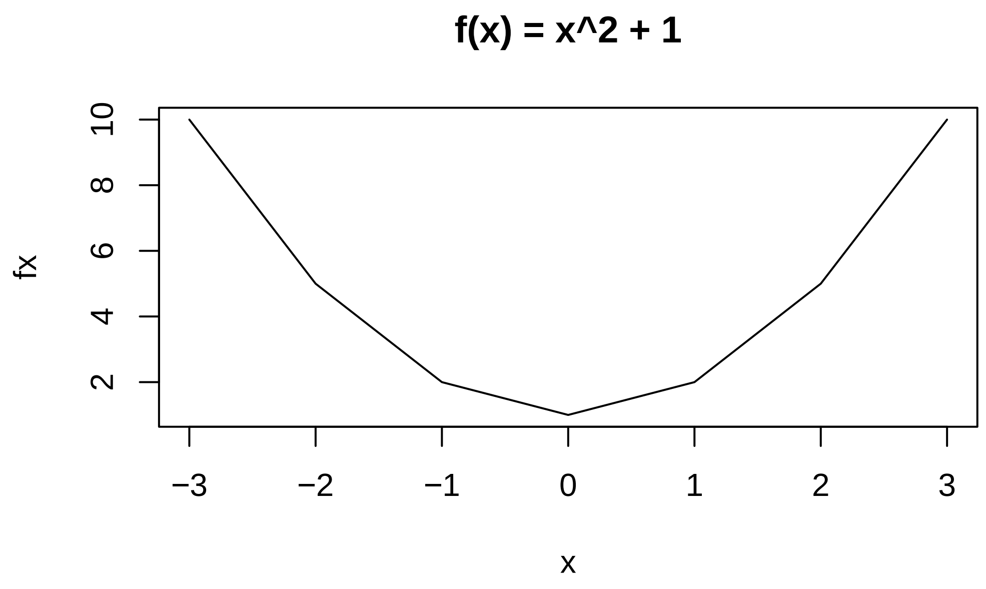
The question remains: How do we know if it is a maximum or minimum?

We need to figure out if our function is concave or convex around this critical value. Convex is a "U" shape, meaning we are at a minimum, while concavity is an upside-down-U, which means we are at a maximum. We do so by taking the second derivative. This just means we take the derivative of the expression we already have for our first derivative. In our case, $f''(x) = 2$. So what? Well the key thing we are looking for is if this result is positive or negative. Here, it is positive, which means our function is convex at this critical value, and therefore, we are at a minimum.

Just look at the function in R if we plot it.

```r
## Let's define an arbitrary set of values for x
x <- -3:3
## Now let's map the elements of x into y using our function
fx <- x^2 + 1

## Let's plot the results
```

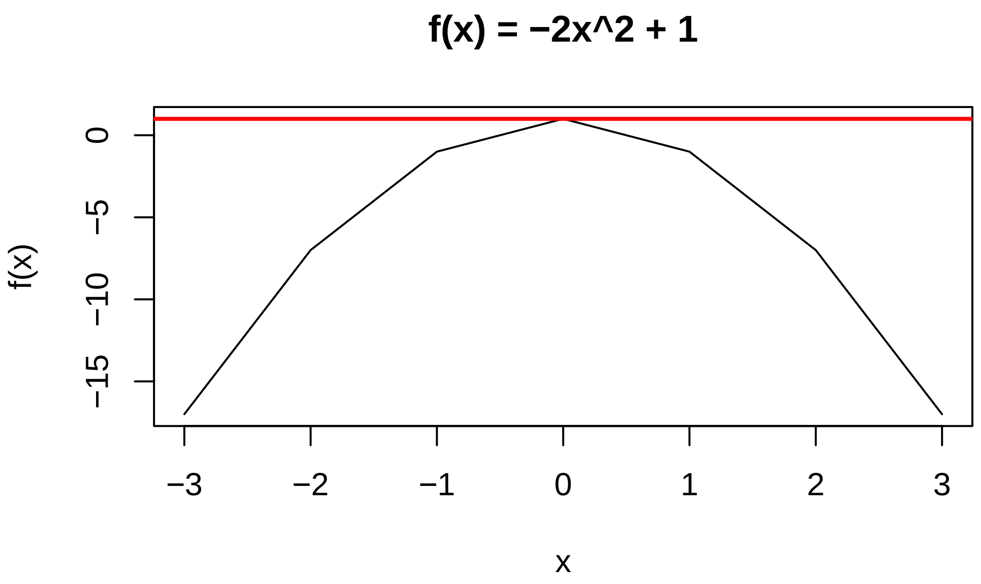
```
plot(x = x, y=fx,
     xlab = "x", type = "l",
     main = "f(x) = x^2 + 1")
```



Notice that when x=0, we are indeed at a minimum, just as the positive value of the second derivative would suggest.

A different example: $f(x) = -2x^2 + 1$. $f'(x) = -4x$ When we set this equal to 0 we find a critical value at $x = 0$. $f''(x) = -4$. Here, the value is negative, and we know it is concave. Sure enough, let's plot it, and notice how we can draw a horiztonal line at the maximum, representing that zero slope at the critical point:

```
x <- -3:3
fx <- -2*x^2 + 1
plot(x = x, y = fx,
    ylab = "f(x)",
     xlab = "x", type = "l",
     main = "f(x) = -2x^2 + 1")
abline(h=1, col = "red", lwd=2)
```



3.3.3 Common Derivative Rules

Below is a cheat sheet of rules for quickly identifying the derivatives of functions.

The derivative of a constant is 0.

- $f(x) = a; f'(x) = 0$
  - Example: The derivative of 5 is 0.

Here is the power rule.

- $f(x) = ax^n; f'(x) = n \times a \times x^{n-1}$
  - Example: The derivative of $x^3 = 3x^{(3-1)} = 3x^2$

We saw logs in the last section, and, yes, we see logs again here.

- $f(x) = e^{ax}; f'(x) = ae^{ax}$
- $f(x) = \log(x); f'(x) = \frac{1}{x}$

A very convenient rule is that a derivative of a sum = sum of the derivatives.

- $f(x) = g(x) + h(x); f'(x) = g'(x) + h'(x)$

Products can be more of a headache. In this course, we will turn some product expressions into summation expressions to avoid the difficulties of taking derivatives with products.

- Product Rule: $f(x) = g(x)h(x); f'(x) = g'(x)h(x) + h'(x)g(x)$

The chain rule below looks a bit tricky, but it can be very helpful for simplifying the way you take a derivative. See this video from NancyPi for a helpful explainer, as well as a follow-up for more complex applications here.

- Chain Rule: $f(x) = g(h(x)); f'(x) = g'(h(x))h'(x)$

Example: What is the derivative of $f(x) = \log 5x$?

- First, we will apply the rule which tells us the derivative of a $\log x$ is $\frac{1}{x}$.
- However, here, we do not just have x, we have $5x$. We are in chain rule territory.
- After we apply the derivative to the log, which is $\frac{1}{5x}$, we then have to take the derivative of $5x$ and multiply the two expressions together.
- The derivative of $5x$ is 5.
- So, putting this together, our full derivative is f (x) $= 5 * \frac{1}{5x} = \frac{1}{x}$.

3.4 Vectors and Matrices

Vectors

For our purposes, a vector is a list or "array" of numbers. For example, this might be a variable in our data– a list of the ages of all politicians in a country.

Addition

- If we have two vectors **u** and **v**, where
  - $\mathbf{u} = (u_1, u_2, \dots u_n)$ and
  - $\mathbf{v} = (v_1, v_2, \dots v_n)$,
  - $\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2, \dots u_n + v_n)$
- Note: **u** and **v** must be of the same dimensionality - number of elements in each must be the same - for addition.

Scalar multiplication

- If we have a scalar (i.e., a single number) λ and a vector **u**
- $\lambda\mathbf{u} = (\lambda u_1, \lambda u_2, \dots \lambda u_n)$

We can implement vector addition and scalar multiplication in R.

Let's create a vector **u**, a vector **v**, and a number lambda.

```r
u <- c(33, 44, 22, 11)
v <- c(6, 7, 8, 2)
lambda <- 3
```

When you add two vectors in R, it adds each component together.

```r
u + v
```

```
[1] 39 51 30 13
```

We can multiply each element of a vector, `u` by `lambda`:

```r
lambda * u
```

```
[1]  99 132  66  33
```

Element-wise Multiplication

Note: When you multiply two vectors together in R, it will take each element of one vector and multiply it by each element of the other vector.

- `u * v` $= (u_1 * v_1, u_2 * v_2, ... u_n * v_n)$

```r
u * v
```

```
[1] 198 308 176  22
```

3.4.1 Matrix Basics

A matrix represents arrays of numbers in a rectangle, with rows and columns.

- A matrix with m rows and n columns is defined as (m x n). What is the dimensionality of the matrix **A** below?

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix}$$

In R, we can think of a matrix as a set of vectors. For example, we could combine the vectors `u` and `v` we created above into a matrix defined as W.

```
## cbind() binds together vectors as columns
Wcol <- cbind(u, v)
Wcol
```

```
      u v
[1,] 33 6
[2,] 44 7
[3,] 22 8
[4,] 11 2
```

```
## rbind() binds together vectors as rows
Wrow <- rbind(u, v)
Wrow
```

```
  [,1] [,2] [,3] [,4]
u   33   44   22   11
v    6    7    8    2
```

There are other ways to create matrices in R, but using `cbind` and `rbind()` are common.

We can find the dimensions of our matrices using `dim()` or `nrow()` and `ncol()` together. For example:

```
dim(Wcol)
```

```
[1] 4 2
```

```
nrow(Wcol)
```

```
[1] 4
```

```
ncol(Wcol)
```

```
[1] 2
```

Note how the dimensions are different from the version created with `rbind()`:

```r
dim(Wrow)
```

```
[1] 2 4
```

```r
nrow(Wrow)
```

```
[1] 2
```

```r
ncol(Wrow)
```

```
[1] 4
```

Extracting specific components

The element a_{ij} signifies the element is in the ith row and jth column of matrix A. For example, a_{12} is in the first row and second column.

- Square matrices have the same number of rows and columns
- Vectors have just one row or one column (e.g., x_1 element of $\mathbf{x}$ vector)

In R, we can use brackets to extract a specific ij element of a matrix or vector.

```r
Wcol
```

```
      u v
[1,] 33 6
[2,] 44 7
[3,] 22 8
[4,] 11 2
```

```r
Wcol[2,1] # element in the second row, first column
```

```
 u 
44 
```

```r
Wcol[2,] # all elements in the second row
```

```
 u  v 
44  7 
```

```r
Wcol[, 1] # all elements in the first column
```

```
[1] 33 44 22 11
```

For matrices, to extract a particular entry in R, you have a comma between entries because there are both rows and columns. For vectors, you only have one entry, so no comma is needed.

```r
u
```

```
[1] 33 44 22 11
```

```r
u[2] # second element in the u vector
```

```
[1] 44
```

3.4.2 Matrix Operations

Matrix Addition

- To be able to add matrix **A** and matrix **B**, they must have the same dimensions.
- Like vector addition, to add matrices, you add each of the components together.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \text{ and } B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$$

$$A + B = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} \end{pmatrix}$$

$$Q = \begin{pmatrix} 2 & 4 & 1 \\ 6 & 1 & 5 \end{pmatrix} + R = \begin{pmatrix} 9 & 4 & 2 \\ 11 & 8 & 7 \end{pmatrix} = Q + R = \begin{pmatrix} 11 & 8 & 3 \\ 17 & 9 & 12 \end{pmatrix}$$

Scalar Multiplication

Take a scalar ν. Just like vectors, we multiply each component of a matrix by the scalar.

$$\nu Q = \begin{pmatrix} \nu q_{11} & \nu q_{12} & \dots & \nu q_{1n} \\ \nu q_{21} & \nu q_{22} & \dots & \nu q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \nu q_{m1} & \nu q_{m2} & \dots & \nu q_{mn} \end{pmatrix}$$

Example: Take $c = 2$ and a matrix A.

$$cA = c * \begin{pmatrix} 4 & 6 & 1 \\ 3 & 2 & 8 \end{pmatrix} = \begin{pmatrix} 8 & 12 & 2 \\ 6 & 4 & 16 \end{pmatrix}$$

Note the Commutativity/Associativity: For scalar c: $c(AB) = (cA)B = A(cB) = (AB)c$.

Matrix Multiplication

A matrix A and B must be conformable to multiply AB.

- To be comformable, for m_A x n_A matrix A and m_B x n_B matrix B, the "inside" dimensions must be equal: $n_A = m_B$.
- The resulting AB has the "outside" dimensions: m_A x n_B.

For each c_{ij} component of $C = AB$, we take the inner product of the i^{th} row of matrix A and the j^{th} column of matrix B.

- Their product C = AB is the m x n matrix where:
- $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ik}b_{kj}$

Example: This 2×3 matrix is multiplied by a 3×2 matrix, resulting in the 2×2 matrix.

$$\begin{pmatrix} 4 & 6 & 1 \\ 3 & 2 & 8 \end{pmatrix} \times \begin{pmatrix} 8 & 12 \\ 6 & 4 \\ 7 & 10 \end{pmatrix} = \begin{pmatrix} (4*8+6*6+1*7) & (4*12+6*4+1*10) \\ (3*8+2*6+8*7) & (3*12+2*4+8*10) \end{pmatrix}$$

For example, the entry in the first row and second column of the new matrix $c_{12} = (a_{11} = 4 * b_{11} = 12) + (a_{12} = 6 * b_{21} = 4) + (a_{13} = 1 * b_{31} = 10)$

We can also do matrix multiplication in R.

```r
## Create a 3 x 2 matrix A
A <- cbind(c(3, 4, 6), c(5, 6, 8))
A
```

```
     [,1] [,2]
[1,]    3    5
[2,]    4    6
[3,]    6    8
```

```r
## Create a 2 x 4 matrix B
B <- cbind(c(6,8), c(7, 9), c(3, 6), c(1, 11))
B
```

```
     [,1] [,2] [,3] [,4]
[1,]    6    7    3    1
[2,]    8    9    6   11
```

Note that the multiplication AB is conformable because the number of columns in A matches the number of rows in B:

```
ncol(A)
nrow(B)
```

```
[1] 2
[1] 2
```

To multiply matrices together in R, we need to add symbols around the standard asterisk for multiplication:

```
A %*% B
```

```
     [,1] [,2] [,3] [,4]
[1,]   58   66   39   58
[2,]   72   82   48   70
[3,]  100  114   66   94
```

That is necessary for multiplying matrices together. It is not necessary for scalar multiplication, where we take a single number (e.g., c = 3) and multiply it with a matrix:

```
c <- 3
c*A
```

```
     [,1] [,2]
[1,]    9   15
[2,]   12   18
[3,]   18   24
```

Note the equivalence of the below expressions, which combine scalar and matrix multiplication:

```
c* (A %*% B)
```

```
     [,1] [,2] [,3] [,4]
[1,]  174  198  117  174
[2,]  216  246  144  210
[3,]  300  342  198  282
```

```
(c* A) %*% B
```

```
     [,1] [,2] [,3] [,4]
[1,]  174  198  117  174
[2,]  216  246  144  210
[3,]  300  342  198  282
```

```
A %*% (c * B)
```

```
     [,1] [,2] [,3] [,4]
[1,]  174  198  117  174
[2,]  216  246  144  210
[3,]  300  342  198  282
```

In social science, one matrix of interest is often a rectangular dataset that includes column vectors representing independent variables, as well as another vector that includes your dependent variable. These might have 1000 or more rows and a handful of columns you care about.

3.5 Additional Matrix Tidbits that Will Come Up

Inverse

An n x n matrix A is invertible if there exists an n x n inverse matrix A^{-1} such that:

- $AA^{-1} = A^{-1}A = I_n$
- where I_n is the identity matrix (n x n), that takes diagonal elements of 1 and off-diagonal elements of 0. Example:

$$I_n = \begin{pmatrix} 1_{11} & 0 & ... & 0 \\ 0 & 1_{22} & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1_{nn} \end{pmatrix}$$

- Multiplying a matrix by the identity matrix returns in the matrix itself: $AI_n = A$

  – It's like the matrix version of multiplying a number by one.

Note: A matrix must be square n x n to be invertible. (But not all square matrices are invertible.) A matrix is invertible if and only if its columns are linearly independent. This is important for understanding why you cannot have two perfectly colinear variables in a regression model.

We will not do much solving for inverses in this course. However, the inverse will be useful in solving for and simplifying expressions.

3.5.1 Transpose

When we transpose a matrix, we flip the i and j components.

- Example: Take a 4 X 3 matrix A and find the 3 X 4 matrix A^T.
- A transpose is usually denoted with as A^T or A'

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix} \text{ then } A^T = \begin{pmatrix} a'_{11} & a'_{12} & a'_{13} & a'_{14} \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} \\ a'_{31} & a'_{32} & a'_{33} & a'_{34} \end{pmatrix}$$

$$\text{If } A = \begin{pmatrix} 1 & 4 & 2 \\ 3 & 1 & 11 \\ 5 & 9 & 4 \\ 2 & 11 & 4 \end{pmatrix} \text{ then } A^T = \begin{pmatrix} 1 & 3 & 5 & 2 \\ 4 & 1 & 9 & 11 \\ 2 & 11 & 4 & 4 \end{pmatrix}$$

Check for yourself: What was in the first row ($i = 1$), second column ($j = 2$) is now in the second row ($i = 2$), first column ($j = 1$). That is $a_{12} = 4 = a'_{21}$.

We can transpose matrices in R using `t()`. For example, take our matrix `A`:

```
A
```

```
     [,1] [,2]
[1,]    3    5
[2,]    4    6
[3,]    6    8
```

```
t(A)
```

```
     [,1] [,2] [,3]
[1,]    3    4    6
[2,]    5    6    8
```

In R, you can find the inverse of a square matrix with `solve()`

```r
solve(A)
```

```
Error in solve.default(A): 'a' (3 x 2) must be square
```

Note, while `A` is not square A'A is square:

```r
AtA <- t(A) %*% A

solve(AtA)
```

```
          [,1]      [,2]
[1,]  2.232143 -1.553571
[2,] -1.553571  1.089286
```

3.5.2 Additional Matrix Properties and Rules

These are a few additional properties and rules that will be useful to us at various points in the course:

- Symmetric: Matrix A is symmetric if $A = A^T$
- Idempotent: Matrix A is idempotent if $A^2 = A$
- Trace: The trace of a matrix is the sum of its diagonal components $Tr(A) = a_{11} + a_{22} + \dots + a_{mn}$

Example of symmetric matrix:

$$D = \begin{pmatrix} 1 & 6 & 22 \\ 6 & 4 & 7 \\ 22 & 7 & 11 \end{pmatrix}$$

```r
## Look at the equivalence
D <- rbind(c(1,6,22), c(6,4,7), c(22,7,11))
D
```

```
     [,1] [,2] [,3]
[1,]    1    6   22
[2,]    6    4    7
[3,]   22    7   11
```

```
t(D)
```

```
     [,1] [,2] [,3]
[1,]    1    6   22
[2,]    6    4    7
[3,]   22    7   11
```

What is the trace of this matrix?

```
## diag() pulls out the diagonal of a matrix
sum(diag(D))
```

```
[1] 16
```

3.5.3 Matrix Rules

Due to conformability and other considerations, matrix operations are somewhat more restrictive, particularly when it comes to commutativity.

- Associative $(A + B) + C = A + (B + C)$ and $(AB)C = A(BC)$
- Commutative $A + B = B + A$
- Distributive $A(B + C) = AB + AC$ and $(A + B)C = AC + BC$
- Commutative law for multiplication does not hold– the order of multiplication matters: $ AB BA$

Rules for Inverses and Transposes

These rules will be helpful for simplifying expressions. Treat A, B, and C as matrices below, and s as a scalar.

- $(A + B)^T = A^T + B^T$
- $(sA)^T = sA^T$
- $(AB)^T = B^T A^T$
- $(A^T)^T = A$ and $(A^{-1})^{-1} = A$
- $(A^T)^{-1} = (A^{-1})^T$
- $(AB)^{-1} = B^{-1}A^{-1}$
- $(ABCD)^{-1} = D^{-1}C^{-1}B^{-1}A^{-1}$

3.5.4 Derivatives with Matrices and Vectors

Let's say we have a $p \times 1$ "column" vector $\mathbf{x}$ and another $p \times 1$ vector $\mathbf{a}$.

Taking the derivative with respect to vector $\mathbf{x}$.

Let's say we have $y = \mathbf{x}'\mathbf{a}$. This process is explained here. Taking the derivative of this is called the gradient.

- $\frac{\delta y}{\delta x} = \begin{pmatrix} \frac{\delta y}{\delta x_1} \\ \frac{\delta y}{\delta x_2} \\ \vdots \\ \frac{dy}{dx_p} \end{pmatrix}$
- y will have dimensions 1×1. y is a scalar.
  - Note: $y = a_1x_1 + a_2x_2 + ... + a_px_p$. From this expression, we can take a set of "partial derivatives":
  - $\frac{\delta y}{\delta x_1} = a_1$
  - $\frac{\delta y}{\delta x_2} = a_2$, and so on
  - $\frac{\delta y}{\delta x} = \begin{pmatrix} \frac{\delta y}{\delta x_1} \\ \frac{\delta y}{\delta x_2} \\ \vdots \\ \frac{\delta y}{\delta x_p} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$
- Well, this is just vector $\mathbf{a}$

Answer: $\frac{\delta}{\delta x}\mathbf{x}^T\mathbf{a} = \mathbf{a}$. We can apply this general rule in other situations.

Example 2

Let's say we want to differentiate the following where vector $\mathbf{y}$ is $n \times 1$, X is $n \times k$, and $\mathbf{b}$ is $k \times 1$. Take the derivative with respect to b.

- $\mathbf{y}'\mathbf{y} - 2\mathbf{b}'X'\mathbf{y}$
- Note that the dimensions of the output are 1×1, a scalar quantity.

Remember the derivative of a sum is the sum of derivatives. This allows us to focus on particular terms.

- The first term has no $\mathbf{b}$ in it, so this will contribute 0.
- The second term is $2\mathbf{b}'X'\mathbf{y}$. We can think about this like the previous example

  - $\frac{\delta}{\delta b}2\mathbf{b}'X'\mathbf{y} = \begin{pmatrix} \frac{\delta}{\delta b_1} \\ \frac{\delta}{\delta b_2} \\ \vdots \\ \frac{\delta}{\delta b_k} \end{pmatrix}$
  - The output is needs to be $k \times 1$ like $\mathbf{b}$, which is what $2 * X'\mathbf{y}$ is.

- The derivative is $-2X'\mathbf{y}$

Example 3

Another useful rule when a matrix A is symmetric: $\frac{\delta}{\delta \mathbf{x}}\mathbf{x}^T A\mathbf{x} = (A + A^T)\mathbf{x} = 2A\mathbf{x}$.

Details on getting to this result. We are treating the vector $\mathbf{x}$ as $n \times 1$ and the matrix A as symmetric.

When we take $\frac{\delta}{\delta \mathbf{x}}$ (the derivative with respect to $\mathbf{x}$), we will be looking for a result with the same dimensions $\mathbf{x}$.

$\frac{\delta}{\delta \mathbf{x}} = \begin{pmatrix} \frac{\delta}{\delta x_1} \\ \frac{\delta}{\delta x_2} \\ \vdots \\ \frac{d}{dx_n} \end{pmatrix}$

Let's inspect the dimensions of $\mathbf{x}^T A\mathbf{x}$. They are 1×1. If we perform this matrix multiplication, we would be multiplying:

$$\begin{pmatrix} x_1 & x_2 & ... & x_i \end{pmatrix} \times \begin{pmatrix} a_{11} & a_{12} & ... & a_{1j} \\ a_{21} & a_{22} & ... & a_{2j} \\ \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & ... & a_{ij} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \end{pmatrix}$$

To simplify things, let's say we have the following matrices, where A is symmetric:

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix} \times \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

We can perform the matrix multiplication for the first two quantities, which will result in a 1×2 vector. Recall in matrix multiplication we take the sum of the element-wise multiplication of the *ith* row of the first object by the *jth* column of the second object. This means multiply the first row of $\mathbf{x}$ by the first column of A for the entry in cell $i = 1; j = 1$, and so on.

$$\begin{pmatrix} (x_1 a_{11} + x_2 a_{21}) & (x_1 a_{12} + x_2 a_{22}) \end{pmatrix}$$

We can then multiply this quantity by the last quantity

$$\begin{pmatrix} (x_1 a_{11} + x_2 a_{21}) & (x_1 a_{12} + x_2 a_{22}) \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

This will results in the 1×1 quantity: $(x_1 a_{11} + x_2 a_{21})x_1 + (x_1 a_{12} + x_2 a_{22})x_2 = x_1^2 a_{11} + x_2 a_{21} x_1 + x_1 a_{12} x_2 + x_2^2 a_{22}$

We can now take the derivatives with respect to $\mathbf{x}$. Because $\mathbf{x}$ is 2×1, our derivative will be a vector of the same dimensions with components:

$\frac{\delta}{\delta \mathbf{x}} = \begin{pmatrix} \frac{\delta}{\delta x_1} \\ \frac{\delta}{\delta x_2} \end{pmatrix}$

These represent the partial derivatives of each component within $\mathbf{x}$

Let's focus on the first: $\frac{\delta}{\delta x_1}$.

$$\begin{aligned} \frac{\delta}{\delta x_1} x_1^2 a_{11} + x_2 a_{21} x_1 + x_1 a_{12} x_2 + x_2^2 a_{22} = \\ = 2x_1 a_{11} + x_2 a_{21} + a_{12} x_2 \end{aligned}$$

We can repeat this for $\frac{\delta}{\delta x_2}$

$$\begin{aligned} \frac{\delta}{\delta x_2} x_1^2 a_{11} + x_2 a_{21} x_1 + x_1 a_{12} x_2 + x_2^2 a_{22} = \\ = a_{21} x_1 + x_1 a_{12} + 2x_2 a_{22} \end{aligned}$$

Now we can put the result back into our vector format:

$\begin{pmatrix} \frac{\delta}{\delta x_1} = 2x_1 a_{11} + x_2 a_{21} + a_{12} x_2 \\ \frac{\delta}{\delta x_2} = a_{21} x_1 + x_1 a_{12} + 2x_2 a_{22} \end{pmatrix}$

Now it's just about simplifying to show that we have indeed come back to the rule.

Recall that for a symmetric matrix, the elements in rows and columns ij = the elements in ji. This allows us to read $a_{21} = a_{12}$ and combine those terms (e.g., $x_2 a_{21} + a_{12} x_2 = 2a_{12} x_2$) :

$\begin{pmatrix} \frac{\delta}{\delta x_1} = 2x_1 a_{11} + 2a_{12} x_2 \\ \frac{\delta}{\delta x_2} = 2a_{21} x_1 + 2x_2 a_{22} \end{pmatrix}$

Second, we can now bring the 2 out front.

$\begin{pmatrix} \frac{\delta}{\delta x_1} \\ \frac{\delta}{\delta x_2} \end{pmatrix} = 2 * \begin{pmatrix} x_1 a_{11} + a_{12} x_2 \\ a_{21} x_1 + x_2 a_{22} \end{pmatrix}$

Finally, let's inspect this and show it is equivalent to this multiplication where we have a 2×2 A matrix multiplied by a 2×1 $\mathbf{x}$ vector. *Minor note: Because any individual element of a* *vector is just a single quantity, we can change the order (e.g., $a_{11} * x_1$ vs. $x_1 * a_{11}$). We just can't do that for full vectors or matrices*

$$2A\mathbf{x} = 2 * \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 2 * \begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \end{pmatrix}$$

The last quantity is the same as the previous step. That's the rule!

Applying the rules

- This has a nice analogue to the derivative we've seen before $q^2 = 2 * q$.
- Let's say we want to take the derivative of $\mathbf{b}'X'X\mathbf{b}$ with respect to $\mathbf{b}$.
- We can think of $X'X$ as if it is A.
  - This gives us $2X'X\mathbf{b}$ as the result.

Why on earth would we care about this? For one, it helps us understand how we get to our estimates for $\hat{\beta}$ in linear regression. When we have multiple variables, we don't just want the best estimate for one coefficient, but a vector of coefficients. See more here.

In MLE, we will find the gradient of the log likelihood function. We will further go into the second derivatives to arrive at what is called the Hessian. More on that later.

3.6 Practice Problems

1. What is $24/3 + 5^2 - (8 - 4)$?
2. What is $\sum_{i=1}^{5}(i * 3)$?
3. Take the derivative of $f(x) = v(4x^2 + 6)^2$ with respect to x.
4. Take the derivative of $f(x) = e^{2x+3}$ with respect to x.
5. Take the derivative of $f(x) = log(x + 3)^2$ with respect to x.

Given X is an n x k matrix,

6. $(X^TX)^{-1}X^TX$ can be simplified to?
7. $((X^TX)^{-1}X^T)^T = ?$
8. If ν is a constant, how does $(X^TX)^{-1}X^T\nu X(X^TX)^{-1}$ simplify?
9. If a matrix P is idempotent, $PP = ?$

3.6.1 Practice Problem Solutions

1. What is $24/3 + 5^2 - (8 - 4)$?

```
24/3 + 5^2 - (8 -4)
```

```
[1] 29
```

2. What is $\sum_{i=1}^{5}(i * 3)$?
   - By hand: $1 \times 3 + 2 \times 3 + 3 \times 3 + 4 \times 3 + 5 \times 3$

```
## sol 1
1*3 + 2*3 + 3*3 + 4*3 + 5*3
```

```
[1] 45
```

```
## sol 2
i <- 1:5
sum(i*3)
```

```
[1] 45
```

3. Take the derivative of $f(x) = v(4x^2 + 6)^2$ with respect to x.
   - We can treat v as a number.

$$\begin{aligned} f'(x) &= 2 * v(4x^2 + 6) * 8x \\ &= 16vx(4x^2 + 6) \end{aligned}$$

4. Take the derivative of $f(x) = e^{2x+3}$ with respect to x.

$$\begin{aligned} f'(x) &= 2 * e^{2x+3} \\ &= 2e^{2x+3} \end{aligned}$$

5. Take the derivative of $f(x) = log(x + 3)^2$ with respect to x.
   - Note we can re-write this as $2 * log(x + 3)$.

$$\begin{aligned} f'(x) &= 2 * \frac{1}{(x+3)} * 1 \\ &= \frac{2}{(x+3)} \end{aligned}$$

If we didn't take that simplifying step, we can still solve:

$$
\begin{aligned}
f'(x) &= \frac{1}{(x+3)^2} * 2 * (x+3) * 1 \\
&= \frac{2}{(x+3)}
\end{aligned}
$$

Given X is an n x k matrix,

6. $(X^T X)^{-1} X^T X$ can be simplified to?
   - I_k the identity matrix
7. $((X^T X)^{-1} X^T)^T = ?$
   - Recall our rule $(AB)^T = B^T A^T$
   - $X(X^T X)^{-1}$
8. If ν is a constant, how does $(X^T X)^{-1} X^T \nu X (X^T X)^{-1}$ simplify?
   - We can pull it out front.

$$
\begin{aligned}
&= \nu (X^T X)^{-1} X^T X (X^T X)^{-1} \\
&= \nu (X^T X)^{-1}
\end{aligned}
$$

9. If a matrix P is idempotent, $PP = ?$
   - P from section 3.5.2

4 Review of OLS

This section will provide a review of OLS.

OLS is the workhorse of empirical political science. We will learn a lot beyond OLS, but OLS is often "good enough" and sometimes preferable for explaining the relationship between variables. That is to say, MLE will expand your toolkit, but OLS should remain a big part of your toolkit.

I recommend that you review the following readings to familiarize yourself with regression. I will make note within this section where particular readings are most relevant. These readings are available on Canvas in the modules- Week 1 section.

- Wheelan, Charles. 2012. *Naked Statistics*. W.W. Norton. Chapter 11. This provides an accessible overview of regression and the interpretation of regression results.
- Gelman, Andrew, and Jennifer Hill. 2006. Data analysis using regression and multilevel/hierarchical models. Cambridge University Press. Chapter 3. This is a slightly more technical overview and includes some R code for running regressions.
- Building models and breaking models.
  - (Optional) Fox, John. 2015. Applied Regression Analysis and Generalized Linear Models, 2nd Edition. Sage. Chapter 11. This reading describes diagnostic tests to probe whether the model is a good fit of the data. We won't go into detail about this in this class, but is material classes focused on linear regression will generally cover.
  - Messing, Solomon. "How to break regression."
  - Lenz, G., & Sahn, A. (2020). "Achieving Statistical Significance with Control Variables and Without Transparency." *Political Analysis*, 1-14. doi:10.1017/pan.2020.31. This paper talks about how to build a regression model, and in particular, why adding more and more controls isn't always a good thing.

4.1 Introducing OLS Regression

The regression method describes how one variable depends on one or more other variables. Ordinary Least Squares regression is a linear model with the matrix representation:

$Y = \alpha + X\beta + \epsilon$

Given values of variables in X, the model predicts the average of an outcome variable Y. For example, if Y is a measure of how wealthy a country is, X may contain measures related to the country's natural resources and/or features of its institutions (things that we think might contribute to how wealthy a country is.) In this equation:

- Y is the outcome variable ($n \times 1$).[1]
- α is a parameter representing the intercept
- β is a parameter representing the slope/marginal effect ($k \times 1$), and
- ϵ is the error term ($n \times 1$).

In OLS, we estimate a line of best fit to predict $\hat{Y}$ values for different values of X:

- $\hat{Y} = \hat{\alpha} + X\hat{\beta}$.
- When you see a "$\hat{hat}$" on top of a letter, that means it is an estimate of a parameter.
- As we will see in the next section, in multiple regression, sometimes this equation is represented as just $\hat{Y} = X\hat{\beta}$, where this generally means that X is a matrix that includes several variables and $\hat{\beta}$ is a vector that includes several coefficients, including a coefficient representing the intercept $\hat{\alpha}$

We interpret linear regression coefficients as describing how a dependent variable is expected to change when a particular independent variable changes by a certain amount. Specifically:

- "Associated with each one unit increase in a variable x_1, there is a $\hat{\beta}_1$ estimated expected average increase in y."
- If we have more than one explanatory variable (i.e., a multiple regression), we add the phrase "controlling on/ holding constant other observed factors included in the model."

We can think of the interpretation of a coefficient in multiple regression using an analogy to a set of light switches:

We ask: How much does the light in the room change when we flip one switch, while holding constant the position of all the other switches?

[1]Recall this notation means rows by columns, Y is a vector of length n (the number of observations), and since there is only 1 outcome measure, it is 1 column.

This would be a good place to review the Wheelan chapter and Gelman and Hill 3.1 and 3.2 to reinforce what a regression is and how to interpret regression results.

4.2 Diving Deeper into OLS Matrix Representation

In this section, we will review the matrix representation of the OLS regression in more detail and discuss how to derive the estimators for the regression coefficients.[2]

OLS in Matrix Form: Let X be an $n \times k$ matrix where we have observations on k independent variables for n observations. Since our model will usually contain a constant term, one of the columns in the X matrix will contain only ones. This column should be treated exactly the same as any other column in the X matrix.

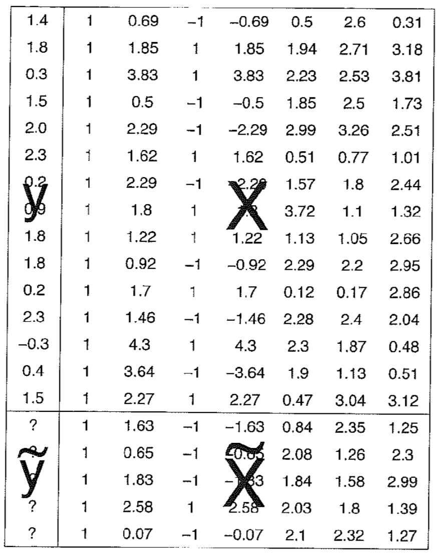
- Let Y be an $n \times 1$ vector of observations on the dependent variable. Note: because Y is a vector (a matrix with just one column), sometimes it is written in lowercase notation as $\mathbf{y}$.
- Let ϵ be an $n \times 1$ vector of disturbances or errors.
- Let β be an $k \times 1$ vector of unknown population parameters that we want to estimate.

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ ... \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & x_{13} & ... & x_{1k} \\ 1 & x_{21} & x_{22} & x_{23} & ... & x_{2k} \\ 1 & x_{31} & x_{32} & x_{33} & ... & x_{3k} \\ 1 & x_{41} & x_{42} & x_{43} & ... & x_{4k} \\ ... & ... & ... & ... & ... & ... \\ 1 & x_{n1} & x_{n2} & x_{n3} & ... & x_{nk} \end{pmatrix} \text{X} \begin{pmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ ... \\ \beta_k \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ ... \\ \epsilon_n \end{pmatrix}$$

Our estimates are then $\hat{\mathbf{y}} = X\hat{\beta}$. What are the dimensions of this quantity?

Gelman and Hill Section 3.4, pg. 38 provides a nice visual of how this representation maps onto what a typical dataset may look like, where we will try to estimate a set of coefficients that map the relationship between the columns of X and $\mathbf{y}$:

[2]This video from Ben Lambert provides additional intuition for understanding OLS in a matrix form and how it can be useful.



1.4	1	0.69	−1	−0.69	0.5	2.6	0.31
1.8	1	1.85	1	1.85	1.94	2.71	3.18
0.3	1	3.83	1	3.83	2.23	2.53	3.81
1.5	1	0.5	−1	−0.5	1.85	2.5	1.73
2.0	1	2.29	−1	−2.29	2.99	3.26	2.51
2.3	1	1.62	1	1.62	0.51	0.77	1.01
0.2	1	2.29	−1	−2.29	1.57	1.8	2.44
0.9	1	1.8	1	1.8	3.72	1.1	1.32
1.8	1	1.22	1	1.22	1.13	1.05	2.66
1.8	1	0.92	−1	−0.92	2.29	2.2	2.95
0.2	1	1.7	1	1.7	0.12	0.17	2.86
2.3	1	1.46	−1	−1.46	2.28	2.4	2.04
−0.3	1	4.3	1	4.3	2.3	1.87	0.48
0.4	1	3.64	−1	−3.64	1.9	1.13	0.51
1.5	1	2.27	1	2.27	0.47	3.04	3.12
?	1	1.63	−1	−1.63	0.84	2.35	1.25
?	1	0.65	−1	−0.65	2.08	1.26	2.3
?	1	1.83	−1	−1.83	1.84	1.58	2.99
?	1	2.58	1	2.58	2.03	1.8	1.39
?	1	0.07	−1	−0.07	2.1	2.32	1.27

Figure 3.6 *Notation for regression modeling. The model is fit to the observed outcomes y given predictors X. As described in the text, the model can then be applied to predict unobserved outcomes $\tilde{y}$ (indicated by small question marks), given predictors on new data $\tilde{X}$.*

This is a good place to review Gelman and Hill 3.4 on different notations for representing the regression model.

4.2.1 Estimating the Coefficients

Models generally start with some goal. In OLS, our goal is to minimize the sum of squared "residuals." Here is a video I created to explain why we can represent this as $\mathbf{e}'\mathbf{e}$.

https://www.youtube.com/watch?v=IFX_n16YTm0

Note: at the end of the video it should read $X\hat{\beta}$, not $\hat{X}\beta$

What is a residual? It's the difference between y and our estimate of y: $y - \hat{y}$. It represents the error in our prediction– how far off our estimate is of the outcome.

We can write this in matrix notation in the following way where $\mathbf{e}$ is an $n \times 1$ vector of residuals– a residual for each observation in the data:

$$\begin{aligned}
\mathbf{e}'\mathbf{e} &= (Y' - \hat{\beta}'X')(Y - X\hat{\beta}) \\
&= Y'Y - \hat{\beta}'X'Y - Y'X\hat{\beta} + \hat{\beta}'X'X\hat{\beta} \\
&= Y'Y - 2\hat{\beta}'X'Y + \hat{\beta}'X'X\hat{\beta}
\end{aligned}$$

Recall we want a line that minimizes this quantity. We minimize the sum of squared residuals by taking the derivative with respect to β. (We want to identify the coefficients that help us achieve the goal of minimizing the squared error.) Because we are now deriving an estimate, we will use the hat over β:

- $\frac{\delta}{\delta \hat{\beta}} = -2X'Y + 2X'X\hat{\beta}$
- So what is our estimate for $\hat{\beta}$? We take first order conditions

$$0 = -2X'Y + 2X'X\hat{\beta}$$
$$\hat{\beta} = (X'X)^{-1}X'Y$$

You may wonder how we got to these answers. Don't worry, you will get your chance to solve this! The important thing to note for now, is that we have an analytic solution to our coefficient estimates.

4.3 OLS Regression in R

To run a linear regression in R, we use the `lm()` function.

The syntax is `lm(y ~ x1, data = mydata)` for a regression with `y` as the name of your dependent variable and there is one explanatory variable `x1` where `mydata` is the name of your data frame.

`lm(y ~ x1 + x2 , data = mydata)` is the syntax for a regression with two explanatory variables `x1` and `x2`, where you would add additional variables for larger multivariate regressions. By default, R will include an intercept term in the regression.

4.3.1 Example: Predicting Current Election Votes from Past Election Votes

In the American presidential election in 2000, there was an actual controversy in how ballots were cast in the state of Florida. Social scientists used data comparing the election results from 1996 in the state with 2000 as one way to help detect irregularities in the 2000 vote count. For more information on the background of this example, you can watch this video.

We will use the data `florida.csv` available here:

```
## Load Data
florida <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/florida.
```

This data set includes several variables described below, where each row represents the voting information for a particular county in Florida.

Name	Description
`county`	county name
`Clinton96`	Clinton's votes in 1996
`Dole96`	Dole's votes in 1996
`Perot96`	Perot's votes in 1996
`Bush00`	Bush's votes in 2000
`Gore00`	Gore's votes in 2000
`Buchanan00`	Buchanan's votes in 2000

In 2000, Buchanan was a third party candidate, similar to Perot in 1996. One might think that counties where Perot received a lot of votes in 1996 should also receive a lot in 2000. That is: with a one-vote increase in Perot's vote, we might expect an average increase in Buchanan's 2000 vote.

We can translate that language into a regression equation:

- $Buchanan2000 = \alpha + Perot1996 * \beta + \epsilon$

In R, we run this regression the following way. We will save it as an object `fit.1`. You can name your regression objects anything you want.

```r
fit.1 <- lm(Buchanan00 ~ Perot96, data = florida)
```

- `summary(model)` provides the summary statistics of the model. In particular, the following statistics are important
  - `Estimate`: point estimate of each coefficient
  - `Std. Error`: standard error of each estimate
  - `t value`: indicates the t-statistic of each coefficient under the null hypothesis that it equals zero
  - `Pr(>|t|)`: indicates the two-sided p-value corresponding to this t-statistic where asterisks indicate the level of statistical significance.
  - `Multiple R-squared`: The coefficient of determination
  - `Adjusted R-squared`: The coefficient of determination adjusting for the degrees of freedom

We will say more to define these quantities in future sections.

```r
summary(fit.1)
```

```
Call:
lm(formula = Buchanan00 ~ Perot96, data = florida)

Residuals:
    Min      1Q  Median      3Q     Max 
-612.74  -65.96    1.94   32.88 2301.66 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.34575   49.75931   0.027    0.979    
Perot96      0.03592    0.00434   8.275 9.47e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 316.4 on 65 degrees of freedom
Multiple R-squared:  0.513, Adjusted R-squared:  0.5055 
F-statistic: 68.48 on 1 and 65 DF,  p-value: 9.474e-12
```

R also allows several shortcuts for accessing particular elements of your regression results. Examples:

```
## Vector of the coefficient estimates only
coef(fit.1)
```

```
(Intercept)     Perot96 
 1.34575212  0.03591504 
```

```
## Compute confidence intervals for these coefficients
confint(fit.1)
```

```
                   2.5 %       97.5 %
(Intercept) -98.03044506 100.72194929
Perot96       0.02724733   0.04458275
```

```
## Table of coefficient results only
summary(fit.1)$coefficients
```

```
              Estimate   Std. Error    t value     Pr(>|t|)
(Intercept) 1.34575212 49.759306434 0.02704523 9.785065e-01
Perot96     0.03591504  0.004340068 8.27522567 9.473505e-12
```

```
## Extract standard errors only
summary(fit.1)$coefficients[,2]
```

```
 (Intercept)      Perot96
49.759306434  0.004340068
```

```
## Variance-Covariance matrix
vcov(fit.1)
```

```
             (Intercept)       Perot96
(Intercept) 2475.9885768 -1.360074e-01
Perot96       -0.1360074  1.883619e-05
```

```
## Note that the square root of the diagonal of this matrix provides the standard errors
sqrt(diag(vcov(fit.1)))
```

```
 (Intercept)      Perot96
49.759306434  0.004340068
```

```
## Degrees of freedom
fit.1$df.residual
```

```
[1] 65
```

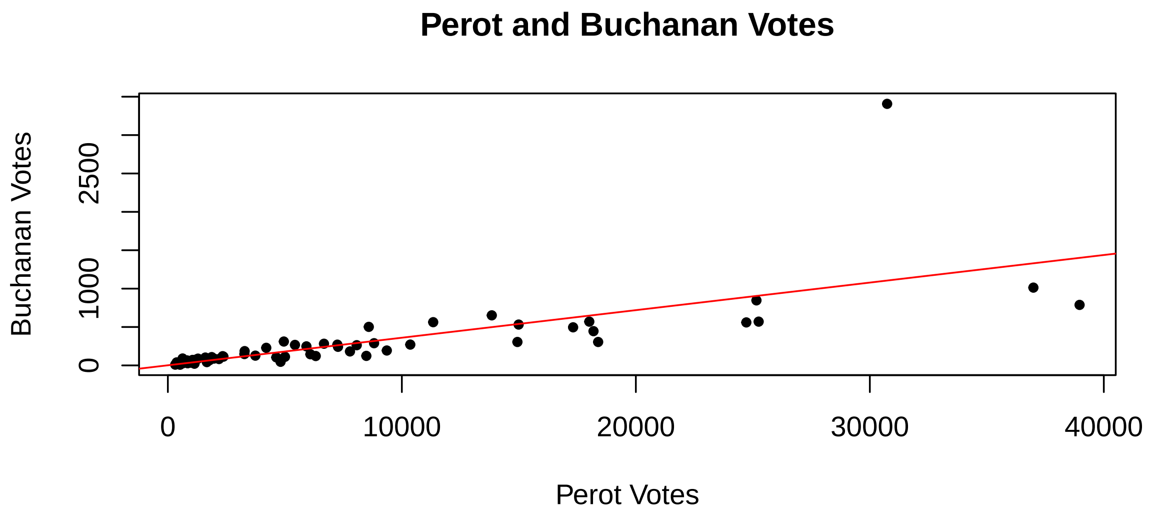
4.3.2 Plotting Regression Results

We often don't want to hide our data under a bushel basket or in complicated regression models. Instead, we might also want to visualize data in R. The function `plot()` and the function `ggplot()` from the package `ggplot2` are two terrific and flexible functions for visualizing data. We will use the `plot()` function to visualize the relationship between Perot and Buchanan votes. The example below provides a few arguments you can use within each of these functions, but they are capable of much more.

At the core, plotting functions generally work as coordinate systems. You tell R specifically at which x and y coordinates you want your points to be located (e.g., by providing R with a vector of x values and a vector of y values). Then, each function has its own way of allowing you to add bells and whistles to your figure, such as labels (e.g., `main, xlab, ylab`), point styles (`pch`), additional lines and points and text (e.g., `abline(), lines(), points(), text()`), or x and y scales for the dimensions of your axes (e.g., `xlim, ylim`). You can create a plot without these additional features, but most of the time, you will add them to make your plots look good! and be informative! We will do a lot of plotting this semester.

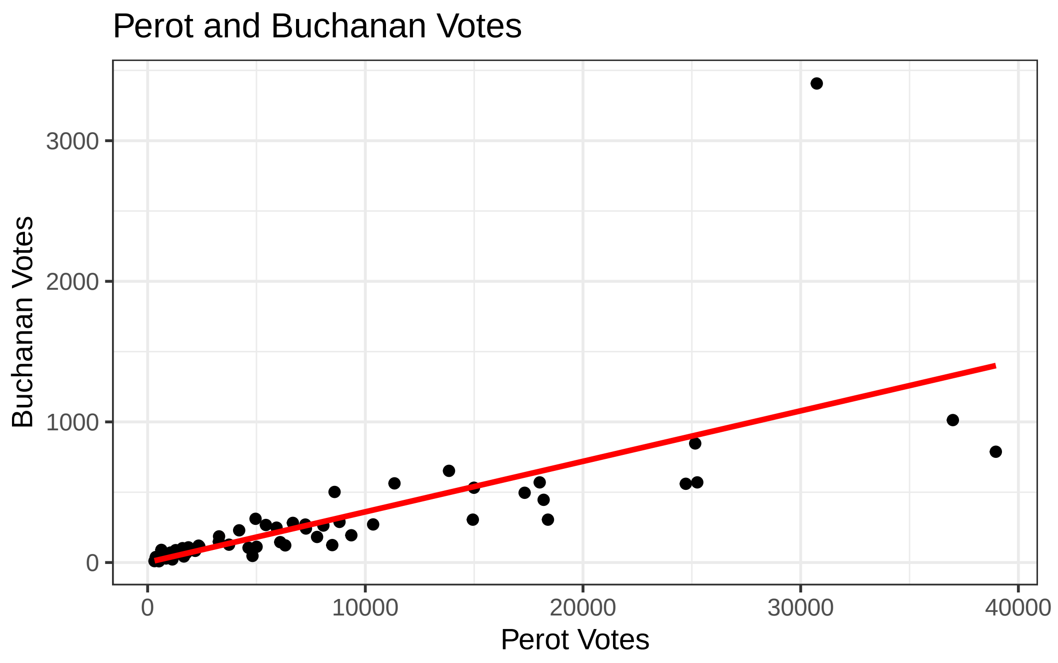
Note: feel free to use `plot()` or `ggplot()` or both. `ggplot` has similar capabilities as `plot` but relies on a different "grammar" of graphics. For example, see the subtle differences in the two plots below.

```
## Plot
plot(x = florida$Perot96, # x-values
     y = florida$Buchanan00, # y-values
     main = "Perot and Buchanan Votes", # label for main title
     ylab = "Buchanan Votes", # y-axis label
     xlab = "Perot Votes", # x-axis label
     pch = 20) # point type
abline(fit.1, col = "red") # adds a red regression line
```



```
## ggplot version
library(ggplot2)
ggplot(data = florida, # which data frame
       mapping = aes(x = Perot96, y = Buchanan00)) + # x and y coordinates
  geom_point() +  # tells R we want a scatterplot
  geom_smooth(method = "lm",
              se = FALSE, colour = "red",
              data = florida, aes(x=Perot96, y=Buchanan00)) + # adds lm regression line
  ggtitle("Perot and Buchanan Votes") + # main title
  labs(x = "Perot Votes", y = "Buchanan Votes") + # x and y labels
  theme_bw() # changes theme (e.g., color of background)
```

```
`geom_smooth()` using formula = 'y ~ x'
```



```
## Note: data = florida, aes(x=Perot96, y=Buchanan00) in the geom_smooth line is not neces
```

Tip: you might want to save your plots as .pdf or .png after you create it. You can do this straight from your R code. How you do it varies by function. The files will save to your working directory unless you specify a different file path. The code below is the same as above except it has additional lines for saving the plots:

```
## Plot
pdf(file = "myfirstmleplot.pdf", width = 7, height = 5) # play around with the dimensions
plot(x = florida$Perot96, # x-values
     y = florida$Buchanan00, # y-values
     main = "Perot and Buchanan Votes", # label for main title
     ylab = "Buchanan Votes", # y-axis label
     xlab = "Perot Votes", # x-axis label
     pch = 20) # point type
abline(fit.1, col = "red") # adds a red regression line
dev.off() # this closes your pdf file

## ggplot version
ggplot(data = florida, # which data frame
       mapping = aes(x = Perot96, y = Buchanan00)) + # x and y coordinates
  geom_point() +  # tells R we want a scatterplot
  geom_smooth(method = "lm",
              se = FALSE, colour = "red",
              data = florida, aes(x=Perot96, y=Buchanan00)) + # adds lm regression line
  ggtitle("Perot and Buchanan Votes") + # main title
  labs(x = "Perot Votes", y = "Buchanan Votes") + # x and y labels
  theme(plot.title = element_text(hjust = 0.5)) +# centers the title
  theme_bw() # changes theme (e.g., color of background)
ggsave("myfirstmleggplot.png", device="png", width = 7, height = 5) # saves the last ggplo
```

4.3.3 Finding Coefficients without `lm`

Let's put our matrix algebra and R knowledge together. In the previous section, we found that $\hat{\beta} = (X'X)^{-1}X'Y$. If we do that math directly in R, there is no need to use `lm()` to find those coefficients.

To do so, we need X and Y.

Recall Y is an $n \times 1$ vector representing the outcome of our model. In this case, Y is `Buchanan00`.

```
Y <- florida$Buchanan00
```

Recall, X is a $n \times k$ matrix representing our independent variables and a column of 1's for the intercept. Let's build this matrix using `cbind` which was introduced in section 2.

```
X <- cbind(1, florida$Perot96)
dim(X)
```

```
[1] 67  2
```

Great, now we have X and Y, so it's just about a little math. Because Y is a vector, let's make sure R knows to treat it like an $n \times 1$ matrix.

```r
Y <- cbind(Y)
dim(Y)
```

```
[1] 67  1
```

Recall the `solve()` and `t()` functions take the inverse and transpose of matrices.

```r
betahat <- solve(t(X) %*% X) %*% t(X) %*% Y
```

Finally, let's compare the results from our model using `lm()` with these results.

```r
betahat
coef(fit.1)
```

```
              Y
[1,] 1.34575212
[2,] 0.03591504
(Intercept)     Perot96 
 1.34575212  0.03591504 
```

We did it! In the problem set, you will get more experience using the analytic solutions to solve for quantities of interest instead of the built-in functions.

4.3.4 OLS Practice Problems

Here are a couple of (ungraded) problems to modify the code above and gain additional practice with data wrangling and visualization in R. As you might have noticed in the example, there is a big outlier in the data. We will see how this observation affects the results.

1. Using a linear regression examine the relationship between Perot and Buchanan votes, controlling for Bill Clinton's 1996 votes.

- Provide a one sentence summary of the relationship between Perot and Buchanan's votes.
- Is the relationship significant at the $p < 0.05$ level? What about the relationship between Clinton and Buchanan votes?
- What are the confidence intervals for the Perot coefficient results?
- What is the residual for the estimate for Palm Beach County– `PalmBeach` in the `county` variable?

2. Let's go back to the bivariate case.

- Subset the data to remove the county `PalmBeach`.
- Create a scatterplot of the relationship between Perot votes and Buchanan votes within this subset. This time make the points blue.
- Add a regression line based on this subset of data.
- Add a second regression line in a different color based on the initial bivariate regression we ran in the example, where all data were included.
- Describe the differences in the regression lines.

4.3.5 Code for solutions

```r
fit.multiple <- lm(Buchanan00 ~ Perot96 + Clinton96, data = florida)
summary(fit.multiple)
```

```

Call:
lm(formula = Buchanan00 ~ Perot96 + Clinton96, data = florida)

Residuals:
    Min      1Q  Median      3Q     Max 
-705.06  -49.17   -4.71   27.34 2254.89 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)   
(Intercept) 14.110353  51.644141   0.273  0.78556   
Perot96      0.027394   0.010095   2.714  0.00854 **
Clinton96    0.001283   0.001372   0.935  0.35325   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 316.7 on 64 degrees of freedom
Multiple R-squared:  0.5196,    Adjusted R-squared:  0.5046 
F-statistic: 34.61 on 2 and 64 DF,  p-value: 6.477e-11
```

```r
confint(fit.multiple)[2,]
```

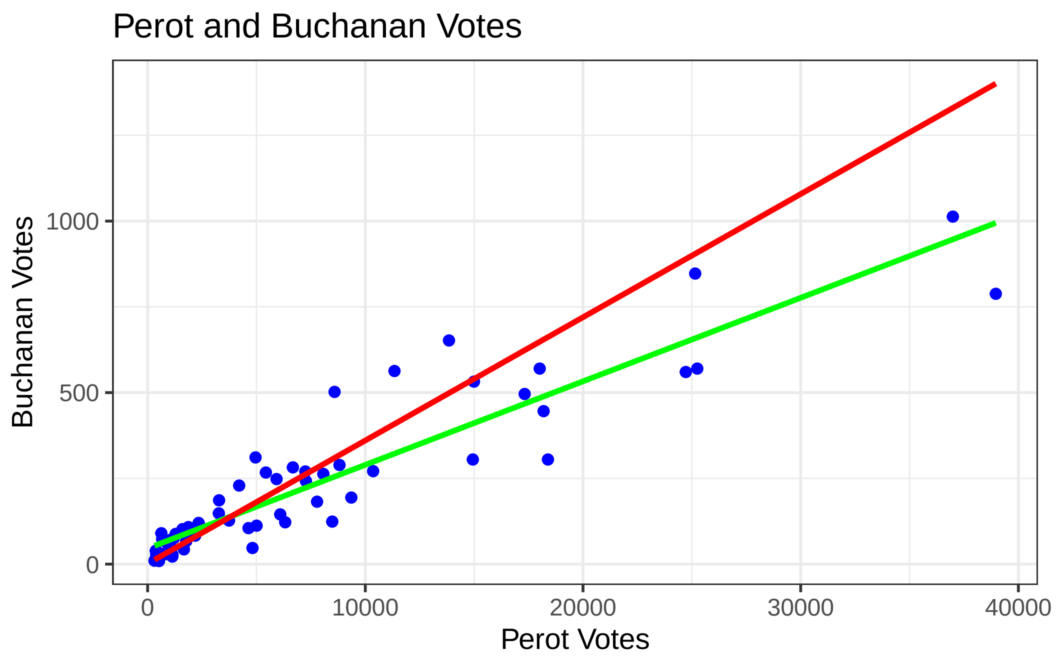
```
      2.5 %      97.5 % 
0.007228254 0.047560638 
```

```
florida$res <- residuals(fit.multiple)
florida$res[florida$county == "PalmBeach"]
```

```
[1] 2254.893
```

```
florida.pb <- subset(florida, subset = (county != "PalmBeach"))
fit2 <- lm(Buchanan00 ~ Perot96, data = florida.pb)

ggplot(data = florida.pb, # which data frame
       mapping = aes(x = Perot96, y = Buchanan00)) + # x and y coordinates
  geom_point(color="blue") +  # tells R we want a scatterplot
  geom_smooth(method = "lm",
              se = FALSE, colour = "green",
              data = florida.pb, aes(x=Perot96, y=Buchanan00)) + # adds lm regression line
    geom_smooth(method = "lm",
              se = FALSE, colour = "red",
              data = florida, aes(x=Perot96, y=Buchanan00)) + # adds lm regression line
  ggtitle("Perot and Buchanan Votes") + # main title
  labs(x = "Perot Votes", y = "Buchanan Votes") + # x and y labels
  theme(plot.title = element_text(hjust = 0.5)) +# centers the title
  theme_bw() # changes theme (e.g., color of background)
```

```
`geom_smooth()` using formula = 'y ~ x'
`geom_smooth()` using formula = 'y ~ x'
```



4.4 Uncertainty and Regression

We have now gone through the process of minimizing the sum of squared errors ($\mathbf{e}'\mathbf{e}$) and deriving estimates for the OLS coefficients $\hat{\beta} = (X'X)^{-1}X'Y$. In this section, we will discuss how to generate estimates of the uncertainty around these estimates.

Where we are going:

- In the last section, we visited an example related to the 2000 election in Florida. We regressed county returns for Buchanan in 2000 (Y) on county returns for Perot in 1996 (X).

```
## Load Data
florida <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/florida.
fit.1 <- lm(Buchanan00 ~ Perot96, data = florida)
summary(fit.1)
```

```
Call:
lm(formula = Buchanan00 ~ Perot96, data = florida)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-612.74  -65.96    1.94   32.88 2301.66

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.34575   49.75931   0.027    0.979
Perot96      0.03592    0.00434   8.275 9.47e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 316.4 on 65 degrees of freedom
Multiple R-squared:  0.513, Adjusted R-squared:  0.5055
F-statistic: 68.48 on 1 and 65 DF,  p-value: 9.474e-12
```

The summary output of the model shows many different quantities in addition to the coefficient estimates. In particular, in the second column of the summary, we see the standard errors of the coefficients. Like many statistical software programs, the `lm()` function neatly places these right next to the coefficients. We will now discuss how we get to these values.

4.4.1 Variance of the Coefficients

The standard error is the square root of the variance, representing the typical deviation we would expect to see between our estimates $\hat{\beta}$ of the parameter β across repeated samples. So to get to the standard error, we just need to get to an estimate of the variance.

Let's take the journey. First the math. As should start becoming familiar, we have our initial regression equation, which describes the relationship between the independent variables and dependent variables.

- Start with the model: $Y = X\beta + \epsilon$
  - We want to generate uncertainty for our estimate of $\hat{\beta} = (X'X)^{-1}X'Y$
- Note: Conditional on fixed values of X (I say fixed values because this is our data. We know X from our dataset.), the only random component is ϵ.
  - What does that mean? Essentially, the random error term in our regression equation is what is giving us the uncertainty. If Y was a deterministic result of X, we would have no need for it, but it's not. The relationship is not exact, varies sample to sample, subject to random perturbations, represented by ϵ.

Below we go through how to arrive at the mathematical quantity representing the variance of $\hat{\beta}$ which we will notate as $\mathbf{V}(\hat{\beta})$. The first part of the math below is just substituting terms:

$$\begin{aligned}
\mathbf{V}(\hat{\beta}) &= \mathbf{V}((X^TX)^{-1}X^TY)) \\
&= \underbrace{\mathbf{V}((X^TX)^{-1}X^T(X\beta+\epsilon))}_{\text{Sub in the expression for Y from above}} \\
&= \underbrace{\mathbf{V}((X^TX)^{-1}X^TX\beta+(X^TX)^{-1}X^T\epsilon)}_{\text{Distribute the term to the items in the parentheses}} \\
&= \underbrace{\mathbf{V}(\beta+(X^TX)^{-1}X^T\epsilon)}_{\text{Using the rules of inverses, the two terms next to } \beta \text{ canceled each other out}}
\end{aligned}$$

The next part of the math requires us to use knowledge of the definition of variance and the rules associated. We draw on two in particular:

- The variance of a constant is zero.
- When you have a constant multipled by a random variable, e.g., $\mathbf{V}(4d)$, it can come out of the variance operator, but must be squared: $16\mathbf{V}(d)$
- Putting these together: $\mathbf{V}(2+4d) = 16\mathbf{V}(d)$

Knowing these rules, we can proceed:

$$\begin{aligned}
\mathbf{V}(\hat{\beta}) &= \mathbf{V}(\beta+(X^TX)^{-1}X^T\epsilon) \\
&= \underbrace{\mathbf{V}((X^TX)^{-1}X^T\epsilon)}_{\beta \text{ drops out because in a regression it is an unkown "parameter"– it's constant, which means its variance is zero.}} \\
&= \underbrace{(X^TX)^{-1}X^T\mathbf{V}(\epsilon)((X^TX)^{-1}X^T)^T}_{\text{We can move } (X^TX)^{-1}X^T \text{ out front because our data are fixed quantities, but in doing so, we have to "square" the matrix.}} \\
&= (X^TX)^{-1}X^T\mathbf{V}(\epsilon)X(X^TX)^{-1}
\end{aligned}$$

The resulting quantity is our expression for the $\mathbf{V}(\hat{\beta})$. However, in OLS, we make an additional assumption that allows us to further simplify the expression. We assume homoscedasticity aka "constant" or "equal error variance" which says that the variance of the errors are the same across observations: $\mathbf{V}(\epsilon) = \sigma^2 I_n$.

- If we assume homoscedastic errors, then $\text{Var}(\epsilon) = \sigma^2 I_n$

$$\begin{aligned}
\mathbf{V}(\hat{\beta}) &= (X^TX)^{-1}X^T\mathbf{V}(\epsilon)X(X^TX)^{-1} \\
&= \underbrace{(X^TX)^{-1}X^T\sigma^2 I_n X(X^TX)^{-1}}_{\text{Assume homoskedasticity}} \\
&= \underbrace{\sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1}}_{\text{Because it is a constant, we can move it out in front of the matrix multiplication, and then simplify the terms.}} \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}$$

All done! This expression: $\sigma^2(X^TX)^{-1}$ represents the variance of our coefficient estimates. Note its dimensions: $k \times k$. It has the same number of rows and columns as the number of our independent variables (plus the intercept).

There is one catch, though. How do we know what σ^2 is? Well, we don't. Just like the unknown parameter β, we have to estimate it in our regression model.

Just like with the coefficients, we notate our estimate as $\hat{\sigma}^2$. Our estimate is based on the observed residual errors in the model and is as follows:

- $\hat{\sigma}^2 = \frac{1}{N-K}\sum_{i=1}^N \widehat{\epsilon_i^2} = \frac{1}{N-K}\mathbf{e}'\mathbf{e}$

That means our **estimate** of the variance of the coefficients is found within: $\hat{\sigma}^2(X^TX)^{-1}$

Again, this is a $k \times k$ matrix and is often called the variance covariance matrix. We can extract this quantity from our linear models in R using `vcov()`.

```
vcov(fit.1)
```

```
              (Intercept)       Perot96
(Intercept) 2475.9885768 -1.360074e-01
Perot96       -0.1360074  1.883619e-05
```

This is the same that we would get if manually we took the residuals and multiplied it by our X matrix according to the formula above:

```
X <- cbind(1, florida$Perot96)
e <- cbind(residuals(fit.1))
sigmahat <- ((t(e) %*% e) / (nrow(florida) -2))
## tell r to stop treating sigmahat as a matrix
sigmahat <-as.numeric(sigmahat)
XtX <- solve(t(X) %*%X)
sigmahat * XtX
```

```
             [,1]          [,2]
[1,] 2475.9885768 -1.360074e-01
[2,]   -0.1360074  1.883619e-05
```

The terms on the diagonal represent the variance of a particular coefficient in the model.The standard error of a particular coefficient k is: s.e.$(\widehat{\beta}_k) = \sqrt{\widehat{\sigma}^2(X'X)^{-1}}_{kk}$. The off-diagonal components represent the covariances between the coefficients.

Recall that the standard error is just the square root of the variance. So, to get the nice standard errors we saw in the summary output, we can take the square root of the quantities on the diagonal of this matrix.

```
sqrt(diag(vcov(fit.1)))
```

```
 (Intercept)      Perot96 
49.759306434  0.004340068 
```

```
summary(fit.1)$coefficients[,2]
```

```
 (Intercept)      Perot96 
49.759306434  0.004340068 
```

Why should I care?

1. Well R actually doesn't make it that easy to extract standard errors from the summary output. You can see above that the code for extracting the standard errors using what we know about them being the square root of the variance is about as efficient as extracting the second column of the coefficient component of the summary of the model.
2. Sometimes, we may think that the assumption of equal error variance is not feasible and that we have unequal error variance or "heteroscedasticity." Researchers have developed alternative expressions to model unequal error variance. Generally, what this means is they can no longer make that simplifying assumption, have to stop at the step with the uglier expression $(X^TX)^{-1}X^T\mathbf{V}(\epsilon)X(X^TX)^{-1}$ and then assume something different about the structure of the errors in order to estimate the coefficients. These alternative variance estimators are generally what are referred to as "robust standard errors." There are many different robust estimators, and you will likely come across them in your research.

Some of you may have learned the formal, general definition for variance as defined in terms of expected value: $\mathbb{E}[(\widehat{m} - \mathbb{E}(\widehat{m}))^2]$. We could also start the derivation there. This is not required for the course, but it is below if you find it useful. In particular, it can help show why we wend up needing to square a term when we move it outside the variance operator:

$$
\begin{aligned}
\mathbf{V}(\hat{\beta}) &= \mathbb{E}[(\hat{\beta} - \mathbb{E}(\hat{\beta}))^2)] \\
&= \mathbb{E}[(\hat{\beta} - \beta)^2] \\
&= \mathbb{E}[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T] \\
&= \mathbb{E}[(X^TX)^{-1}X^TY - \beta)(X^TX)^{-1}X^TY - \beta)^T] \\
&= \mathbb{E}[(X^TX)^{-1}X^T(X\beta + \epsilon) - \beta)(X^TX)^{-1}X^T(X\beta + \epsilon) - \beta)^T] \\
&= \mathbb{E}[(X^TX)^{-1}X^TX\beta + (X^TX)^{-1}X^T\epsilon - \beta)(X^TX)^{-1}X^TX\beta + (X^TX)^{-1}X^T\epsilon - \beta)^T] \\
&= \mathbb{E}[(\beta + (X^TX)^{-1}X^T\epsilon - \beta)(\beta + (X^TX)^{-1}X^T\epsilon - \beta)^T] \\
&= \mathbb{E}[(X^TX)^{-1}X^T\epsilon)(X^TX)^{-1}X^T\epsilon)^T] \\
&= (X^TX)^{-1}X^T\mathbb{E}(\epsilon\epsilon^T)X(X^TX)^{-1} \\
&= \underbrace{(X^TX)^{-1}X^T\sigma^2 I_n X(X^TX)^{-1}}_{\text{Assume homoskedasticity}} \\
&= \sigma^2(X^TX)^{-1}X^TX(X^TX)^{-1} \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}
$$

Note: Along the way, in writing $\mathbb{E}(\hat{\beta}) = \beta$, we have implicitly assumed that $\hat{\beta}$ is an "unbiased" estimator of β. This is not free. It depends on an assumption that the error term in the regression ϵ is independent of our independent variables. This can be violated in some situations, such as when we have omitted variable bias, which is discussed at the end of our OLS section.

4.4.2 Hypothesis Testing

Most of the time in social science, we run a regression because we have some hypothesis about how a change in our independent variable affects the change in our outcome variable.

In OLS, we can perform a hypothesis test for each independent variable in our data. The structure of the hypothesis test is:

- Null hypothesis: $\beta_k = 0$
  - This essentially means that we don't expect a particular x_k independent variable to have a relationship with our outcome variable.
- Alternative hypothesis: $\beta_k \neq 0$
  - We do expect a positive or negative relationship between a particular x_k and the dependent variable.

We can use our estimates for $\hat{\beta}$ coefficients and their standard errors to come to a conclusion about rejecting or failing to reject the null hypothesis of no relationship by using a t-test.

In a t-test, we take our coefficient estimates and divide them by the standard error in order to "standardize" them on a scale that we can use to determine how likely it is we would have observed a value for $\hat{\beta}$ as extreme or more extreme as the one we observed in a world where the true $\beta = 0$. This is just like a t-test you might have encountered before for a difference in means between groups, except this time our estimate is $\hat{\beta}$.

$$t_{\hat{\beta}_k} = \frac{\hat{\beta}_k}{s.e.(\hat{\beta}_k)}$$

Generally speaking, when t is about +/-2 or greater in magnitude, the coefficient will be "significant" at conventional levels (i.e., $p < 0.05$), meaning that we are saying that it is really unlikely we would have observed a value as big as $\hat{\beta}_k$ if the null hypothesis were true. Therefore, we can reject the null hypothesis.

However, to get a specific quantity, we need to calculate the p-value, which depends on the t-statistic and the degrees of freedom in the model. The degrees of freedom in a regression model are $N - k$, the number of observations in the model minus the number of independent variables plus the intercept.

In R, we can calculate p-values using the `pt()` function. By default, most people use two-sided hypothesis tests for regression. So to do that, we are going to find the area on each side of the t values, or alternatively, multiply the area to the right of our positive t-value by 2.

```r
## Let's say t was 2.05 and
## And there were 32 observations and 3 variables in the regression plus an intercept
t <- 2.05
df.t <- 32 -4
p.value <- 2 * (pt(abs(t), df=df.t, lower.tail = FALSE))
p.value
```

```
[1] 0.04983394
```

Let's do this for the florida example. First, we can find t by dividing our coefficients by the standard errors.

```r
t <- coef(fit.1) / (sqrt(diag(vcov(fit.1))))
t
```

```
(Intercept)     Perot96 
 0.02704523  8.27522567 
```

```
## Compare with output
summary(fit.1)$coefficients[, 3]
```

```
(Intercept)     Perot96 
 0.02704523  8.27522567 
```

We can then find the p-values.

```
t <- coef(fit.1) / (sqrt(diag(vcov(fit.1))))
df.t <- fit.1$df.residual
p.value <- 2 * (pt(abs(t), df=df.t, lower.tail = FALSE))
p.value
```

```
 (Intercept)      Perot96 
9.785065e-01 9.473505e-12 
```

```
summary(fit.1)$coefficients[, 4]
```

```
 (Intercept)      Perot96 
9.785065e-01 9.473505e-12 
```

We see that the coefficient for `Perot96` is significant. The p-value is tiny. In R, for small numbers, R automatically shifts to scientific notation. The 9.47e-12 means the p-value is essentially zero, with the stars in the summary output indicating the p-value is $p < 0.001$. R will also output a test of the significance of the intercept using the same formula as all other coefficients. This generally does not have much interpretive value, so you are usually safe to ignore it.

Confidence Intervals

Instead of representing the significance using p-values, sometimes it is helpful to report confidence intervals around the coefficients. This can be particularly useful when visualizing the coefficients. The 95% confidence interval represents roughly 2 standard errors above and below the coefficient. The key thing to look for is whether it overlaps with zero (not significant) or does not (in which case the coefficient is significant).

The precise formula is

$\hat{\beta}_k$ Confidence intervals: $\hat{\beta}_k - t_{crit.value} \times s.e._{\hat{\beta}_k}, \hat{\beta}_k + t_{crit.value} \times s.e_{\hat{\beta}_k}$

In R, we can use `qt()` to get the specific critical value associated with a 95% confidence interval. This will be around 2, but fluctuates depending on the degrees of freedom in your model (which are function of your sample size and how many variables you have in the model.) R also has a shortcut `confint()` function to extract the coefficients from the model. Below we do this for the `Perot96` coefficient.

```
## Critical values from t distribution at .95 level
qt(.975, df = fit.1$df.residual) # n- k degrees of freedom
```

```
[1] 1.997138
```

```
## Shortcut
confint(fit.1)[2,]
```

```
     2.5 %     97.5 %
0.02724733 0.04458275
```

```
## By hand
coef(fit.1)[2] - qt(.975, df = fit.1$df.residual)*sqrt(diag(vcov(fit.1)))[2]
```

```
   Perot96
0.02724733
```

```
coef(fit.1)[2] + qt(.975, df = fit.1$df.residual)*sqrt(diag(vcov(fit.1)))[2]
```

```
   Perot96
0.04458275
```

4.4.3 Goodness of Fit

A last noteworthy component to the standard regression output is the goodness of fit statistics. For this class, we can put less attention on these, though there will be some analogues when we get into likelihood.

These are measures of how much of the total variation in our outcome measure can be explained by our model, as well as how far off are our estimates from the truth.

For the first two measures R-squared and Adjusted R-squared, we draw on three quantities:

- Total Sum of Squares–how much variance in Y_i is there to explain?
  - $TSS : \sum_{i=1}^{N}(Y_i - \overline{Y}_i)^2$
- Estimated Sum of Squares–how much of this variance do we explain?
  - $ESS : \sum_{i=1}^{N}(\widehat{Y}_i - \overline{Y}_i)^2$
- Residual Sum of Squares–how much variance is unexplained?
  - $RSS : \sum_{i=1}^{N}(Y_i - \widehat{Y}_i)^2$
- $TSS = ESS + RSS$
- Multiple R-squared: $\frac{ESS}{TSS}$
  - This is a value from 0 to 1, representing the proportion of the variance in the outcome that can be explained by the model. Higher values are generally considered better, but there are many factors that can affect R-squared values. In most social science tasks where the goal is to engage in hypothesis testing of coefficients, this measure is of less value.
- Adjusted R-squared: $1 - \frac{\frac{RSS}{n-k}}{\frac{TSS}{n-1}}$
  - This is essentially a penalized version of R-squared. When you add additional predictors to a model, the R-squared value can never decrease, even if the predictors are useless. The Adjusted R-squared adds a consideration for the degrees of freedom into the equation, creating a penalty for adding more and more predictors.
- Residual standard error aka root mean squared error aka square root of the mean squared residual: $r.s.e = \sqrt{\frac{RSS}{n-k}}$
  - This represents the typical deviation of an estimate of the outcome from the actual outcome. This quantity is often used to assess the quality of prediction exercises. It is used less often in social science tasks where the goal is hypothesis testing of the relationship between one or more independent variables and the outcome.

F-Statistic

So far we have conducted hypothesis tests for each individual coefficient. We can also conduct a global hypothesis test, where the null hypothesis is that all coefficients are zero, with the alternative being that at least one coefficient is nonzero. This is the test represented by the F-statistic in the regression output.

The F-statistic helps us test the null hypothesis that **all** of the regression slopes are 0: $H_0 = \beta_1 = \beta_2 = \cdots = \beta_k = 0$

- $F_0 = \frac{ESS/(k-1)}{RSS/(n-k)}$
- The F-Statistic has two separate degrees of freedom.
  - The model sum of squares degrees of freedom (ESS) are $k-1$.
  - The residual error degrees of freedom (RSS) are $n-k$.
  - In a regression output, the model degrees of freedom are generally the first presented: "F-statistic: 3.595 on $(k-1)=1$ and $(n-k)=48$ DF."

Note: This test is different from our separate hypothesis tests that a k regression slope is 0. For that, we use the t-tests discussed above.

4.5 Generating predictions from regression models

The regression coefficients tell us how much Y is expected to change for a one-unit change in x_k. It does not immediately tell us the values we estimate our outcome ($\hat{Y}$) to take conditional on particular values of x_k. While often knowing our independent variables have a significant effect on the outcome and the size of the coefficient is sufficient for testing our hypotheses, it can be helpful for interpretation's sake, to see the estimated values for the outcome. This is going to be particularly important once we get into models like logistic regression, where the coefficients won't be immediately interpretable.

Recall that our equation for estimating values of our outcomes is:

- $\hat{Y} = X\hat{\beta}$ This can also be written out in long form for any particular observation i:
- $\hat{y}_i = \hat{\alpha} + \hat{\beta}_1 * x_1 i + \hat{\beta}_2 * x_2 i + ...\hat{\beta}_k * x_k i$

The estimated values of our regression $\hat{Y}$ are often called the "fitted values." In R, you can identify the estimated values for each observation using the `fitted()` command.

```r
## Y hat for the first observation in the data
fitted(fit.1)[1]
```

```
      1 
291.252 
```

Again, this is just the multiplication of the matrix X and $\hat{\beta}$. If we have already run a regression model in R, one shortcut for getting the X matrix, is to use the `model.matrix` command. We can get $\hat{\beta}$ using the `coef()` command.

```r
X <- model.matrix(fit.1)
head(X) # head() shows about the first six values of an object
```

```
  (Intercept) Perot96
1           1    8072
2           1     667
3           1    5922
4           1     819
5           1   25249
6           1   38964
```

```r
betahat <- coef(fit.1)
```

Our fitted values are then just

```r
yhat <- X %*% betahat
head(yhat)
```

```
        [,1]
1  291.25196
2   25.30108
3  214.03462
4   30.76017
5  908.16461
6 1400.73939
```

If I want to generate an estimate for any particular observation, I could also just extract its specific value for `Perot96`.

```r
florida$Perot96[1]
```

```
[1] 8072
```

Let's estimate the Buchanan 2000 votes for the first county in the data with Perot 96 votes of 8072. We can write it out as $\widehat{Buchanan00}_1 = \hat{\alpha} + \hat{\beta} * Perot96_1$

```r
buch00hat <- coef(fit.1)[1] + coef(fit.1)[2]*florida$Perot96[1]
buch00hat
```

```
(Intercept)
    291.252
```

What is useful about this is that now we have the coefficient estimates, we can apply them to any values of X we wish in order to generate estimates/predictions of the values Y will take given particular values of our independent variables.

One function that is useful for this (as a shortcut) is the `predict(fit, newdata=newdataframe)` function in R. It allows you to enter in "newdata"– meaning values of the X variables for which you want to generate estimates of Y based on the coefficient estimates of your regression model.

For example, let's repeat the calculation from above for `Perot96 = 8072`.

```
predict(fit.1, newdata = data.frame(Perot96 = 8072))
```

```
      1 
291.252 
```

We can also generate confidence intervals around these estimates by adding `interval = "confidence"` in the command.

```
predict(fit.1, newdata = data.frame(Perot96 = 8072), interval="confidence")
```

```
      fit      lwr      upr
1 291.252 213.7075 368.7964
```

We can also simultaneously generate multiple predictions by supplying a vector of values in the `predict()` command. For example, let's see the estimated Buchanan votes for when the Perot 1996 votes took values of 1000 to 10,000 by intervals of 1,000.

```
predict(fit.1, newdata = data.frame(Perot96 = c(1000, 2000, 3000, 4000, 5000, 6000, 7000,
```

```
        1         2         3         4         5         6         7         8 
 37.26079  73.17583 109.09087 145.00591 180.92095 216.83599 252.75104 288.66608 
        9        10 
324.58112 360.49616 
```

The important thing to note about the `predict()` command is that if you have multiple independent variables, you have to specify the values you want each of them to take when generating the estimated values of y.

```
fit.2 <- lm(Buchanan00 ~ Perot96 + Clinton96, data = florida)
```

For example, let's build a second model with `Clinton96` as an additional predictor. In order to generate the same prediction for different values of Perot 1996 votes, we need to tell R at what values we should "hold constant" `Clinton96`. I.e., we want to see how hypothetical changes in `Perot96` votes influence changes in Buchanan 2000 votes while also leaving the Clinton votes identical. This is that lightswitch metaphor– flipping one switch, while keeping the rest untouched.

There are two common approaches to doing this. 1) We can hold constant `Clinton96` votes at its mean value in the data 2) We can keep `Clinton96` at its observed values in the data. In linear regression, it's not going to matter which approach you take. In other models we will talk about later, this distinction may matter more substantially because of how our quantities of interest change across different values of $X\hat{\beta}$.

The first approach is easily implemented in predict.

```
predict(fit.2, newdata = data.frame(Perot96 = 8072, Clinton96 = mean(florida$Clinton96)))
```

```
      1
283.997
```

For the second approach, what we will do is generate an estimate for Buchanan's votes in 2000 when Perot96 takes 8072 votes, and we keep Clinton96's votes at whatever value it currently is in the data. That is, we will generate n estimates for Buchanan's votes when Perot takes 8072. Then, we will take the mean of this as our "average estimate" of Buchanan's votes in 2000 based on Perot's votes at a level of 8072. We can do this in one of two ways:

```
## Manipulating the X matrix
X <- model.matrix(fit.2)
## Replace Perot96 column with all 8072 values
X[, "Perot96"] <- 8072
head(X) #take a peek
```

```
  (Intercept) Perot96 Clinton96
1           1    8072     40144
2           1    8072      2273
3           1    8072     17020
4           1    8072      3356
5           1    8072     80416
6           1    8072    320736
```

```r
## Generate yhat
yhats <-X %*% coef(fit.2)

## take the mean
mean(yhats)
```

```
[1] 283.997
```

```r
## Use predict
yhats <- predict(fit.2, newdata = data.frame(Perot96=8072, Clinton96=florida$Clinton96))
mean(yhats)
```

```
[1] 283.997
```

Now often, after we generate these predicted values, we want to display them for the whole world to see. You will get a chance to visualize values like this using the plotting functions in the problem sets. We have already seen one example of this in the simple bivariate case, when R plotted the bivariate regression line in section 4.3.2. However, the predict function extends are capabilities to plot very specific values of X and $\hat{Y}$ for bivariate or multiple regressions.

- The `predict()` function is also very relevant when we move to logistic, probit, etc. regressions. This is just the start of a beautiful friendship between you and `predict()` and associated functions.

4.6 Wrapping up OLS

Linear regression is a great way to explain the relationship between one or more independent variables and an outcome variables. However, there is no free lunch. We have already mentioned a couple of assumptions along the way. Below we will summarize these and other assumptions. These are things you should be mindful of when you use linear regression in your own work. Some conditions that generate violations of these assumptions can also motivate why we will seek out alternative methods, such as those that rely on maximum likelihood estimation.

This is a good place to review Gelman section 3.6.

- Exogeneity. This one we haven't discussed yet, but is an important assumption for letting us interpret our coefficients $\hat{\beta}$ as "unbiased" estimates of the true parameters β. We assume that the random error in the regression model ϵ is indeed random, and uncorrelated with and independent of our independent variables X. Formally:

- $\mathbb{E}(\epsilon|X) = \mathbb{E}(\epsilon) = 0$.
- This can be violated, for example, when we suffer from Omitted Variable Bias due to having an "endogenous explanatory variable" that is correlated with some unobserved or unaccounted for factor. This bias comes from a situation where there is some variable that we have left out of the model (Z), and is therefore a part of the unobserved error term. Moreover this variable which is correlated with–and a pre-cursor of– our independent variables and is a cause of our dependent variable. A failure to account for omitted variables can create bias in our coefficient estimates. Concerns about omitted variable bias often prompt people to raise their hands in seminars and ask questions like, "Well have you accounted for this? Have you accounted for that? How do you know it is X driving your results and not Z?" If we omit important covariates, we may wrongly attribute an effect to X when it was really the result of our omitted factor Z. Messing discusses this here.
- This is a really tough assumption. The only real way to guarantee the independence of your error term and the independent variables is if you have randomly assigned values to the independent variables (such as what you do when you randomly assign people to different treatment conditions in an experiment). Beyond random assignment, you have to rely on theory to understand what variables you need to account for in the regression model to be able to plausibly claim your estimate of the relationship between a given independent variable and the dependent variable is unbiased. Failing to control for important factors can lead to misleading results, such as what happens in Simpson's paradox, referenced in the Messing piece.
- Danger Note 1: The danger here, though, is that the motivation for avoiding omitted variable bias might be to keep adding control after control after control into the regression model. However, model building in this way can sometimes be atheoretical and result in arbitrary fluctuations in the size of your coefficients and their significance. At its worse, it can lead to "p-hacking" where researchers keep changing their models until they find the results they like. The Lenz and Sahn article on Canvas talks more about the dangers of arbitrarily adding controls to the model.
- Danger Note 2: We also want to avoid adding "bad controls" to the model. Messing talks about this in the medium article as it relates to collider bias. We want to avoid adding controls to our model, say W that are actually causes of Y and causes of X instead of the other way around.
- Model building is a delicate enterprise that depends a lot on having a solid theory that guides the choice of variables.

- Homoscedasticity. We saw this when defining the variance estimator for the OLS coefficients. We assume constant error variance. This can be violated when we think observations at certain values of our independent variables may have different magnitudes of error than observations at other values of our independent variables.

  - No correlation in the errors. The error terms are not correlated with each other. This can be violated in time series models (where we might think past, present, and future errors are correlated) or in cases where our observations are nested in some hierarchical structures (e.g., perhaps students in a school) and the errors are correlated.

- No perfect collinearity. The X matrix must be full rank: We cannot have linear dependence between columns in our X matrix. We saw this in the tutorial when we tried to add the dummy variables for all of our racial groups into a regression at once. When there is perfect collinearity between variables, our regression will fail.
  - We should also avoid situations where we have severe multicollinearity. This can happen when we include two or more variables in a regression model that are highly correlated (just not perfectly correlated). While the regression will still run in this case, it can inflate the standard errors of the coefficients, making it harder to detect significant effects. This is particularly problematic in smaller samples.
- Linearity. The relationship between the independent and dependent variables needs to be linear in the parameters. It should be modeled as the addition of constants or parameters multiplied by the independent variables. If instead the model requires the multiplication of parameters, this is no longer linear (e.g., β^2). Linearity also often refers to the shape of the model. Our coefficients tell us how much change we expect in the outcome for each one-unit change in an independent variable. We might think some relationships are nonlinear– meaning this rate of change varies across values of the independent variables. If that is the case, we need to shift the way our model is specified to account for this or change modeling approaches.
  - For example, perhaps as people get older (one-unit changes in age), they become more politically engaged, but at some age level, their political engagement starts to decline. This would mean the slope (that expected change in political engagement for each one-unit change in age) is not constant across all levels of age. There, we might be violating linearity in the curvature of the relationship between the independent and dependent variables. This is sometimes why you might see age^2 or other nonlinear terms in regression equations to better model this curvature.
  - Likewise, perhaps each additional level of education doesn't result in the same average increase in y. If not, you could consider including categorical dummy variables for different levels of education instead of treating education as a numeric variable.
- Normality. We assume that the errors are normally distributed. As Gelman 3.6 notes, this is a less important assumption and is generally not required.

OLS Properties

Why we like OLS. When we meet our assumptions, OLS produces the best linear unbiased estimates (BLUE). A discussion of this here. We have linearity in our parameters (e.g., β and not β^2 for example). The unbiasedness means that the expected value (aka the average over repeated samples) of our estimates $\mathbb{E}(\hat{\beta}) = \beta$ is the true value. Our estimates are also efficient, which has to do with the variance, not only are our estimates true in expectation, but we also have lower variance than an alternative linear unbiased estimator could get us. If our assumptions fail, then we might no longer have BLUE. OLS estimates are also consistent, meaning that as the sample gets larger and larger, the estimates start converging to the truth.

Now, a final hidden assumption in all of this is that the sample of our data is representative of the population we are trying to make inferences about. If that is not the case, then we may no longer be making unbiased observations to that population level. Further adjustments may be required (e.g., analyses of survey data sometimes use weights to adjust estimates to be more representative).

When we violate these assumptions, OLS may no longer be best, and we may opt for other approaches. More soon!

4.6.1 Practice Problems

1. Let's use the `florida` data. Run a regression according to the following formula:
    - $Buchanan00_i = \alpha + \beta_1 * Perot96_i + \beta_2 * Dole96 + \beta_3 * Gore00 + \epsilon$
2. Report the coefficient for `Perot96`. What do you conclude about the null hypothesis that there is no relationship between 1996 Perot votes and 2000 Buchanan votes?
3. What is the confidence interval for the `Perot96` coefficient estimate?
4. When Perot 1996 vote is 5500, what is the expected 2000 Buchanan vote?

4.6.2 Practice Problem Code for Solutions

```
fit.practice <- lm(Buchanan00 ~ Perot96 + Dole96 + Gore00, data = florida)

coef(fit.practice)["Perot96"]
```

```
   Perot96
0.02878927
```

```
confint(fit.practice)["Perot96", ]
```

```
      2.5 %      97.5 %
0.004316382 0.053262150
```

```r
expbuch <- model.matrix(fit.practice)
expbuch[,"Perot96"] <- 5500
mean(expbuch %*% as.matrix(coef(fit.practice)))
```

```
[1] 211.1386
```

4.7 Week 2 Example

This example is based on Dancygier, Rafaela; Egami, Naoki; Jamal, Amaney; Rischke, Ramona, 2020, "Hate Crimes and Gender Imbalances: Fears over Mate Competition and Violence against Refugees", published in the *American Journal of Political Science*. Replication data is available here. We will draw on the survey portion of the article and replicate Table 1 in the paper. The pre-print is available here.

The abstract is: *As the number of refugees rises across the world, anti-refugee violence has become a pressing concern. What explains the incidence and support of such hate crime? We argue that fears among native men that refugees pose a threat in the competition for female partners is a critical but understudied factor driving hate crime. Employing a comprehensive dataset on the incidence of hate crime across Germany, we first demonstrate that hate crime rises where men face disadvantages in local mating markets. Next, we complement this ecological evidence with original survey measures and confirm that individual-level support for hate crime increases when men fear that the inflow of refugees makes it more difficult to find female partners. Mate competition concerns remain a robust predictor even when controlling for antirefugee views, perceived job competition, general frustration, and aggressiveness. We conclude that a more complete understanding of hate crime and immigrant conflict must incorporate marriage markets and mate competition.*

The authors summarize their hypotheses as, "the notion that male refugees are engaged in romantic relationships with German women has received considerable media attention from a variety of sources, with coverage ranging from the curious to the outright hostile. We argue that the prospect of refugee-native mate competition can trigger or compound resentment against refugees, including support for hate crime" pg. 14

```r
library(foreign)
dat_use <- read.dta("https://github.com/ktmccabe/teachingdata/blob/main/dat_use.dta?raw=tr
```

The data include wave 4 of an online survey fielded in Germany through Respondi from September 2016 to December 2017). Each wave was designed to be nationally representative on age (starting at 18), gender, and state (Bundesland) with a sample of about 3,000 respondents in each wave.

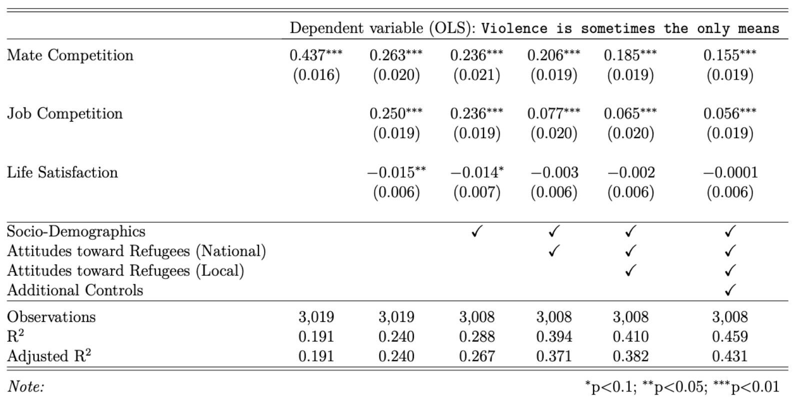
Key variables include

- `hate_violence_means` representing respondents' agreement or disagreement to the Only Means question: "When it comes to the refugee problem, violence is sometimes the only means that citizens have to get the attention of German politicians." from (1) disagree strongly to (4) agree strongly.
- `MateComp_cont`, Mate Competition. "The inflow of refugees makes it more difficult for native men to find female partners." from (1) disagree strongly to (4) agree strongly.
- The data include several other variables related to the demographics of the respondents and measures representing potential alternative explanations, such as `JobComp_cont` (agreement with "the inflow of young male refugees makes it more difficult for young native men to find apprenticeships and jobs") and `LifeSatis_cont` (0-10 scale, ranging from extremely dissatisfied to extremely satisfied).

Let's pause here to ask a few questions about research design.

- What is the outcome? What is the independent variable of interest?
  - How would we write out the bivariate regression model?
- Why OLS? (e.g., why not experiment?)
- What types of alternative explanations might exist?

Ok let's move to replication of the first two regression models in the table:



	Dependent variable (OLS): `Violence is sometimes the only means`					
Mate Competition	0.437***	0.263***	0.236***	0.206***	0.185***	0.155***
	(0.016)	(0.020)	(0.021)	(0.019)	(0.019)	(0.019)
Job Competition		0.250***	0.236***	0.077***	0.065***	0.056***
		(0.019)	(0.019)	(0.020)	(0.020)	(0.019)
Life Satisfaction		−0.015**	−0.014*	−0.003	−0.002	−0.0001
		(0.006)	(0.007)	(0.006)	(0.006)	(0.006)
Socio-Demographics			✓	✓	✓	✓
Attitudes toward Refugees (National)				✓	✓	✓
Attitudes toward Refugees (Local)					✓	✓
Additional Controls						✓
Observations	3,019	3,019	3,008	3,008	3,008	3,008
R^2	0.191	0.240	0.288	0.394	0.410	0.459
Adjusted R^2	0.191	0.240	0.267	0.371	0.382	0.431
Note:						*p<0.1; **p<0.05; ***p<0.01

Table 1: Mate Competition Predicts Support for Hate Crime. *Note*: OLS with standard errors in parentheses (results are very similar when using ordered logit). The dependent variable is *Only Means*, and the main independent variable is *Mate Competition* (both range from (1) disagree strongly to (4) agree strongly).

Try to code these on your own, then click for the solution

```r
lm1 <- lm(hate_violence_means ~ MateComp_cont, data=dat_use)

lm2 <- lm(hate_violence_means ~ MateComp_cont + JobComp_cont + LifeSatis_cont, data=dat_us
```

Now, let's compare the summary output of each output.

Try on your own, then click for the solution

```
summary(lm1)
```

```

Call:
lm(formula = hate_violence_means ~ MateComp_cont, data = dat_use)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.6804 -0.3694 -0.3694  0.6306  2.6306 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.93235    0.03302   28.24   <2e-16 ***
MateComp_cont  0.43702    0.01635   26.73   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7993 on 3017 degrees of freedom
Multiple R-squared:  0.1915,	Adjusted R-squared:  0.1912 
F-statistic: 714.6 on 1 and 3017 DF,  p-value: < 2.2e-16
```

```
summary(lm2)
```

```

Call:
lm(formula = hate_violence_means ~ MateComp_cont + JobComp_cont + 
    LifeSatis_cont, data = dat_use)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.8275 -0.4783 -0.1842  0.3171  2.8452 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)     0.788623   0.057849   13.63   <2e-16 ***
MateComp_cont   0.263437   0.020261   13.00   <2e-16 ***
JobComp_cont    0.249956   0.018672   13.39   <2e-16 ***
LifeSatis_cont -0.014725   0.006292   -2.34   0.0193 *  
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7751 on 3015 degrees of freedom
Multiple R-squared:  0.2403,    Adjusted R-squared:  0.2395
F-statistic: 317.9 on 3 and 3015 DF,  p-value: < 2.2e-16
```

Questions about the output

- How should we interpret the coefficients?
  - Do they support the researchers' hypotheses?
- How would we extract confidence intervals from the coefficients?
- How should we interpret the goodness of fit statistics at the bottom of the output?

Additional Models We can also run regressions with even more covariates, as the authors do in models 3-6 in the paper.

Click to reveal regression code below.

```
lm3 <- lm(hate_violence_means ~ MateComp_cont + JobComp_cont + LifeSatis_cont +
            factor(age_group) +     # age group
            factor(gender) +     # gender
            factor(state) +     # state
            factor(citizenship) +    # german citizen
            factor(marital) +    # marital status
            factor(religion) +    # religious affiliation
            eduyrs +    # education
            factor(occupation) +    # main activity
            factor(income) +   # income
            factor(household_size) +   # household size
            factor(self_econ),    # subjective social status
          data=dat_use)

lm4 <- lm(hate_violence_means ~ MateComp_cont + JobComp_cont + LifeSatis_cont +
            factor(age_group) +   # age group
            factor(gender) +   # gender
            factor(state) +   # state
            factor(citizenship) +  # german citizen
            factor(marital) +  # marital status
            factor(religion) +  # religious affiliation
            eduyrs +  # education
            factor(occupation) +  # main activity
            factor(income) + # income
            factor(household_size) + # household size
            factor(self_econ) + # subjective social status
            factor(ref_integrating) + # Refugee Index (National-level; Q73) 8 in total
            factor(ref_citizenship) + factor(ref_reduce) + factor(ref_moredone) + factor(r
            factor(ref_economy) + factor(ref_crime) + factor(ref_terror),
          data=dat_use)

lm5 <- lm(hate_violence_means ~ MateComp_cont + JobComp_cont + LifeSatis_cont +
            factor(age_group) +      # age group
            factor(gender) +      # gender
            factor(state) +      # state
            factor(citizenship) +     # german citizen
            factor(marital) +     # marital status
            factor(religion) +     # religious affiliation
            eduyrs + # education
            factor(occupation) +     # main activity
            factor(income) +    # income
            factor(household_size) +    # household size
            factor(self_econ) +    # subjective social status
            factor(ref_integrating) + # Refugee Index (National-level; Q73) 8 in total
            factor(ref_citizenship) + factor(ref_reduce) + factor(ref_moredone) + factor(r
            factor(ref_economy) + factor(ref_crime) + factor(ref_terror) +
            factor(ref_loc_services) +    # Refugee Index (Local, Q75)
            factor(ref_loc_economy) + factor(ref_loc_crime) + factor(ref_loc_culture) + fa
            factor(ref_loc_schools) + factor(ref_loc_housing) + factor(ref_loc_wayoflife),
          data=dat_use)

formula.5 <-
  as.character("hate_violence_means ~ MateComp_cont + JobComp_cont +
               LifeSatis_cont +  factor(age_group) + factor(gender) +
               factor(state) + factor(citizenship) + factor(marital) +
               factor(religion) + eduyrs + factor(occupation) +
               factor(income) + factor(household_size) + factor(self_econ) +
               factor(ref_integrating) + factor(ref_citizenship) + factor(ref_reduce) +
               factor(ref_moredone) + factor(ref_cultgiveup) +
               factor(ref_economy) + factor(ref_crime) + factor(ref_terror)  +
               factor(ref_loc_services) +  factor(ref_loc_economy) + factor(ref_loc_crime)
               factor(ref_loc_culture) + factor(ref_loc_islam) +
               factor(ref_loc_schools) + factor(ref_loc_housing) + factor(ref_loc_wayoflif
```

```r
formula.6 <- paste(formula.5, "factor(distance_ref) + factor(settle_ref)",
                   "lrscale + afd + muslim_ind + afd_ind + contact_ind",
                   sep="+", collapse="+")

lm6 <- lm(as.formula(formula.6), data=dat_use)
```

Table 1: Mate Competition Predicts Support for Hate Crime.

Model 1

Model 2

Model 3

Model 4

Model 5

Model 6

(Intercept)

0.9323***

0.7886***

1.3982***

1.4437***

1.4372***

1.3878***

(0.0330)

(0.0578)

(0.2293)

(0.2296)

(0.2388)

(0.2372)

MateComp__cont

0.4370***

0.2634***

0.2361***

0.2064***

0.1848***

0.1550***

(0.0163)

(0.0203)

(0.0206)

(0.0194)

(0.0195)

(0.0189)

JobComp_cont

0.2500***

0.2358***

0.0772***

0.0650***

0.0559***

(0.0187)

(0.0189)

(0.0195)

(0.0196)

(0.0189)

LifeSatis_cont

-0.0147**

-0.0136*

-0.0034

-0.0020

-0.0001

(0.0063)

(0.0070)

(0.0065)

(0.0065)

(0.0062)

factor(age_group)30-39

-0.1323**

-0.1800***

-0.1821***

-0.1957***

(0.0525)

(0.0489)

(0.0488)

(0.0471)

factor(age_group)40-49

-0.2088***

-0.2771***

-0.2709***

-0.2808***

(0.0525)

(0.0490)

(0.0490)

(0.0474)

factor(age_group)50-59

-0.2876***

-0.3621***

-0.3480***

-0.3580***

(0.0535)

(0.0501)

(0.0502)

(0.0486)

factor(age_group)60 and older

-0.3362***

-0.3427***

-0.3199***

-0.3073***

(0.0678)

(0.0631)

(0.0631)

(0.0610)

factor(gender)Female

-0.0247

-0.0528*

-0.0451

-0.0233

(0.0299)

(0.0281)

(0.0282)

(0.0272)

factor(state)Bayern

0.0097

-0.0168

-0.0148

-0.0229

(0.0531)

(0.0494)

(0.0491)

(0.0474)

factor(state)Berlin

0.0106

-0.0023

-0.0259

0.0037

(0.0776)

(0.0722)

(0.0720)

(0.0706)

factor(state)Brandenburg

-0.1572*

-0.1023

-0.0949

-0.1082

(0.0896)

(0.0833)

(0.0834)

(0.0805)

factor(state)Bremen

-0.1266

-0.1252

-0.1750

-0.0508

(0.1531)

(0.1423)

(0.1415)

(0.1365)

factor(state)Hamburg

-0.0208

-0.0140

-0.0255

-0.0269

(0.1016)

(0.0946)

(0.0941)

(0.0914)

factor(state)Hessen

-0.1207*

-0.0931

-0.0766

-0.0853

(0.0647)

(0.0604)

(0.0601)

(0.0578)

factor(state)Mecklenburg-Vorpommern

-0.0849

-0.1008

-0.1015

-0.1572*

(0.1035)

(0.0961)

(0.0959)

(0.0928)

factor(state)Niedersachsen

-0.0993

-0.1052*

-0.1055*

-0.1190**

(0.0607)

(0.0564)

(0.0561)

(0.0543)

factor(state)Nordrhein-Westfalen

-0.0299

-0.0277

-0.0414

-0.0414

(0.0501)

(0.0465)

(0.0465)

(0.0450)

factor(state)Rheinland-Pfalz

-0.1178

-0.1137

-0.1089

-0.1407**

(0.0750)

(0.0700)

(0.0697)

(0.0675)

factor(state)Saarland

-0.0264

0.0227

0.0353

-0.0250

(0.1293)

(0.1203)

(0.1199)

(0.1162)

factor(state)Sachsen

-0.0357

-0.0813

-0.1118

-0.1470**

(0.0734)

(0.0683)

(0.0684)

(0.0662)

factor(state)Sachsen-Anhalt

-0.0193

-0.0811

-0.0765

-0.1024

(0.0927)

(0.0862)

(0.0863)

(0.0836)

factor(state)Schleswig-Holstein

-0.2402***

-0.1693**

-0.1725**

-0.1839**

(0.0862)

(0.0806)

(0.0802)

(0.0773)

factor(state)Thringen

0.0090

-0.0076

-0.0081

-0.0654

(0.0957)

(0.0889)

(0.0887)

(0.0858)

factor(citizenship)1

-0.0621

-0.0831

-0.0739

-0.0314

(0.1064)

(0.0990)

(0.0983)

(0.0946)

factor(marital)With partner, not living together

0.0825

0.0323

0.0145

0.0099

(0.0572)

(0.0532)

(0.0529)

(0.0509)

factor(marital)With partner, living together

0.0968*

0.0570

0.0582

0.0342

(0.0562)

(0.0524)

(0.0521)

(0.0501)

factor(marital)Married

0.0884

0.0487

0.0509

0.0165

(0.0538)

(0.0500)

(0.0497)

(0.0479)

factor(marital)Registered partnership

0.0982

0.1345

0.1601

0.1976

(0.1880)

(0.1753)

(0.1744)

(0.1679)

factor(marital)Divorced / separated

0.1150*

0.0938

0.0853

0.0877

(0.0616)

(0.0573)

(0.0569)

(0.0549)

factor(marital)Widowed

0.1612*

0.1556*

0.1309

0.1243

(0.0920)

(0.0855)

(0.0855)

(0.0824)

factor(religion)Roman Catholic

-0.0034

-0.0300

-0.0333

-0.0713*

(0.0407)

(0.0378)

(0.0377)

(0.0364)

factor(religion)Protestant

-0.0640*

-0.0396

-0.0249

-0.0556*

(0.0375)

(0.0348)

(0.0349)

(0.0337)

factor(religion)Protestant Free Church

-0.0225

-0.0240

-0.0170

-0.0780

(0.1022)

(0.0951)

(0.0945)

(0.0911)

factor(religion)Other Protestant

0.7822**

0.9286***

0.9234***

0.8768**

(0.3855)

(0.3587)

(0.3565)

(0.3441)

factor(religion)Eastern Orthodox

0.2751

0.1666

0.1350

0.1455

(0.1869)

(0.1744)

(0.1735)

(0.1672)

factor(religion)Other Christian

-0.0119

0.0406

0.0645

0.0954

(0.1633)

(0.1518)

(0.1514)

(0.1458)

factor(religion)Jewish

0.0827

-0.1329

-0.0855

-0.2074

(0.3460)

(0.3217)

(0.3202)

(0.3081)

factor(religion)Muslim

-0.0578

0.0586

0.0046

0.0906

(0.1667)

(0.1554)

(0.1549)

(0.1509)

factor(religion)Eastern religion (Buddhism, Hinduism, Sikhism, Shinto, Tao, etc.)

-0.0026

0.0043

0.0289

0.0138

(0.1215)

(0.1130)

(0.1129)

(0.1086)

factor(religion)Other non-Christian religion

0.3904*

0.4759**

0.4412**

0.2675

(0.2333)

(0.2175)

(0.2168)

(0.2089)

factor(religion)Christian, but not close to a particular religious community

-0.0270

0.0102

0.0083

-0.0177

(0.0611)

(0.0567)

(0.0566)

(0.0544)

factor(religion)No answer

0.1273

0.2257**

0.2106**

0.1926*

(0.1116)

(0.1039)

(0.1036)

(0.0999)

eduyrs

-0.0179***

-0.0139***

-0.0121***

-0.0088**

(0.0042)

(0.0039)

(0.0039)

(0.0038)

factor(occupation)Parental leave

0.1940

0.1368

0.1319

0.1606

(0.1589)

(0.1478)

(0.1473)

(0.1417)

factor(occupation)In schooling / vocational training, student

-0.0690

-0.1196

-0.1279

-0.1244

(0.0926)

(0.0862)

(0.0859)

(0.0828)

factor(occupation)Unemployed / seeking work

0.0088

-0.0339

-0.0420

-0.0451

(0.1080)

(0.1004)

(0.0999)

(0.0961)

factor(occupation)Retired

0.1055

0.0493

0.0470

0.0298

(0.0816)

(0.0758)

(0.0755)

(0.0726)

factor(occupation)Permanently sick or disabled

-0.1574

-0.1490

-0.1458

-0.1507

(0.1287)

(0.1198)

(0.1194)

(0.1149)

factor(occupation)Unskilled worker

0.1800*

0.0893

0.0642

0.0221

(0.0934)

(0.0870)

(0.0865)

(0.0833)

factor(occupation)Skilled worker

0.2026**

0.1379*

0.1393*

0.1073

(0.0871)

(0.0812)

(0.0808)

(0.0778)

factor(occupation)Employee in low / medium position

0.1065

0.0560

0.0524

0.0653

(0.0740)

(0.0688)

(0.0685)

(0.0659)

factor(occupation)Employee in high position

0.0536

-0.0116

-0.0165

-0.0292

(0.0824)

(0.0766)

(0.0763)

(0.0735)

factor(occupation)Civil servant

-0.0009

-0.1061

-0.1342

-0.1687

(0.1307)

(0.1214)

(0.1209)

(0.1166)

factor(occupation)Senior civil servant

0.0173

0.0085

0.0262

-0.0256

(0.1481)

(0.1380)

(0.1372)

(0.1325)

factor(occupation)Senior civil servant <96> highest level

-0.0083

-0.1016

-0.0793

-0.0611

(0.1262)

(0.1174)

(0.1169)

(0.1127)

factor(occupation)Self-employed / freelancer

0.1171

0.0323

0.0396

0.0693

(0.0889)

(0.0828)

(0.0824)

(0.0794)

factor(occupation)Other

0.2269

0.0467

0.0232

0.0080

(0.1724)

(0.1607)

(0.1601)

(0.1540)

factor(income)500 to below 1,000 <80>

0.0260

0.0768

0.0750

-0.0028

(0.1024)

(0.0953)

(0.0948)

(0.0914)

factor(income)1,000 to below 1,500 <80>

0.0677

0.0714

0.0642

-0.0274

(0.1011)

(0.0943)

(0.0937)

(0.0904)

factor(income)1,500 to below 2,000 <80>

0.1360

0.1289

0.1319

0.0564

(0.1024)

(0.0955)

(0.0949)

(0.0914)

factor(income)2,000 to below 2,500 <80>

0.1320

0.1155

0.1146

0.0028

(0.1045)

(0.0974)

(0.0969)

(0.0935)

factor(income)2,500 to below 3,000 <80>

0.0479

0.0606

0.0615

-0.0466

(0.1071)

(0.0998)

(0.0992)

(0.0956)

factor(income)3,000 to below 3,500 <80>

0.1659

0.1531

0.1557

0.0384

(0.1108)

(0.1031)

(0.1025)

(0.0989)

factor(income)3,500 to below 4,000 <80>

0.2256**

0.2133**

0.2101**

0.0785

(0.1138)

(0.1059)

(0.1054)

(0.1017)

factor(income)4,000 to below 4,500 <80>

0.0770

0.0396

0.0271

-0.0996

(0.1211)

(0.1127)

(0.1121)

(0.1081)

factor(income)4,500 to below 5,000 <80>

0.2446*

0.1782

0.1755

0.0431

(0.1274)

(0.1188)

(0.1182)

(0.1140)

factor(income)5,000 or more <80>

0.2017

0.1350

0.1128

0.0250

(0.1227)

(0.1143)

(0.1136)

(0.1095)

factor(income)No answer

0.0325

0.0498

0.0501

-0.0453

(0.1062)

(0.0990)

(0.0984)

(0.0949)

factor(household_size)2

0.0362

0.0390

0.0316

0.0617

(0.0511)

(0.0475)

(0.0473)

(0.0457)

factor(household_size)3

0.0404

0.0374

0.0403

0.0675

(0.0574)

(0.0535)

(0.0533)

(0.0515)

factor(household_size)4

0.0289

0.0129

0.0114

0.0516

(0.0647)

(0.0602)

(0.0599)

(0.0578)

factor(household_size)5

0.0161

0.0255

0.0347

0.0345

(0.1046)

(0.0972)

(0.0968)

(0.0934)

factor(household_size)6

0.3162*

0.3629**

0.4030**

0.3646**

(0.1793)

(0.1666)

(0.1658)

(0.1599)

factor(household_size)7

0.0387

0.0311

0.0495

-0.0145

(0.3181)

(0.2957)

(0.2939)

(0.2838)

factor(household_size)8

0.7654**

0.9534***

0.8352**

0.7004**

(0.3876)

(0.3615)

(0.3615)

(0.3479)

factor(household_size)12

-0.0289

0.0946

-0.0435

-0.1011

(0.7761)

(0.7217)

(0.7188)

(0.6915)

factor(self_econ)2

-0.0271

-0.1166

-0.1141

-0.0368

(0.1629)

(0.1517)

(0.1511)

(0.1457)

factor(self_econ)3

-0.2019

-0.2069

-0.2058

-0.1859

(0.1513)

(0.1408)

(0.1405)

(0.1353)

factor(self_econ)4

-0.1501

-0.1394

-0.1358

-0.1343

(0.1510)

(0.1405)

(0.1403)

(0.1351)

factor(self_econ)5

-0.2279

-0.1700

-0.1705

-0.1569

(0.1491)

(0.1389)

(0.1386)

(0.1335)

factor(self_econ)6

-0.2812*

-0.2191

-0.2186

-0.2051

(0.1503)

(0.1400)

(0.1397)

(0.1345)

factor(self_econ)7

-0.3444**

-0.2527*

-0.2484*

-0.2383*

(0.1518)

(0.1415)

(0.1414)

(0.1360)

factor(self_econ)8

-0.2107

-0.1598

-0.1765

-0.1973

(0.1573)

(0.1466)

(0.1462)

(0.1407)

factor(self_econ)9

-0.1747

-0.0476

-0.0684

-0.0658

(0.1933)

(0.1804)

(0.1801)

(0.1731)

factor(self_econ)10 (TOP)

0.3679

0.2960

0.2701

0.2253

(0.2349)

(0.2192)

(0.2183)

(0.2100)

factor(self_econ)0 (BOTTOM)

-0.0023

-0.0450

-0.0278

0.0017

(0.2077)

(0.1933)

(0.1925)

(0.1853)

factor(ref_integrating)2

-0.0585

-0.0390

-0.0304

(0.0913)

(0.0924)

(0.0890)

factor(ref_integrating)3

-0.0921

-0.0692

-0.0772

(0.0935)

(0.0951)

(0.0918)

factor(ref_integrating)4

0.0787

0.0928

0.0585

(0.0998)

(0.1015)

(0.0980)

factor(ref_citizenship)2

0.0020

-0.0202

-0.0245

(0.0444)

(0.0447)

(0.0429)

factor(ref_citizenship)3

0.0893*

0.0720

0.0349

(0.0493)

(0.0497)

(0.0480)

factor(ref_citizenship)4

0.1626***

0.1425**

0.1000*

(0.0571)

(0.0581)

(0.0561)

factor(ref_reduce)2

-0.0253

-0.0103

-0.0041

(0.0595)

(0.0599)

(0.0576)

factor(ref_reduce)3

0.0162

0.0326

-0.0106

(0.0636)

(0.0644)

(0.0623)

factor(ref_reduce)4

0.0354

0.0465

-0.1045

(0.0744)

(0.0754)

(0.0736)

factor(ref_moredone)2

0.0930*

0.0802*

0.0559

(0.0482)

(0.0484)

(0.0467)

factor(ref_moredone)3

0.1947***

0.1834***

0.0920*

(0.0528)

(0.0532)

(0.0520)

factor(ref_moredone)4

0.3050***

0.2874***

0.1561**

(0.0618)

(0.0623)

(0.0609)

factor(ref_cultgiveup)2

-0.0125

-0.0212

-0.0309

(0.0586)

(0.0590)

(0.0569)

factor(ref_cultgiveup)3

0.0145

-0.0054

-0.0505

(0.0582)

(0.0587)

(0.0568)

factor(ref_cultgiveup)4

0.1507**

0.1281*

0.0733

(0.0652)

(0.0656)

(0.0636)

factor(ref_economy)2

0.0237

0.0456

0.0510

(0.0549)

(0.0582)

(0.0561)

factor(ref_economy)3

-0.0004

0.0343

0.0145

(0.0606)

(0.0663)

(0.0638)

factor(ref_economy)4

0.1657**

0.2379***

0.1524**

(0.0702)

(0.0778)

(0.0750)

factor(ref_crime)2

0.0183

-0.0033

-0.0037

(0.0606)

(0.0645)

(0.0620)

factor(ref_crime)3

0.0794

-0.0061

-0.0290

(0.0661)

(0.0715)

(0.0688)

factor(ref_crime)4

0.2506***

0.1343

0.0431

(0.0774)

(0.0835)

(0.0806)

factor(ref_terror)2

-0.0689

-0.0975*

-0.1060*

(0.0568)

(0.0578)

(0.0556)

factor(ref_terror)3

-0.0330

-0.0818

-0.1054*

(0.0608)

(0.0617)

(0.0595)

factor(ref_terror)4

-0.0338

-0.0865

-0.1144*

(0.0707)

(0.0715)

(0.0689)

factor(ref_loc_services)2

0.0852

0.0888

(0.0680)

(0.0653)

factor(ref_loc_services)3

0.0765

0.0788

(0.0681)

(0.0655)

factor(ref_loc_services)4

0.0577

0.0699

(0.0761)

(0.0732)

factor(ref_loc_economy)2

-0.1186*

-0.1209*

(0.0718)

(0.0690)

factor(ref_loc_economy)3

-0.1420*

-0.1562**

(0.0767)

(0.0738)

factor(ref_loc_economy)4

-0.2892***

-0.2972***

(0.0864)

(0.0832)

factor(ref_loc_crime)2

0.0813

0.0727

(0.0582)

(0.0560)

factor(ref_loc_crime)3

0.2474***

0.2050***

(0.0662)

(0.0640)

factor(ref_loc_crime)4

0.3110***

0.2766***

(0.0800)

(0.0774)

factor(ref_loc_culture)2

0.0051

0.0068

(0.0524)

(0.0507)

factor(ref_loc_culture)3

0.0297

0.0078

(0.0612)

(0.0594)

factor(ref_loc_culture)4

0.1232*

0.0013

(0.0737)

(0.0718)

factor(ref_loc_islam)2

-0.0128

-0.0085

(0.0531)

(0.0511)

factor(ref_loc_islam)3

-0.0085

-0.0453

(0.0550)

(0.0531)

factor(ref_loc_islam)4

-0.0281

-0.1068*

(0.0660)

(0.0638)

factor(ref_loc_schools)2

0.1254

0.1084

(0.0887)

(0.0853)

factor(ref_loc_schools)3

0.0552

0.0573

(0.0832)

(0.0801)

factor(ref_loc_schools)4

-0.0806

-0.0809

(0.0845)

(0.0814)

factor(ref_loc_housing)2

0.0134

0.0095

(0.0586)

(0.0566)

factor(ref_loc_housing)3

0.0068

0.0008

(0.0564)

(0.0547)

factor(ref_loc_housing)4

0.0432

0.0433

(0.0608)

(0.0590)

factor(ref_loc_wayoflife)2

-0.0586

-0.0653

(0.0621)

(0.0597)

factor(ref_loc_wayoflife)3

-0.0341

-0.0515

(0.0609)

(0.0586)

factor(ref_loc_wayoflife)4

0.0694

0.0550

(0.0707)

(0.0682)

factor(distance_ref)3-5 kilometers

-0.0375

(0.0362)

factor(distance_ref)6-10 kilometers

0.0256

(0.0405)

factor(distance_ref)11-20 kilometers

0.0165

(0.0494)

factor(distance_ref)21-50 kilometers

0.0644

(0.0534)

factor(distance_ref)More than 50 kilometer

0.0568

(0.0794)

factor(distance_ref)Don’t know

-0.0389

(0.0418)

factor(settle_ref)1 <96> 49

-0.0133

(0.0668)

factor(settle_ref)50 <96> 249

-0.0178

(0.0661)

factor(settle_ref)250 <96> 499

-0.0455

(0.0694)

factor(settle_ref)500 <96> 999

-0.0213

(0.0732)

factor(settle_ref)1000 and more

-0.0536

(0.0687)

lrscale

0.0235***

(0.0078)

afd

0.0044***

(0.0006)

muslim_ind

0.3152***

(0.0701)

afd_ind

0.3390***

(0.0489)

contact_ind

0.0741

(0.0522)

R2

0.1915

0.2403

0.2883

0.3942

0.4097

0.4592

Adj. R2

0.1912

0.2395

0.2673

0.3712

0.3821

0.4308

Num. obs.

3019

3019

3008

3008

3008

3008

***p < 0.01; **p < 0.05; *p < 0.1

Final Questions

Even with all these covariates accounted for, the authors still engage in a discussion about possible violations of the OLS assumptions that could bias their results, as well as potential alternative modelling strategies.

- Is their survey representative? They replicate using another polling firm.
- Are there even more alternative explanations?
- Is OLS the right choice?
- Validity (discussed in Gelman and Hill). Does the outcome accurately measure the concept? They consider alternative outcomes and visualize the coefficient results in Figure 4.
    - Message: Attacks against refugee homes are sometimes necessary to make it clear to politicians that we have a refugee problem.
    - Justified : Hostility against refugees is sometimes justified, even when it ends up in violence.
    - Prevent : Xenophobic acts of violence are defensible if they result in fewer refugees settling in town.
    - Condemn: Politicians should condemn attacks against refugees more forcefully.

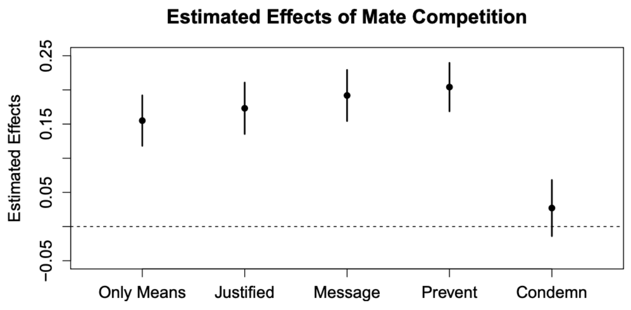


Figure 4: Estimated Effects of Mate Competition on Support for Hate Crime. *Note*: Outcome variables range from 1 to 4 with higher values corresponding to stronger hate crime support (or stronger opposition to elite sanctions). The main predictor, *Mate Competition*, ranges from (1) disagree strongly to (4) agree strongly. For each outcome, we present the effect of changing *Mate Competition* by one unit. Results are based on the specification as shown in Table 1 column 6 (with varying dependent variables).

Additional Practice Questions.

1. Find the average expected level of "Only Means" agreement at each level of mate competition. Plot the results. Base these results on `lm2`.

2. Fit `lm2` using the generalized linear model `glm` approach (with a normal distribution) instead of the `lm`
3. What are some of the conceptual differences between ordinary least squares and maximum likelihood estimation?