

Data Science for Political Science

2023-09-01

Section Contents

Course Notes	6
1 Introduction	7
1.1 What have I signed up for?	7
1.1.1 Data Science Can Help Social Scientists	8
1.1.2 Course Goals	10
1.2 Setup in R	11
1.3 Open R Script in RStudio	12
1.3.1 Using R as a Calculator	14
1.3.2 Working in an R Script	15
1.3.3 Saving the R Script	15
1.3.4 Annotating your R script	16
1.3.5 Running Commands in your R script	18
1.3.6 Objects	19
1.4 R Markdown	21
1.4.1 Getting started with RMarkdown	22
1.5 Assignment 1	27
2 Description	29
2.1 Process of Describing	29
2.1.1 Example Process	30
2.2 Summarizing univariate data	30
2.3 Functions to summarize univariate data	33
2.3.1 Using functions in R (overview)	33
2.4 Loading data into R	34
2.4.1 Working with datasets in R	35
2.4.2 Measuring the Turnout in the US Elections	37
2.4.3 Getting to know your data	38
2.5 Comparing VEP and VAP turnout	40
2.5.1 Creating new variables in R	40
2.6 Comparing Presidential vs. Midterm turnout	42
2.6.1 R shortcut for writing vectors	43
2.7 Creating dataframes from within R	44
2.8 Wrapping Up Description	45
2.8.1 Summary of R tools	46

3 Causation with Experiments	48
3.1 What separates causation from correlation?	48
3.1.1 Potential Outcomes Framework	49
3.1.2 Causal Effects	51
3.1.3 Fundamental Problem of Causal Inference	52
3.2 Randomized Controlled Trials	52
3.2.1 Experiments: Why Randomize?	53
3.2.2 Experiments: How to Analyze	54
3.2.3 Ingredients of an Experiment	54
3.3 Application: Is there racial discrimination in the labor market?	55
3.3.1 Variable classes	56
3.4 Making tables	58
3.4.1 Crosstabulation	58
3.5 Conditional Means	58
3.6 Relational Operators in R	59
3.7 Subsetting data in R	60
3.7.1 Getting Boooooooooolean	61
3.8 Creating New Variables using Conditional statements	61
3.8.1 ifelse statements	63
3.9 Types of Experiments	63
3.10 Wrapping Up Causation with Experiments	65
3.10.1 Summary of R tools in this section	65
4 Visualization	67
4.1 Application: Social Status and Economic Views	68
4.2 Boxplots	69
4.2.1 Data Summary: Boxplot	70
4.3 Barplots	73
4.3.1 Saving Plots	76
4.3.2 Creating New Variables	77
4.4 Application: Changing Minds on Gay Marriage	79
4.4.1 Recall: Creating new variables	81
4.4.2 Recall: Using ifelse to create new variable	81
4.4.3 Calculating the Average Treatment Effect	82
4.4.4 Visualize means in a barplot	82
4.5 Scatterplots	83
4.6 Histograms	86
4.6.1 Happy research ending	88
4.7 Line Plots	89
4.8 Application: Trends during COVID	91
4.9 Visual tips and tricks	102
4.10 Common R plotting functions and arguments	105
4.11 A note on ggplot	106

5 Causality with Non-Experimental Data	114
5.1 Why can't we always experiment?	114
5.1.1 What can we do instead?	115
5.2 Causal Identification Strategies	115
5.2.1 Example: Travis Kelce Jersey Sale and Instagram Gains	116
5.2.1.1 Line Plots	116
5.2.2 Example: Social Mobility Data	118
5.2.3 Causal claims from before vs. after comparisons	129
5.3 Three Common Identification Strategies	133
5.3.1 Threats to Cross-Section Designs	133
5.3.2 Threats to Before-After Designs	134
5.3.3 Threats to Diff-in-Diff Designs	134
5.4 Application: Economic Effects of Basque Terrorism	135
5.4.1 Applying 3 Identification Strategies	135
5.5 Placebo Tests	137
5.6 Wrapping Up Causality	138
6 Loops in R	139
6.1 The anatomy of a loop	140
6.1.1 The key parts of a loop	141
6.1.2 A short example	142
6.1.3 Troubleshooting a loop	143
6.1.4 Your turn	144
6.2 Application: U.S. Supreme Court	144
6.2.1 Troubleshooting the loop	151
6.2.2 Visualizing the Results	152
6.2.3 Enhancing the plot	154
6.2.4 Wrapping Up	157
7 Prediction	159
7.1 Prediction Overview	160
7.2 Process of Prediction	163
7.3 Example: Forecasting 2020 US Election based on 2016 Results	164
7.3.1 Choose Approach	165
7.3.2 Assess Accuracy	166
7.3.3 Iterate to improve predictions	167
7.4 Example: Using polls to predict the 2020 election results	167
7.4.1 Choose an approach: Let's analyze some polls	169
7.4.2 Loop through all states	171
7.4.3 Check Accuracy	172
7.4.3.1 Quantitative Measures of Accuracy	172
7.4.3.2 Classification	176

8 Prediction with Regression	178
8.1 Regression in the wild.	178
8.2 Application: Baseball Predictions	180
8.3 Step 1: Approach- Regression in R	184
8.3.1 Visualizing a regression	186
8.3.2 Making predictions with regression	187
8.4 Step 2: Checking accuracy of model	189
8.4.1 Root Mean Squared Error	191
8.5 Step 3: Iterate and Compare Models	192
8.5.1 Regression with Multiple Predictors	192
8.6 Application: Predicting Campaign Donations	192
8.6.1 Visualizing the results	194
8.6.2 Step 1: Calculate Predictions	195
8.6.3 Step 2: Check Accuracy	195
8.6.4 Step 3: Iterate	196
8.6.5 Adding Model Predictors	196
8.7 Uncertainty with Prediction	198
8.7.1 Example: Butterfly Ballot in Florida	198
8.7.2 Multiple Predictors	200
8.7.3 Confidence Intervals	202
8.8 Cross-Validation	204
8.8.1 Cross-Validation Process	205
8.8.2 Application: Forecasting Election Results	206
8.8.2.1 Steps 1 and 2	206
8.8.2.2 Step 3: Predict and assess accuracy with test data	207
8.8.2.3 Step 4: Repeat process across all data	207
8.8.2.4 Step 5: Summarize performance	208
8.8.2.5 Applying Model to New Data	208
8.8.2.6 Challenge	209

Course Notes

This document will include important links and course notes for 01:790:391:01: Data Science for Political Science for the fall 2023 semester.

- This site will be updated throughout the semester with new content.
- The Canvas modules will provide links to the relevant sections to review for a given week of the course.
- The primary text for the course is [Quantitative Social Science: An Introduction](#) by Kosuke Imai. We will refer to this as QSS in the notes.
- This is a living document. If you spot errors or have questions or suggestions, please email me at k.mccabe@rutgers.edu or post to the course Canvas site.
- Occasionally the notes are updated with embedded video explainers of portions of the code in different sections. These will be located in the playlist linked [here](#).

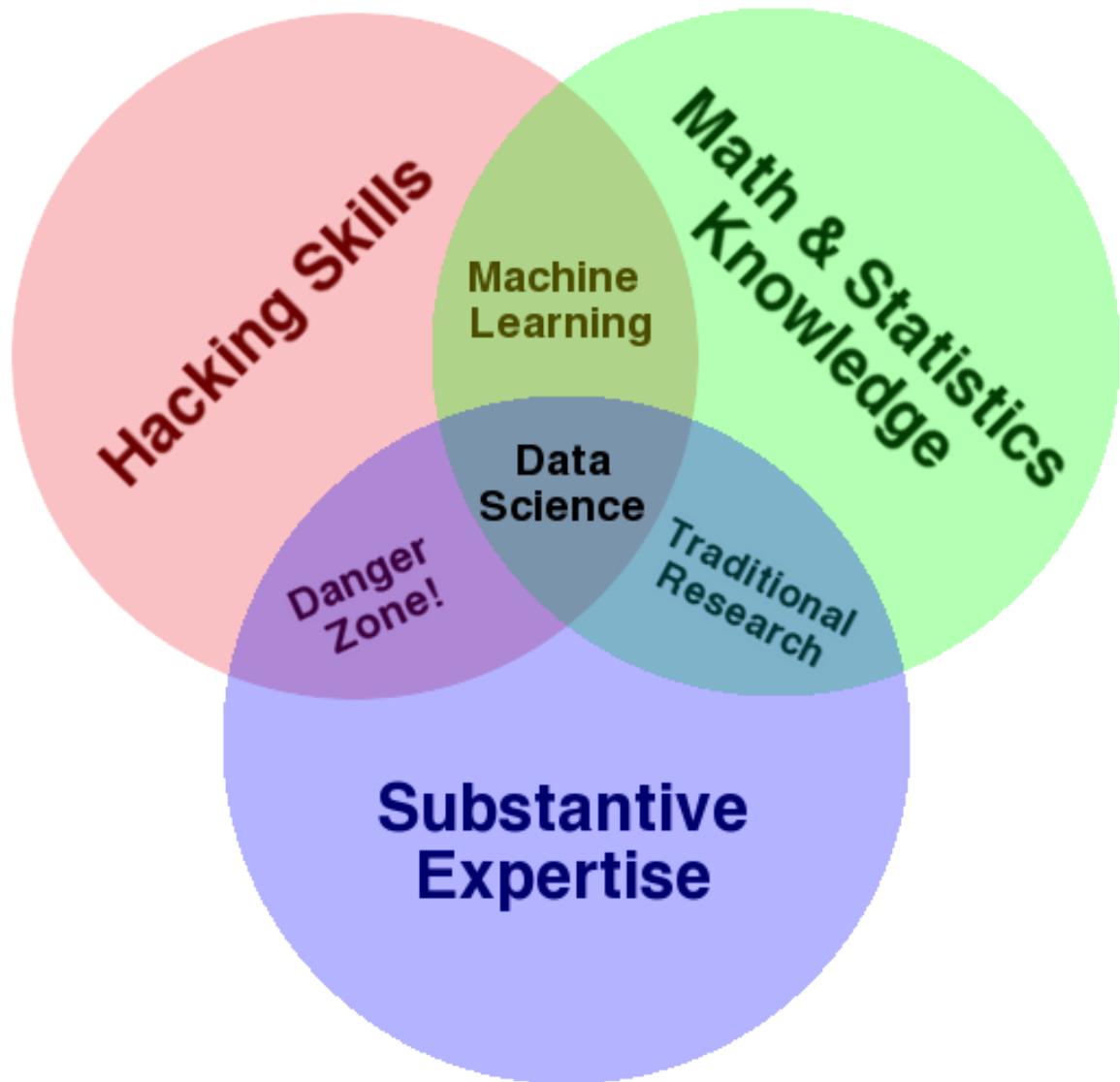
In addition, the chat bot below can be used to ask questions about material included on lecture slides posted on Canvas as well as the pdf version of this website, which can be accessed using the pdf icon in the top-left sidebar of this page.

1 Introduction

1.1 What have I signed up for?

First: What is Data Science?

- Data Science involves a combination of math/statistics and programming/coding skills, which, for our purposes, we will combine with social science knowledge.
 - [Drew Conway](#) has a nice venn diagram of how these different skill sets intersect.
 - Note: This course will not assume prior familiarity with data science in general or coding, specifically. For those brand new to data science, the idea of learning to code may seem intimidating, but anyone can succeed with a bit of patience and an open mind.



Next: What is political science?

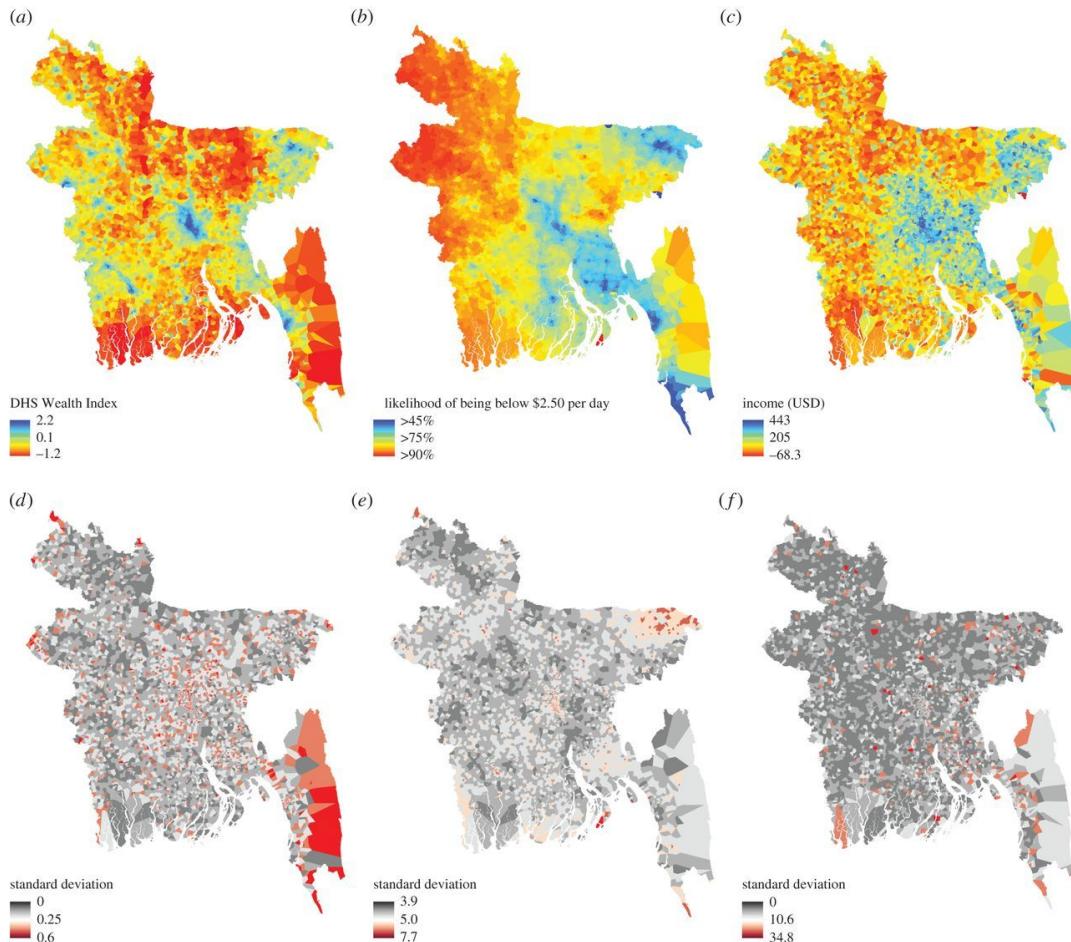
- The science of politics, of course! Politics focuses on studying governance and the distribution of power in society, broadly conceived.
 - How else might you define politics and political science? What do we study in political science?

1.1.1 Data Science Can Help Social Scientists

Example: Mapping poverty using mobile phone and satellite data

Researchers used modern data sources, including mobile phone data, as a way to *predict* and *describe* poverty in different geographic regions. These tools helped social scientists come up with methods that are much more cost-effective and efficient, but still as accurate as traditional methods for this type of measurement.

- How might measures of global poverty be useful to political scientists?



[Steele et al. 2017](#): “Poverty is one of the most important determinants of adverse health outcomes globally, a major cause of societal instability and one of the largest causes of lost human potential. Traditional approaches to measuring and targeting poverty rely heavily on census data, which in most low- and middle-income countries (LMICs) are unavailable or out-of-date. Alternative measures are needed to complement and update estimates between censuses. This study demonstrates how public and private data sources that are commonly available for LMICs can be used to provide novel insight into the spatial distribution of poverty. We evaluate the relative value of modelling three traditional poverty measures using aggregate data from mobile operators and widely available geospatial data.”

1.1.2 Course Goals

Social Science Goals

We have several goals in social science. Here are four that data science can help us pursue:

- **Describe** and measure
 - Has the U.S. population increased?
- **Explain**, evaluate, and recommend (study of causation)
 - Does expanding Medicaid improve health outcomes?
- **Predict**
 - Who will win the next election?
- **Discover**
 - How do policies diffuse across states?

What are other examples of these goals?

Note: In this course, we are exploiting the benefits of quantitative data to help achieve goals of social science. However, quantitative data have their shortcomings, too. We will also discuss the limitations of various applications of social science data, and we encourage you to always think critically about how we are using data.

This course will provide you with a taste of each of these social science goals, and how the use of data can help achieve these goals. By the end of the course, you should be able to

- Provide examples of how quantitative data may be used to help answer social science research questions.
- Compare and contrast the goals of description, causation, prediction, and discovery in social science research.
- Use the programming language R to import and explore social science data and conduct basic statistical analyses.
- Interpret and describe visual displays of social science data, such as graphs and maps.
- Develop your own analyses and visualizations to understand social science phenomena.

If you are someone that loves data, we hope you will find this course engaging. If you are someone who loathes or finds the idea of working with data and statistics alarming, we hope you keep an open mind. We will meet you where you are. This course will not assume knowledge of statistical software, and there will be plenty of opportunities to ask questions and seek help from classmates and the instructor throughout the semester.

The first section of course will walk people through how to use the statistical program– R– that we will employ this semester.

Will this course help me in the future?

Even if you do not plan on becoming a social scientist or a data scientist, an introduction to these skills may prove helpful throughout your academic and professional careers.

- To become an informed consumer of news articles and research involving quantitative analyses.
- To practice analytical thinking to make informed arguments and decisions.
- To expand your toolkit for getting a job that may involve consuming or performing some data analysis, even if that is not the traditional role.
 - Example: Journalism- [How 5 Data Dynamos Do Their Jobs](#)

1.2 Setup in R

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), feel comfortable using R as a calculator, and making documents using the R Markdown file type within RStudio.

R is an application that processes the R programming language. RStudio is also an application, which serves as a user interface that makes working in R easier. We will primarily open and use RStudio to work with R.

In other classes, you may come across Stata, SPSS, Excel, or SAS, which are programs that also conduct data analysis. R has the advantage of being free and open-source. Even after you leave the university setting, you will be able to use R/RStudio for free. As an open-source program, it is very flexible, and a community of active R/RStudio users is constantly adding to and improving the program. You might also encounter the Python language at some point. R and Python have similarities, and learning R can also make learning Python easier down the road.

R and RStudio Installation

This content follows and reinforces section QSS 1.3 in our book. Additional resources are also linked below.

- This [video](#) from Professor Christopher Bail explains why many social scientists use R and describes 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.

<https://www.youtube.com/watch?v=uIIv0NiVTs4>

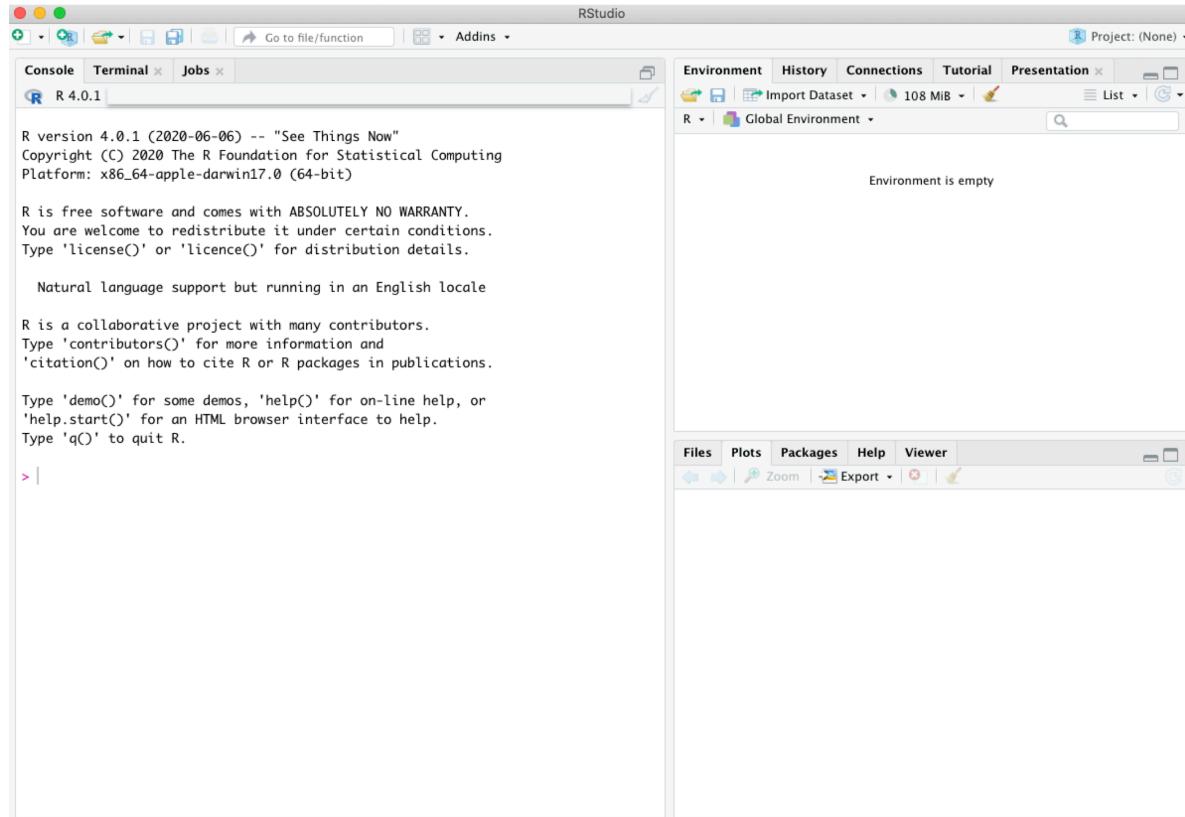
1.3 Open R Script in RStudio

This next section provides a few notes on using R and RStudio now that you have installed it. 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 (.R file)
- Creating, saving, and compiling an R Markdown document (.Rmd) into an html document (.html)

This section highlights important concepts from QSS chapter 1.

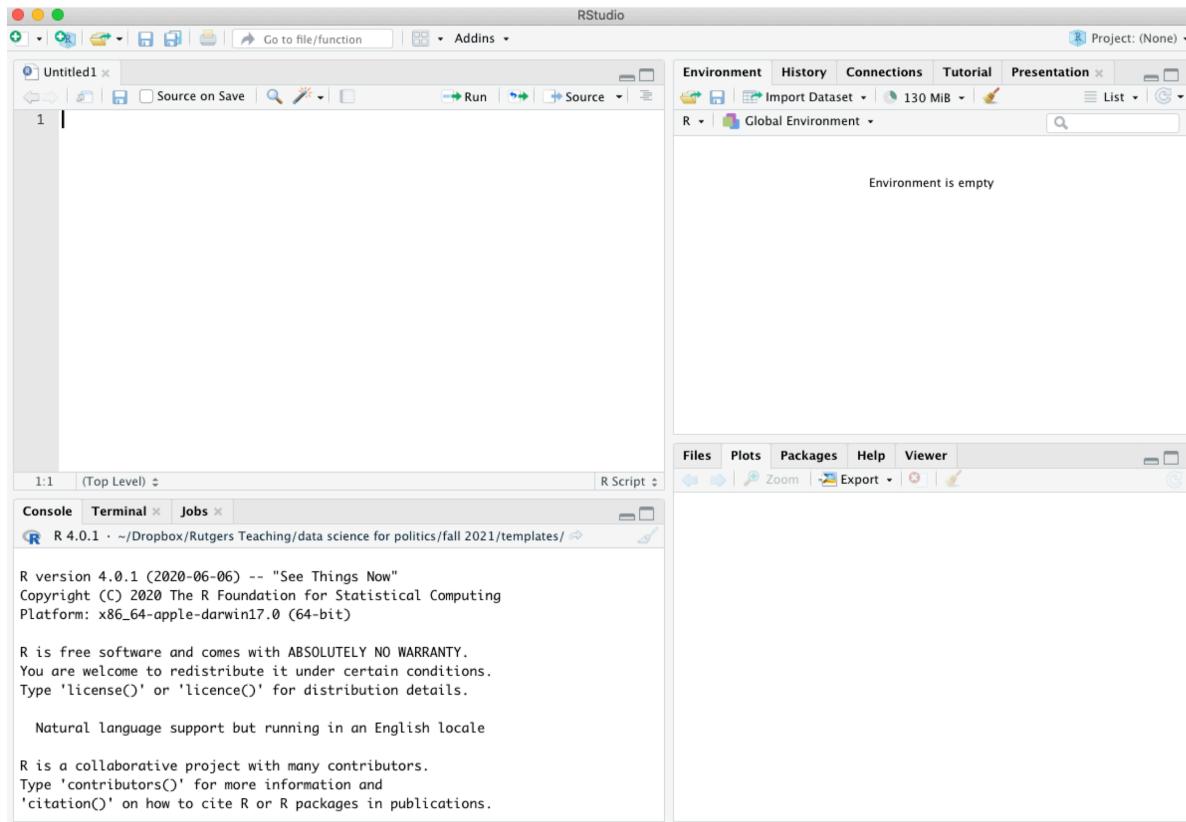
RStudio is an open-source and free program that greatly facilitates the use of R, especially for users new to programming. Once you have downloaded and installed R and RStudio, to work in R, all you need to do now is **open RStudio** (it will open R). It should look like this, though your version numbers will be different:



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:



- 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.

1.3.1 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
```

```
5 - 3
```

```
[1] 2
```

```
5^2
```

```
[1] 25
```

```
5 * 3
```

```
[1] 15
```

```
5/3
```

```
[1] 1.666667
```

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

```
[1] 16
```

Again, 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.

1.3.2 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()`

Many times 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()
```

```
[1] "/Users/ktmccabe/Dropbox/GitHub2/dsps23"
```

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/")
```

1.3.3 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.

1.3.4 Annotating 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— annotations that explain what your code is doing. 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()`

```

1 ##########
2 ## Problem Set XX #####
3 ## Name: Your name #####
4 ## People you worked with: #####
5 ##########
6
7 # enter the path of your working directory
8 setwd()
9
10
11 #####
12 # Problem 1
13 #####
14
15 ## add comments like this to help explain your steps
16
17 # I added two numbers
18 sum53 <- 5 + 3
19 sum53
20
21 #####
22 # Problem 2
23 #####
24
25
26 #####
27 # Problem 3
28 #####
29
30

```

- 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.

1.3.5 Running Commands in your R script

The last thing we will note in this section 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.

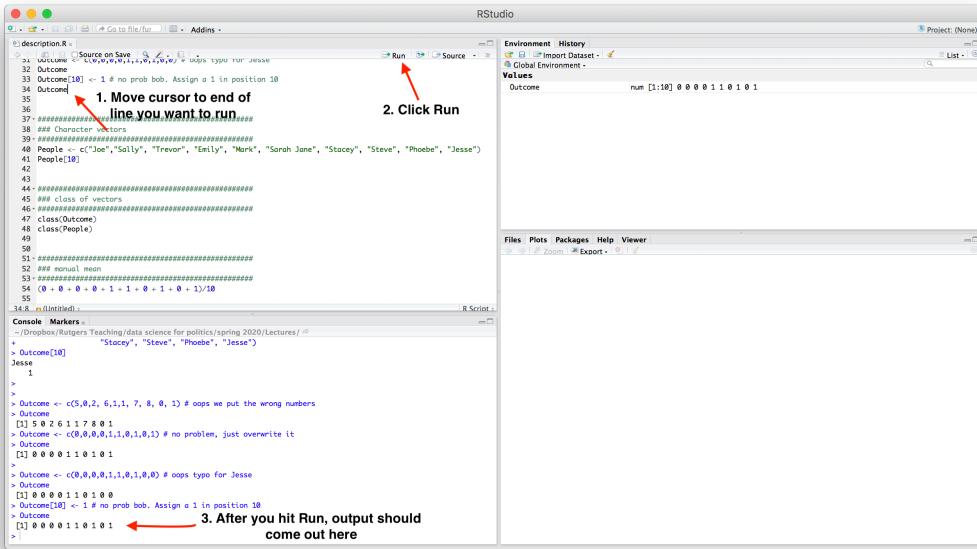
Try it: Type 5 + 3 in the R script. Then, try to execute 5 + 3. It should look something like this:

```
1 - #####
2 - ## Problem Set 1 #####
3 - ## Name: Katherine McCabe #####
4 - ## People you worked with: Just me #####
5 - #####
6
7 ## enter the path of your working directory
8 setwd("/Users/ktmccabe/Dropbox/Rutgers Teaching/data science for politics")
9
10
11 - #####
12 ## Problem 1 #####
13 - #####
14
15 ## Add 5 + 3
16 5 + 3
```

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

```
5 + 3
```

```
[1] 8
```



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.

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

1.3.6 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`:

```
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:

```
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:

```
sum53 - 2
```

```
[1] 6
```

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

```
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.

```
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()`

```
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:

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

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

```
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:

```
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 @thosleeper · Sep 17

Data science is 90% fixing punctuation and 10% maximum likelihood estimation.

1.4 R Markdown

An R Markdown document, which you can also create in RStudio, allows you to weave together regular text, R code, and the output of R code in the same document. This can be very convenient when conducting data analysis because it allows you more space to explain what you are doing in each step. We will use it as an effective platform for writing up problem sets.

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
```

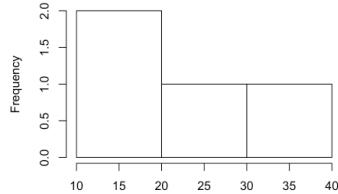
Written answer: The answer to this is 8.

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")
```

Toy plot



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.

```
# 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")
```

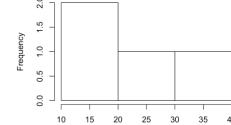
```

Problem 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")
```

Toy plot



1.4.1 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>.

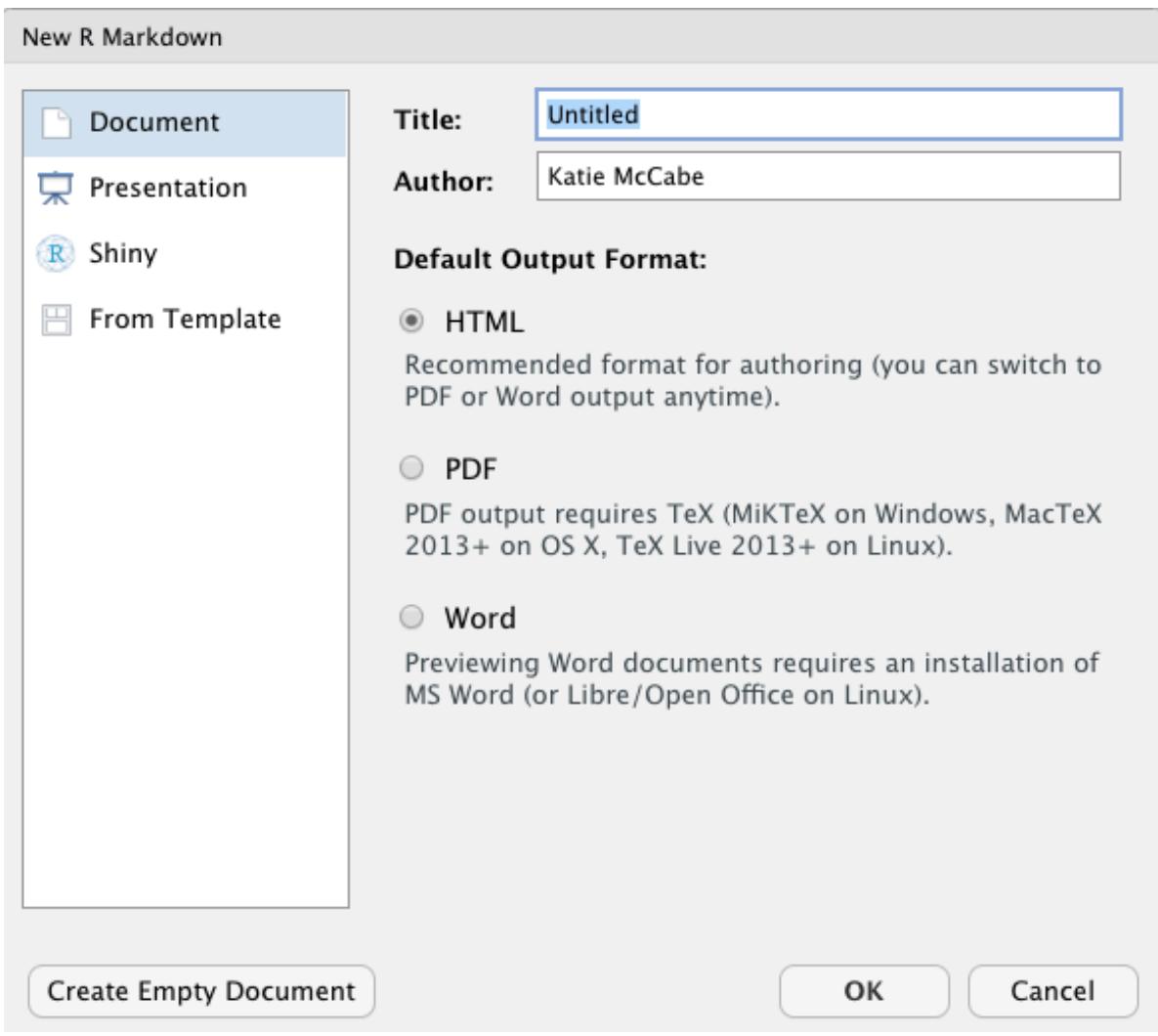
The **first time** you will be working in R Markdown, you will want to install two packages: `rmarkdown` and `knitr`. You can do this in the Console window in RStudio (remember the lower-left window!).

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

```
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.



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.

```

1 ---  

2 title: "Problem Set 1"  

3 author: "Katie McCabe"  

4 date: "9/7/2021"  

5 output: html_document  

6 ---  

7  

8 ```{r setup, include=FALSE}  

9 knitr::opts_chunk$set(echo = TRUE)  

10 ``````  

11  

12 ## R Markdown  

13  

14 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.  

15  

16 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:  

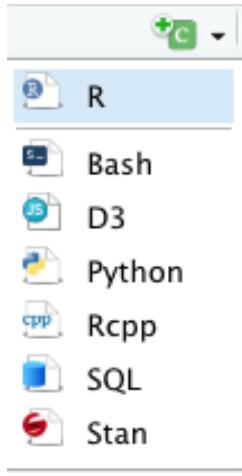
17  

18 ````{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.

A screenshot of an RStudio code editor. The code in the editor is:

```
```{r}
5 + 3 + 2
8-4

sum26 <- 2 + 6
sum26```

```

The output pane shows the results of the code execution:

```
[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.

## 1.5 Assignment 1

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

We will start walking through this assignment together during class, but you are welcome to try to do this ahead of time on your own.

You will submit three documents on Canvas:

- An **R script** (.R) file with your code. Follow the best practices by titling your script and using # comments to explain your steps. This code should be clean. I should be able to run your code to verify that the code produces the answers you write down.
- An **.Rmd document and** a compiled RMarkdown **.html document** that you get after “knitting” the .Rmd file. This should also have a title including your name and use text or # comments to explain your steps.

You can create these documents from scratch using the guidance in the previous sections, or you can download and open the .R and .Rmd templates, provided on Canvas, in RStudio to get started. This video provides a brief overview of opening an R script and R Markdown file in RStudio. The notes provide additional details.

[https://www.youtube.com/watch?v=g37\\_-icdPMc](https://www.youtube.com/watch?v=g37_-icdPMc)

### *Assignment Exercises*

1. Create a .R script saved as “LastnameSetup1.R” (use your last name). Within this file, make sure to title it and provide your name.
  1. Set your working directory, and include the file pathway (within `setwd()`) at the top of your .R script
  2. Do the calculation  $8 + 4 - 5$  in your R script. Store it as an object with an informative name. Report the answer as a comment # below the code.
  3. Do the calculation  $6 \times 3$  in your R script. Store it as an object with an informative name. Report the answer as a comment # below the code.

4. Add these two calculations together. Note: do this by adding together the objects you created, not the underlying raw calculations. Report the answer as a # below the code.
2. In this problem, we will just re-format what we did in the first problem in an R Markdown format. Create a .Rmd R Markdown file saved as “LastnameSetup1.Rmd.” Within this file, make sure to title it and provide your name.
  1. Create a Markdown heading # Problem 2.1. Underneath this, create an R code chunk in which you do the calculation  $8 + 4 - 5$ . Store it as an object with an informative name. Report the answer in plain language below the code chunk.
  2. Create a Markdown heading # Problem 2.2. Underneath this, create an R code chunk in which you do the calculation  $6 \times 3$  in your R script. Store it as an object with an informative name. Report the answer in plain language below the code chunk.
  3. Create a Markdown heading # Problem 2.3. Underneath this, create an R code chunk in which you add the previous two calculations together. Note: do this by adding together the objects you created, not the underlying raw calculations. Report the answer in plain language below the code chunk.
  4. Create a Markdown heading # Problem 2.4. Write down how you will complete your R assignments this semester. For example, if you have a personal laptop with R and RStudio on it, you will simply write “I will use my personal laptop.” If you don’t have a personal computer or laptop, please indicate where on campus or off-campus you will have regular access to a computer with R/RStudio to do your work. It is **essential** that you have regular access to a computer so that you will not fall behind in this course.
3. Create a compiled .html file by “knitting” the .Rmd file into a .html document. Save the file as “LastnameSetup1.html.”

All done! Submit the three documents on Canvas.

## 2 Description

What are things we want to describe in political science?

- Unemployment rate, GDP
- Voter turnout, vote share for a party in an election
- Percentage of women in the labor force
- Poverty rates over time

What else? What does description help us achieve?

- Identify tendencies
- Identify patterns or trends
- Identify relationships between two or more factors
- Help us generalize from anecdotes, what is common vs. what is uncommon?
- Diagnose demand, needs, potential problems, likely outcomes

Generate ideas for other goals, such as explanation and prediction

### 2.1 Process of Describing

How do we go about a descriptive quantitative analysis?

1. Substantive Expertise: Start with a topic, puzzle, or question (e.g., How is the economy doing?)
2. Find outcome data relevant to that question (e.g., GDP)
  - Start from a concept: what we want to describe (i.e., health of the economy)
  - Move toward an “operationalization” (i.e., a way to measure it)
  - Easy! except... social science is messy. Our concepts are rich, while our measures may be very narrow or concrete.
    - For example, GDP is one way to measure economic health, but is it the only measure?
    - Choose measures based on validity, reliability, cost
3. Find multiple relevant units or “data points”
  - E.g., Multiple years of data (e.g., U.S., from 1900 to 2020)

- E.g., Multiple countries from one year (e.g., U.S. to Germany to other countries)
4. Summarize the data to help answer the question

### 2.1.1 Example Process

1. How is the economy doing?
2. Find outcome data relevant to that question
  - Let's ask people
3. Find multiple relevant units or data points
  - We will ask several people. Each person will be a data point.
4. Summarize the data
  - Let's take the mean

Let's say we ask 10 people, "Is the economy doing well?" We will give a person a 1 if they say yes, a 0 if they say no. We will index each person by  $i$ , and we have a total of  $N = 10$  people.

i	People	Outcome
1	Joe	0
2	Sally	0
3	Trevor	0
4	Emily	0
5	Mark	1
6	Sarah Jane	1
7	Stacey	0
8	Steve	1
9	Phoebe	0
10	Jesse	1

How would you summarize information in explaining it to another person? You would probably want to describe how most people feel about the economy. In other words, you would describe the “central tendency” of people’s responses (the central tendency of the data).

## 2.2 Summarizing univariate data

For a video explainer of the code in this section, see below. The video only discusses the code. Use the notes and lecture discussion for additional context. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)

<https://www.youtube.com/watch?v=80tbdiWuljc>

Univariate data refers to data coming from one “variable,” where a variable captures the values of a changing characteristic.

Our set of values is Outcome = {0,0,0,0,1,1,0,1,0,1}.

- We will call this a vector of values, where a vector is just a collection of things.
- Because our vector contains only numbers, we will call it a *numeric* vector.
- Each value can be indexed by *i*, denoting the position of the value in the
- For example, Jesse is in position *i*=10 of the vector, and his value is 1

We can create vectors in R by using `c()` and assigning `<-` it to an object we will call `Outcome`.

```
Outcome <- c(0,0,0,0,1,1,0,1,0,1) # Use commas to separate values
```

We can extract a particular value within our vector using brackets and the value's numeric position in the vector.

```
Outcome[10] # what value is in the 10th position?
```

```
[1] 1
```

We can label our outcomes using `names()`

```
names(Outcome) <- c("Joe", "Sally", "Trevor", "Emily", "Mark",
 "Sarah Jane", "Stacey", "Steve", "Phoebe", "Jesse")
Outcome[10]
```

```
Jesse
```

```
1
```

We can overwrite whole vectors or values within a vector

```
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
```

Oops we accidentally type a 0 for Jesse.

```
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
```

Vectors do not have to be numeric. Character vectors contain a collection of words and phrases.  
In R, we use quotations around character values

Example: let's create a vector of names that we will call `People`.

```
People <- c("Joe", "Sally", "Trevor", "Emily", "Mark", "Sarah Jane", "Stacey", "Steve", "Ph
People[10]
```

```
[1] "Jesse"
```

We can use the R function `class()` to tell us the type of object we have.

```
class(Outcome)
```

```
[1] "numeric"
```

```
class(People)
```

```
[1] "character"
```

## 2.3 Functions to summarize univariate data

For univariate data, often we are interested in describing the range of the values and their central tendency.

- range: the minimum (`min()`) and maximum (`max()`) values
- mean: the average value (`mean()`)

The average is the sum of the values divided by the number of values:

$$\bar{X} = \frac{\text{sum of values}}{\text{number of values}} = \frac{x_1 + x_2 + \dots + x_N}{N} = \frac{1}{N} \sum_{i=1}^{i=N} x_i$$

Let's do this in R for our set of 10 values

```
(0 + 0 + 0 + 0 + 1 + 1 + 0 + 1 + 0 + 1)/10
```

```
[1] 0.4
```

The average outcome is .4. Note: when a variable contains only 0's and 1's its mean is the proportion of 1's. 40% of people think the economy is doing well.

### 2.3.1 Using functions in R (overview)

A function is an action(s) that you request R to perform on an object or set of objects. For example, we will use the `mean()` function to ask R to take the mean or “average” of a vector.

- Inside the function you place inputs or “arguments.”

```
mean(Outcome)
```

```
[1] 0.4
```

R also has functions that take the sum `sum()` of a vector of values.

```
sumofvalues <- sum(Outcome)
```

And that count the total number of values or “length” `length()` of the vector.

```
numberofvalues <- length(Outcome)
```

Note that the below is also equivalent to the mean

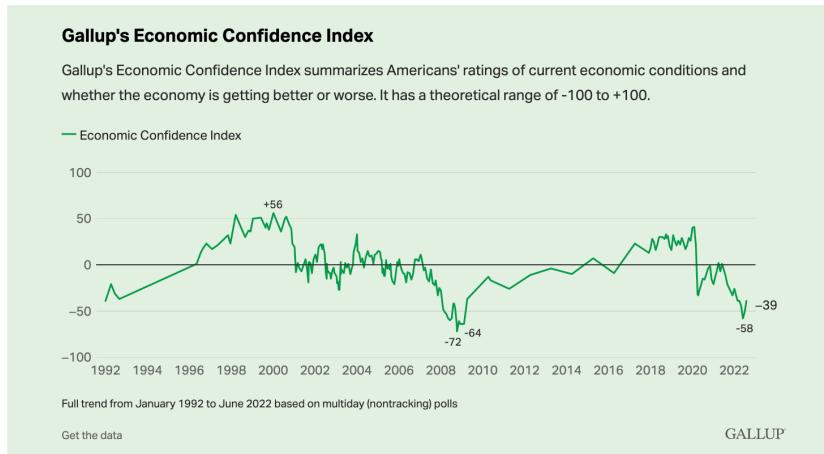
```
sumofvalues / numberofvalues
```

```
[1] 0.4
```

Returning to our example, we found that 40% of people surveyed thought the economy was doing well. Surveying people about their opinions on how the country is doing is a common way that social scientists use description. We could extend this exercise in many ways going forward, even with the same question.

- Start with a question: How is the economy doing?
- Let's find a measure: Ask people if the economy is doing well.
- Find data points: Multiple people (we could stop there with the average!), or add more variables:
  - Across time: Survey people across multiple years
  - Across type of people: Survey different partisan groups

These types of trends are often used by news organizations and public opinion organizations like, Gallup.



This was just a first example of description in political science. There are many other ways to describe how the economy is doing and many other topics we might want to describe in politics.

## 2.4 Loading data into R

For this section, our motivating example will be methods to measure voter turnout in the United States.

Describing voter turnout

- What is a typical level of voter turnout?
- How has turnout changed over time?
- Is turnout higher in presidential years or in midterm years?

How can we measure turnout? Think about the validity, reliability, and cost of different approaches.

Example: Dataset on Voter Turnout in the U.S. across multiple years

	year	VEP	VAP	total	ANES	felons	noncit	overseas
1	1980	159635	164445	86515	71	802	5756	1803
2	1982	160467	166028	67616	60	960	6641	1982
3	1984	167702	173995	92653	74	1165	7482	2361
4	1986	170396	177922	64991	53	1367	8362	2216
5	1988	173579	181955	91595	70	1594	9280	2257
6	1990	176629	186159	67859	47	1901	10239	2659
7	1992	179656	190778	104405	75	2183	11447	2418
8	1994	182623	195258	75106	56	2441	12497	2229
9	1996	186347	200016	96263	73	2586	13601	2499
10	1998	190420	205313	72537	52	2920	14988	2937
11	2000	194331	210623	105375	73	3083	16218	2937
12	2002	198382	215462	78382	62	3168	17237	3308
13	2004	203483	220336	122295	77	3158	18068	3862
14	2008	213314	230872	131304	78	3145	19392	4972

In this dataset, each row is an election year. Each column contains information about the population, potential voters, or voter turnout. These will help us compute the turnout rate in a given year. To work with this dataset, we need to load it into R.

#### 2.4.1 Working with datasets in R

For a video explainer of the code in this section, see below. The video only discusses the code. Use the notes and lecture discussion for additional context. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)

[https://www.youtube.com/watch?v=rm\\_g0rrgIEQ](https://www.youtube.com/watch?v=rm_g0rrgIEQ)

Often the variables we care about are stored inside of rectangular datasets

- These have a number of rows `nrow()` and columns `ncol()`

- Each row is an “observation,” representing the information collected from an individual or entity
- Each column is a variable, representing a changing characteristic across multiple observations

When we import a dataset into R, we have a few options.

Option 1: Download dataset to your computer

- Move the dataset to your working directory
- Identify the file type (e.g., csv, dta, RData, txt)
- Pick the appropriate R function to match the type (e.g., `read.csv()`, `read.dta()`, `load()`, `read.table()`)
- Assign the dataset to an object. This object will now be `class()` of `data.frame`

```
turnout <- read.csv("turnout.csv")
```

Click here for an alternative function for csv files.

Some scholars prefer to use the function `read_csv` to load csv data. It is better at handling more complicated types of data. We will not need to use this function in this course, but you may encounter it elsewhere.

To use this function, the first time we will go about using it, we have to first install a “package” called `readr`. Packages in R give us additional tools beyond what the base version of R provides. It is like installing an extra app on your phone.

```
install.packages("readr")
```

Once we have that installed, now anytime we want to use the function, we will call (open) the “`readr`” package using `library()`, and then the syntax is just like using the `read.csv` function.

```
library(readr)
turnout <- read_csv("turnout.csv")
```

Option 2: Read file from a url provided

- Need an active internet connection for this to work
- URL generally must be public
- Include the url inside the function used to read the data

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

```
class(turnout)
```

```
[1] "data.frame"
```

You can also open up a window to view the data:

```
View(turnout)
```

## 2.4.2 Measuring the Turnout in the US Elections

Relevant questions with voter turnout

- What is a typical level of voter turnout?
- Is turnout higher in presidential years or in midterm years?
- Is turnout higher or lower based on voting-eligible (VEP) or voting-age (VAP) populations? We have a lot of people who are citizens 18 and older who are ineligible to vote. This makes the VEP denominator smaller than the VAP.

Voter Turnout in the U.S.

- Numerator: **total**: Total votes cast (in thousands)
- Denominator:
  - VAP: (voting-age population) from Census
  - VEP (voting-eligible population)  $VEP = VAP + \text{overseas voters} - \text{ineligible voters}$
- Additional Variables and Descriptions
  - **year**: election year
  - **ANES**: ANES self-reported estimated turnout rate
  - **VEP**: Voting Eligible Population (in thousands)
  - **VAP**: Voting Age Population (in thousands)
  - **total**: total ballots cast for highest office (in thousands)
  - **felons**: total ineligible felons (in thousands)
  - **noncitizens**: total non-citizens (in thousands)
  - **overseas**: total eligible overseas voters (in thousands)
  - **osvoters**: total ballots counted by overseas voters (in thousands)

### 2.4.3 Getting to know your data

```
How many observations (the rows)?
nrow(turnout)

[1] 14

How many variables (the columns)?
ncol(turnout)

[1] 9

What are the variable names?
names(turnout)

[1] "year" "VEP" "VAP" "total" "ANES" "felons" "noncit"
[8] "overseas" "osvoters"

Show the first six rows
head(turnout)

 year VEP VAP total ANES felons noncit overseas osvoters
1 1980 159635 164445 86515 71 802 5756 1803 NA
2 1982 160467 166028 67616 60 960 6641 1982 NA
3 1984 167702 173995 92653 74 1165 7482 2361 NA
4 1986 170396 177922 64991 53 1367 8362 2216 NA
5 1988 173579 181955 91595 70 1594 9280 2257 NA
6 1990 176629 186159 67859 47 1901 10239 2659 NA
```

Extract a particular column (vector) from the data using the \$.

```
turnout$year
```

```
[1] 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2008
```

Extract the 10th year. Just like before! We use 10 to indicate the value of the `year` column in position (row 10) of the data.

```
turnout$year[10]
```

```
[1] 1998
```

We can take the `mean()` of a particular column, too. Let's take it of the total number of voters.

```
mean(turnout$total)
```

```
[1] 89778.29
```

And get the `class()` (Note: integer is just a type of numeric variable)

```
class(turnout$total)
```

```
[1] "integer"
```

We can also use brackets in the full data frame, but because our data frame has BOTH rows and columns, we cannot just supply one position `i`. Instead, we have to tell R which row AND which column by using a comma between the positions.

```
turnout[1,2] # value in row 1, column 2
```

```
[1] 159635
```

We can use the column name instead

```
turnout[1, "VEP"]
```

```
[1] 159635
```

If we leave the second entry blank, it will return all columns for the specified row

```
turnout[1,] # All variable values for row 1
```

	year	VEP	VAP	total	ANES	felons	noncit	overseas	osvoters
1	1980	159635	164445	86515	71	802	5756	1803	NA

The opposite is true if we leave the first entry blank.

```
turnout[,2] # VEP for all rows
```

```
[1] 159635 160467 167702 170396 173579 176629 179656 182623 186347 190420
[11] 194331 198382 203483 213314
```

## 2.5 Comparing VEP and VAP turnout

### 2.5.1 Creating new variables in R

Let's create a new variable that is VAP that adds overseas voters.

```
Use $ to add a new variable (i.e., column) to a data frame
turnout$VAPplusoverseas <- turnout$VAP + turnout$overseas
```

Under the hood, what this is doing is taking each value of `turnout$VAP` and adding it to its corresponding values of `turnout$overseas`.

And, yes, this new variable shows up as a new column in `turnout`. Go ahead, `View()` it

```
View(turnout)
```

This does not change the underlying `turnout.csv` file, only the `turnout data.frame` we are working with in the current R session.

- This is an advantage of using an R script.
- You don't have to worry about overwriting/messing up the raw data.
- You start from the original raw data when you load `turnout.csv`, and then everything else is done within R.

This is our new denominator. Now we can calculate turnout based on this denominator.

```
turnout$newVAPturnout <- turnout$total / turnout$VAPplusoverseas
```

Just like with adding two vectors, when we divide, each value in the first vector is divided by its corresponding value in the second vector.

```
turnout$newVAPturnout
```

```
[1] 0.5203972 0.4024522 0.5253748 0.3607845 0.4972260 0.3593884 0.5404097
[8] 0.3803086 0.4753376 0.3483169 0.4934211 0.3582850 0.5454777 0.5567409
```

Let's calculate the VEP turnout rate and turn it into a percentage. This time, we do it in one step.

- $(\text{total votes} / \text{VEP}) \times 100$ :

```
turnout$newVEPturnout <- (turnout$total / turnout$VEP) * 100
turnout$newVEPturnout
```

```
[1] 54.19551 42.13701 55.24860 38.14115 52.76848 38.41895 58.11384 41.12625
[9] 51.65793 38.09316 54.22449 39.51064 60.10084 61.55433
```

Let's change it from a proportion to a percentage. How? Multiply each value of `turnout$newVAP` by 100

```
turnout$newVAPturnout <- turnout$newVAPturnout * 100
```

This multiplies each number within the vector by 100.

```
turnout$newVAPturnout
```

```
[1] 52.03972 40.24522 52.53748 36.07845 49.72260 35.93884 54.04097 38.03086
[9] 47.53376 34.83169 49.34211 35.82850 54.54777 55.67409
```

What is typical turnout?

```
mean(turnout$newVAPturnout)
```

```
[1] 45.45658
```

```
mean(turnout$newVEPturnout)
```

```
[1] 48.94937
```

We find that turnout based on the voting age population is lower than turnout based on the voting eligible population. This is a pattern that political scientists have examined, going back several decades. For example, in a 2001 article McDonald and Popkin show that it is the ineligible population that grew from the 1970s onward and not the population of people who simply prefer not to vote. (See more [here](#).)

**FIGURE 1. National VAP and VEP Presidential Turnout Rates, 1948–2000**

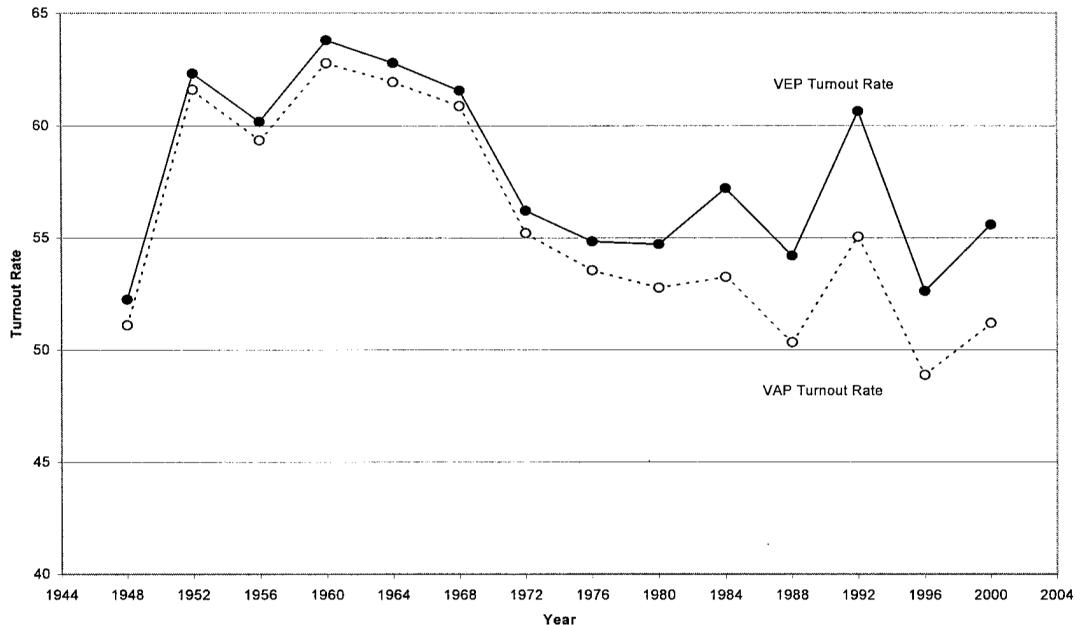


Figure 2.1: McDonald and Popkin 2001

## 2.6 Comparing Presidential vs. Midterm turnout

How does turnout compare in presidential vs. midterm years? Sometimes using a single summary of turnout may obscure important underlying differences in the data. To detect these differences, we may want to summarize different parts of the data.

Oh dear. We need to extract specific years from the turnout data frame. Which rows contain the years we want?

```
turnout$year
```

```
[1] 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2008
```

Ok: rows 1,3,5,7,9,11,13,14 are the presidential. And rows 2,4,6,8,10,12 are midterms.

```
we can extract all of these at once by using c()
turnout$year[c(1,3,5,7,9,11,13,14)] # presidential
```

```
[1] 1980 1984 1988 1992 1996 2000 2004 2008
```

Let's take the mean VEP turnout for presidential years.

```
mean(turnout$newVEPturnout[c(1,3,5,7,9,11,13,14)])
```

```
[1] 55.983
```

Let's take the mean VEP turnout for midterm years.

```
mean(turnout$newVEPturnout[c(2,4,6,8,10,12)])
```

```
[1] 39.5712
```

Let's take the difference by storing each mean and then subtracting

```
mean.VEP.pres <- mean(turnout$newVEPturnout[c(1,3,5,7,9,11,13,14)])
mean.VEP.mid <- mean(turnout$newVEPturnout[c(2,4,6,8,10,12)])
mean.VEP.pres - mean.VEP.mid
```

```
[1] 16.41181
```

Presidential turnout, on average, is higher than midterm turnout.

### 2.6.1 R shortcut for writing vectors

Sometimes we write numbers that are in a predictable sequence (e.g., 1,2,3,4,5). In R, we have functions that prevent us from having to type each number when this is the case.

```
c(1,2,3,4,5) # is equivalent to:
```

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

```
1:5 # is equivalent to:
```

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

```
seq(from = 1, to = 5, by = 1)
```

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

We can use the last one to our advantage to extract the midterm years, which go by 2

```
mean(turnout$newVEPturnout[c(2,4,6,8,10,12)]) # is the same as
```

```
[1] 39.5712
```

```
mean(turnout$newVEPturnout[seq(2, 12, 2)])
```

```
[1] 39.5712
```

Not a big deal now, but imagine if you had to write 100 numbers or 1 MILLION NUMBERS!

## 2.7 Creating dataframes from within R

While importing data from outside of R is the most common way to work with dataframes in R, you can also create dataframes from inside R. Ultimately, a dataframe just binds together multiple vectors / columns to create a rectangular object.

For example, let's say we want to create a dataframe with columns indicating just the midterm years and their VEP turnout. These correspond to the two vectors:

- `turnout$newVEPturnout[seq(2, 12, 2)]`
- `turnout$year[seq(2, 12, 2)]`

In R, you can create a rectangular `data.frame` object with the `data.frame` function.

- Within this function, you can make several entries that follow the syntax `colname = values`. We supply what we would like the name of the column to be, such as `midyear`, and then provide R with a set of values. We can then provide a comma and add more columns.
  - You just want to make sure each column has the same number of values.

```
midtermdata <- data.frame(midyear = turnout$year[seq(2, 12, 2)],
 VEPturnout = turnout$newVEPturnout[seq(2, 12, 2)])
```

You can supply the values for each column using objects or just vectors of raw numeric values like the below:

```
midtermdata <- data.frame(midyear = c(1982, 1986, 1990, 1994, 1998, 2002),
 VEPturnout = c(42.13701, 38.14115, 38.41895, 41.12625, 38.09316,
```

The result is a nice rectangular dataframe similar to what we loaded using the `turnout.csv` dataset from outside of R.

```
midtermdata

 midyear VEPturnout
1 1982 42.13701
2 1986 38.14115
3 1990 38.41895
4 1994 41.12625
5 1998 38.09316
6 2002 39.51064
```

Now, because our dataframe has a different name. If we want to access columns from this dataframe, we start with `midterm$` followed by the variable name.

```
midtermdata$midyear

[1] 1982 1986 1990 1994 1998 2002
```

## 2.8 Wrapping Up Description

In this section, we have described voter turnout using multiple measures and types of elections. There are several other questions that political scientists may be interested in when it comes to voter turnout.

For example, during and following the 2020 elections, many states passed laws that changed election procedures: Ability to vote by mail, Ballot dropboxes, Length of early voting. What else?

- What effect (if any) do these laws have on voter turnout?

In the next section, we start to examine how to evaluate causal claims.

### 2.8.1 Summary of R tools

We have touched on a number of R tools thus far. Here is a summary of some of the key items to remember going forward:

- `setwd()`: sets the working directory in R, which tells R which folder on your computer contains the datasets or other R files where you will be working. You should get into the habit of setting your working directory each time you work in RStudio.
  - Can set this in the toolbar Session -> Set Working Directory -> Choose Directory, followed by clicking the “Open” button on the folder where you want to work.
  - Example: `setwd("~/Downloads/Data Science")`
- `##`: Hashtags are used to help annotate your code. Anything behind a hashtag is treated as plain text
- `+ - * /`: These are some of the mathematical operators you can use in R
  - You can also control which operations are performed first using () just like you would do with math outside of R. For example, try to compare the answer to `6 + 4 * 3` with `(6 + 4) * 3`
- `<-`: This is an assignment tool that allows us to store calculations, vectors, datasets, and more as *objects* in R.
  - Example: `sum53 <- 5 + 3` creates an object called `sum53` that stores the calculation on the right.
- `[]`: Brackets are used to extract specific components of objects we create. The number(s) inside the brackets tell us which entries to extract.
  - Example: `Outcome[2]` will tell us to extract the second entry in the object `Outcome`
  - Note: when we use datasets, the brackets will have two entries, one corresponding to the row entry and one corresponding to the column. Example `turnout[1,2]` means the entry in the first row and second column.

*Functions* We have already started using a number of *functions* in R, which are operations we ask R to do for us, such as creating vectors, importing data, or summarizing data by finding the mean, range, etc. Functions come in the same format, which starts with the function name followed by parentheses. Example: `mean()`. Each function then takes a particular input(s). When you “run” a line of code with a function, R applies the function to the input.

- `c()`: This is a function that combines a set of values into a vector in R. The values can be numbers or text items and should be separated by commas. If text, each text item should be in quotation marks.
  - Example: `Outcome <- c(3, 4, 6, 2, 1)`
  - Example: `People <- c("Sam", "Julie", "Mark")`

- `mean()`, `median()`, `min()`, `max()`, `range()`: These functions summarize vectors that are numeric/integers in nature.
  - Example `mean(Outcome)` takes the average of the values in the `Outcome` vector
- `read.csv()`: This function loads a rectangular .csv file into R as a `data.frame`
  - Example: `turnout <- read.csv("turnout.csv")`
  - Not all datasets will be .csv files. In the future, we will use other functions, such as `load()` or `read.dta()` to import datasets of different file types.

### *Dataframes*

We have started working with dataframes in R. These objects are rectangular datasets that include a collection of vectors. Every column in a dataframe generally represents a different concept or “variable,” while each row represents a different unit or “observation.”

- `$`: When we are working with vectors that are inside of a dataframe (the columns inside of a dataframe), we use the `$` to access them.
  - Example: `turnout$year` will show us the values in the `year` column vector inside our `turnout` rectangular dataframe
- `nrow()`, `ncol()`, `dim()`, `head()`, `names()`: These functions help us explore the dataframes by telling us the number of rows and columns (the dimensions), giving us a sneak peek of the first 6 rows of the dataframe, or showing us the names of the variables (columns) in the data.
  - Example: `nrow(turnout)`

# 3 Causation with Experiments

Recall that we said, four primary goals of social science include:

- **Describe** and measure
  - Has the U.S. population increased?
- **Explain**, evaluate, and recommend (study of causation)
  - Does expanding Medicaid improve health outcomes?
- **Predict**
  - Who will win the next election?
- **Discover**
  - How do policies diffuse across states?

In this section, we start to explore the goal of explanation-making causal claims.

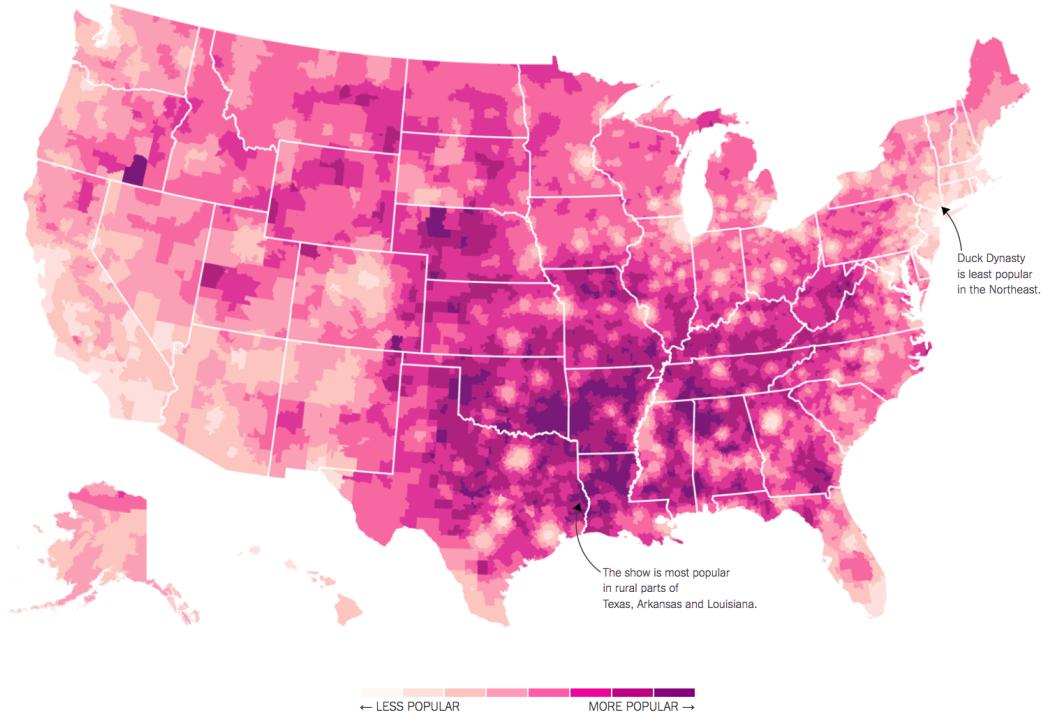
## 3.1 What separates causation from correlation?

Here's an example. In 2016, researchers at the [NY Times](#) noticed that areas in the country where the television show *Duck Dynasty* was popular also tended to support Donald Trump at higher rates.

---

## 1. Duck Dynasty

---



If we put our social scientist hat on, we might want to distinguish whether this is a causal or, more likely, just a correlational relationship:

- Correlation: Areas that watch Duck Dynasty are more likely to support Trump (degree to which two variables “move together”)
- Causality: Watching Duck Dynasty (vs. not watching) increases your support of Trump.

Causal Question: Does the manipulation of one factor (the treatment), (holding everything else constant), cause a change in an outcome?

### 3.1.1 Potential Outcomes Framework

When studying causal relationships, we distinguish two concepts:

- treatment: variable whose change may produce a change in the outcome (e.g., watching vs. not watching *Duck Dynasty*)
- outcome ( $Y$ ): what may change as a result (e.g., their support for Trump)

We imagine two states of the world or “potential outcomes.”

- $Y(1)$ : the outcome if the treatment is administered (e.g., watching the show)
- $Y(0)$ : the outcome if the treatment is NOT administered or maybe something else is (e.g., not watching the show)

Political Science Example: How does voter turnout ( $Y$ ) change as a result of varying whether someone receives a mail-in ballot (the treatment)?

- $Y(\text{sent a mail-in ballot})$ : do you vote or not
- $Y(\text{not sent a mail-in ballot})$ : do you vote or not

We compare your likelihood of turning out to vote in a world where you did receive a mail-in ballot vs. a counterfactual state of the world in which you did not receive a mail-in ballot, generally assuming that this is the only thing that is different between these two potential states of the world.

In many cases in social science, we might start by observing some connection in the real world. To make a causal claim, we then have to imagine what that counterfactual state of the world would be. Examples:

Causal Question: Does the minimum wage increase the unemployment rate?

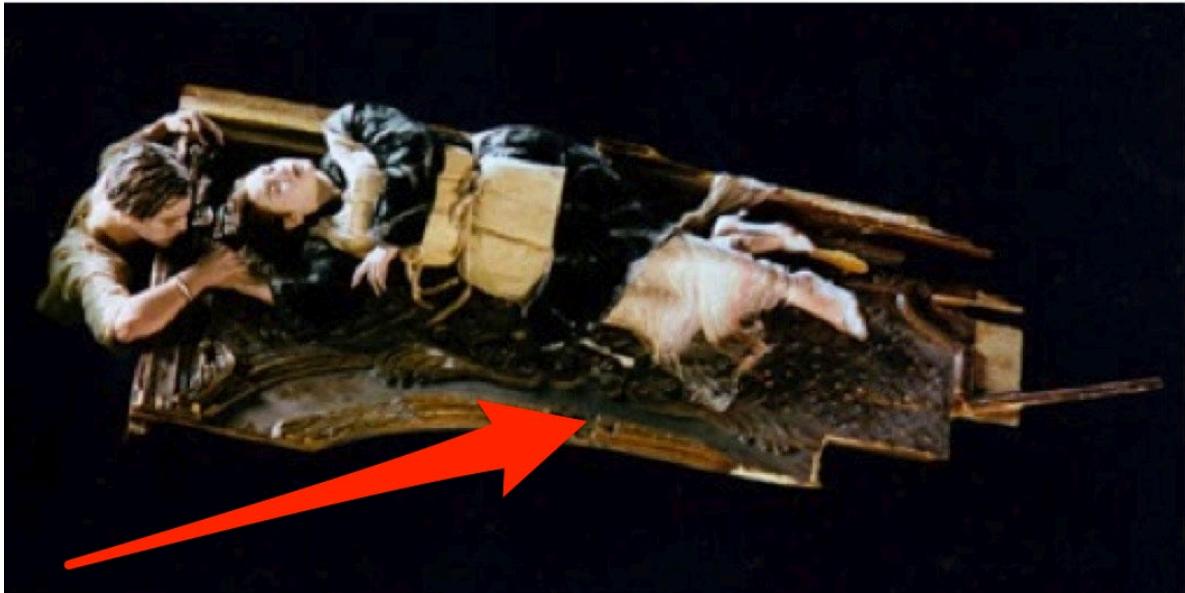
- (Hypothetical) Factual: An unemployment rate went up after the minimum wage increased
- Implied Counterfactual: Would the unemployment rate have gone up, had the minimum wage increase not occurred?

Causal Question: Does the gender of a political messenger influence the persuasiveness of the message?

- (Hypothetical) Factual: Suppose a political messenger perceived as a man had a somewhat persuasive effect delivering a message on abortion.
- Implied Counterfactual: Would a political messenger perceived as a woman have a similar or different persuasive effect?

We use causal logic all of the time outside of social science.

For example, many viewers get angry after watching the movie *Titanic* because they believe Jack did not have to die. We can place their claims in our causal framework:



- Outcome: Jack Surviving the Titanic
- Potential Outcomes in two states of the world
  - Rose did not share the floating door, and Jack died.
  - Counterfactual question: If Rose had shared the floating door, would Jack have lived?

In *Bit by Bit*, Matt Salganik notes that sometimes cause-and-effect questions are implicit. For example, in more general questions about maximization of some performance metric, we might want to compare several alternatives:

The question “What color should the donate button be on an NGO’s website?” is really lots of questions about the effect of different button colors on donations.

- Factual: A voter donates some amount with a black button
- Counterfactual: What would a voter donate if the button were blue?
- Counterfactual: What would a voter donate if the button were red?

What other causal questions might social scientists or data scientists ask?

### 3.1.2 Causal Effects

When we are conducting a causal analysis, we will want to estimate a causal effect.

- Causal effects are all about ideal comparisons between treated vs. untreated

A causal effect is the change in the outcome Y that is caused by a change in the treatment variable.

- $Y(1) - Y(0)$  = causal effect or “treatment effect”
- e.g., Donation if contacted - Donation if not contacted

We often want to know the **average treatment effect** in some population, not just the causal effect for a single individual. Here, we might ask, on average, how much would our outcome change if our units were treated instead of untreated. To do so, we simply sum up all of the causal effects and divide them by the number of units in our population.

- $\frac{1}{N} \sum_{i=1}^N (Y_i(1) - Y_i(0))$  = “average treatment effect” (ATE)
  - e.g., Average donations if contacted - Average donations if not contacted

Note: If the math above is helpful, you can use it. If it is difficult to read, focus on the plain language definitions that go before it. The notation here is less important than the conceptual understanding.

### 3.1.3 Fundamental Problem of Causal Inference

The problem: Fundamental Problem of Causal Inference

What makes the evaluation of causal claims difficult, is that in the real world, we suffer from the fundamental problem of causal inference:

- For any individual, we only get to see (observe) the result from one state of the world
  - This makes that subtraction of potential outcomes impossible.

(Unless we are in [Groundhog Day](#)

## 3.2 Randomized Controlled Trials

One approach for addressing the fundamental problem of causal inference is to simulate two potential states of the world through random assignment: Randomized Controlled Trials / Experiments

Experiments approximate an ideal factual vs. counterfactual comparison

- We randomly assign one group to receive a “treatment” and another not to receive a treatment (the control)
  - There can be more than two groups. The key is that each group varies (is manipulated) in some way.

- When treatment assignment is **randomized**, the only thing that distinguishes the treatment group from the control group, besides the treatment itself, is chance.

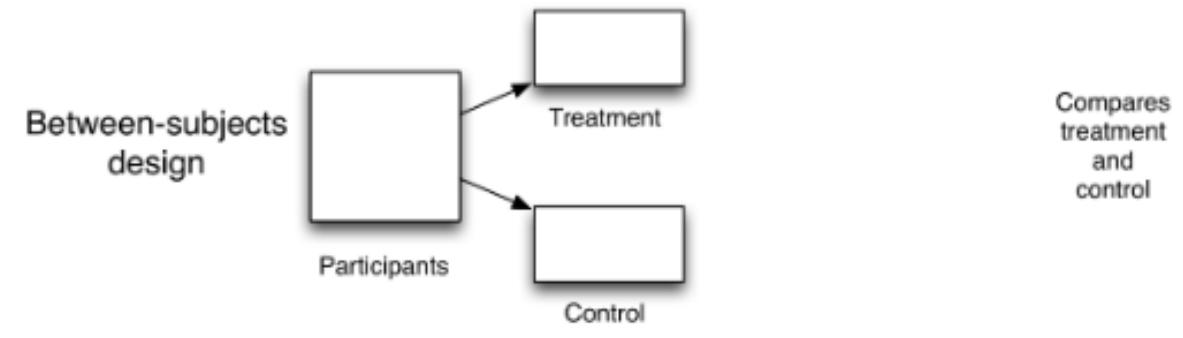


Figure 3.1: Salganik Bit by Bit Chapter 4.4

This allows us to compare the average outcomes between groups in order to estimate our causal effects (more on this below).

### 3.2.1 Experiments: Why Randomize?

Randomization is essential for being able to *identify* and *isolate* the causal effect of the treatment on the outcome.

Without randomization, there may be several reasons why two groups differ beyond the treatment of interest.

- For example, if we randomly assigned half of Rutgers seniors to watch the movie *Oppenheimer* and half to watch *Barbie* we would expect the groups to have about equal proportions of female students, average age, racial composition, majors, etc.
  - (If we didn't randomly assign, and just let people “select” into watching a particular movie, the groups could look very different.)

But because we randomized assignment, on average, we'd expect the two groups to be identical except for the treatment— in this case, which movie they watched.

- Great news! This means any differences in the outcomes between the two groups can be attributed to the treatment. So if we wanted to see if *Top Gun Maverick* leads people to take up flying lessons, we could compare the average number of flying lessons among seniors who watched *Top Gun Maverick* compared to those who didn't, and instead watched a different movie.

### 3.2.2 Experiments: How to Analyze

Difference in Means: We compare each group's average outcome by subtracting one from the other to estimate the average treatment effect (ATE) aka the average causal effect of the treatment.

- $\widehat{ATE} = \bar{Y}(treatment) - \bar{Y}(control)$

This is an estimate of, on average, how much our outcome would change if units went from being untreated to treated.

- E.g., on average how much a person donates to a campaign if contacted by phone compared to if not contacted by phone.

### 3.2.3 Ingredients of an Experiment

From *Bit by Bit*

## 4.2 What are experiments?

**Randomized controlled experiments have four main ingredients: recruitment of participants, randomization of treatment, delivery of treatment, and measurement of outcomes.**

For every experiment, you should be able to

- State the causal question or relationship of interest
- Describe how the experiment will be implemented (e.g., recruitment of subjects)
- Identify and describe the randomization into treatment group(s) and control group and what happens in each group
- Identify the outcome of interest, how it is measured
- Evaluate the relevant comparison (between two different experimental conditions)

We will turn to an example in the next section.

### 3.3 Application: Is there racial discrimination in the labor market?

Marianne Bertrand and Sendhil Mullainathan. 2004. “Are Emily and Greg more employable than Lakisha and Jamal? A field experiment on labor market discrimination.”

“We perform a field experiment to measure racial discrimination in the labor market. We respond with fictitious resumes to help-wanted ads in Boston and Chicago newspapers.”

- Recruitment: Construct resumes to send to ads
- Randomization: To manipulate perception of race, each resume is (randomly) assigned
- Treatment: either a very African American sounding name
- Control: or a very White sounding name
- Outcome: Does the resume receive a callback?
- Comparison: Callback rates for African American (sounding) names vs. White (sound-ing) names (the difference in means between groups)

*For a video explainer of the code in this section, see below. The video only discusses the code. Use the notes and lecture discussion for additional context. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)*

<https://www.youtube.com/watch?v=LeJkRydMruM>

Let’s load the data. Note: When we have variables that are text-based categories, we may want to tell R to treat these “strings” of text information as factor variables, a particular type of variable that represents data as a set of nominal (unordered) or ordinal (ordered) categories. We do this with the `stringsAsFactors` argument.

```
resume <- read.csv("resume.csv", stringsAsFactors = T)

resume <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/resume.csv",
 stringsAsFactors = T)
```

Variables and Description

- `firstname`: first name of the fictitious job applicant
- `sex`: sex of applicant (female or male)
- `race`: race of applicant (black or white)
- `call`: whether a callback was made (1 = yes, 0 = no)

The data contain 4870 resumes and 4 variables.

```
nrow(resume) # number of rows
```

```
[1] 4870
```

```
 ncol(resume) # number of columns

[1] 4

 dim(resume) # number of rows and columns

[1] 4870 4
```

Note: These data look a little different from what we used last week. For example, the `sex` and `race` variables contain words, not numbers.

```
 head(resume)

 firstname sex race call
1 Allison female white 0
2 Kristen female white 0
3 Lakisha female black 0
4 Latonya female black 0
5 Carrie female white 0
6 Jay male white 0
```

### 3.3.1 Variable classes

We can check the class of each variable: Look, we have a new type, a “factor” variable.

```
 class(resume$firstname)

[1] "factor"

 class(resume$sex)

[1] "factor"

 class(resume$race)

[1] "factor"
```

```
class(resume$call)
```

```
[1] "integer"
```

We have now encountered **numeric**, **character**, and **factor** vectors and/or variables in R. Note: This is simply how R understands them. Sometimes R can get it wrong. For example, if we write:

```
somenumbers <- c("1", "3", "4")
class(somenumbers)
```

```
[1] "character"
```

Because we put our numbers in quotation marks, R thinks the values in **somenumbers** are text. The number “3” might as well be the word “blue” for all R knows. Fortunately, we can easily switch between classes.

```
somenumbers <- as.numeric(somenumbers)
class(somenumbers)
```

```
[1] "numeric"
```

Here, we used **as.numeric()** to overwrite and change the character vector into a numeric vector.

#### Rules of Thumb

- Usually, we want **character** variables to store text (e.g., open-ended survey responses)
- We want **numeric** variables to store numbers.
- Usually, we want **factor** variables to store categories.
  - Within R, factor variables assign a number to each category, which is given a label or **level** in the form of text.
  - Categories might be ordinal or “ordered” (e.g., Very likely, Somewhat likely, Not likely) or
  - Unordered (e.g., “male”, “female”)
  - R won’t know if a factor variable is ordered or unordered. Alas, we have to be smarter than R.
  - R might think you have a character variable when you want it to be a factor or the reverse.
    - \* That’s when **as.factor()** and **as.character()** are useful.
- Always check **class()** to find out the variable type

## 3.4 Making tables

A nice thing about numeric and factor variables is we can use the `table` command to see how many observations in our data fall into each category or numerical value.

```
Example: how many black vs. white sounding resumes
table(resume$race)
```

```
black white
2435 2435
```

As mentioned, `factor` variables have levels:

```
levels(resume$race)
```

```
[1] "black" "white"
```

### 3.4.1 Crosstabulation

We can also use the `table` command to show a crosstabulation: a table that displays the frequency of observations across two variables.

```
Example: how many black vs. white sounding resumes by call backs
We can label the two dimensions of the table with the =
table(calledback = resume$call, race = resume$race)
```

```
 race
calledback black white
 0 2278 2200
 1 157 235
```

## 3.5 Conditional Means

Recall how to take a mean of a variable in our data. For example, let's take the mean of the variable `call`.

```
mean(resume$call)
```

```
[1] 0.08049281
```

This gives us the average callbacks (or callback rate) for everyone in our data. In experiments, we want to take the mean for a specific group within our data—the treatment group, and then the mean for the control group.

Somehow, we have to identify, within our data, which rows were part of the treatment group and which were a part of the control group. In this study, we want to identify resumes with an assigned name perceived to be black vs. perceived to be white. This is in our `race` variable.

We will cover a couple of tools to do this, with the first being `tapply`.

To find how the average of one variable (e.g., our outcome—the callback rate) varies across different categories of our factor variable, we use `tapply()`.

```
take the mean of input1 by categories of input2
mean of the call variable conducted separately by race
tapply(resume$call, INDEX=resume$race, mean)
```

```
black white
0.06447639 0.09650924
```

This tells us the callback rate for each group of people in our data. That's not the only way to do this, however. We can also use the tools below.

## 3.6 Relational Operators in R

Goal: Compare callback rates for white sounding names to black sounding names, so we need to be able to filter by race.

Good news: We have several relational operators in R that evaluate logical statements:

- `==`, `<`, `>`, `<=`, `>=`, `!=`
- We have a statement and R evaluates it as `TRUE` or `FALSE`

```
for each observation, does the value of race equal "black"?
resume$race == "black"
```

By putting this logical statement within `[ ]`, we are asking R to take the `mean()` of the variable `resume$call` for the subset of observations for which this logical statement is `TRUE`.

```
mean(resume$call[resume$race == "black"])
```

```
[1] 0.06447639
```

Ultimately, each of these paths has led us to a place where we can estimate the average treatment effect by calculating the difference in means: the difference in callback rates for black and white applicants.

We said the ATE =  $\bar{Y}(\text{treatment}) - \bar{Y}(\text{control})$

```
ate <- mean(resume$call[resume$race == "black"]) -
 mean(resume$call[resume$race == "white"])
ate
```

```
[1] -0.03203285
```

How can we interpret this? Do white applicants have an advantage?

## 3.7 Subsetting data in R

Subsetting Dataframes in R

Maybe we are interested in differences in callbacks for females. One approach for looking at the treatment effect for female applicants, only, is to subset our data to include only female names.

- To do this, we will assign a new `data.frame` object that keeps only those rows where `sex == "female"` and retains all columns
- Below are two approaches for this subsetting, one that uses brackets and one that uses the `subset` function

```
option one
females <- resume[resume$sex == "female",]
option two using subset() - preferred
females <- subset(resume, sex == "female")
```

Now that we have subset the data, this simplifies estimating the ATE for female applicants only.

We said the ATE =  $\bar{Y}(\text{treatment}) - \bar{Y}(\text{control})$

```
ate.females <- mean(females$call[females$race == "black"]) -
 mean(females$call[females$race == "white"])
ate.females
```

```
[1] -0.03264689
```

### 3.7.1 Getting Boooooooooolean

We can make this slightly more complex by adding more criteria. Let's say we wanted to know the callback rates for just female black (sounding) names.

- R allows use to use & (and) and | (or)

```
femaleblack <- subset(resume, sex == "female" & race == "black")
```

We could now find the callback rate for Black females using the tools from above:

```
mean(femaleblack$call)
```

```
[1] 0.06627784
```

## 3.8 Creating New Variables using Conditional statements

We can instead create a new variable in our main dataframe. Let's make a variable that takes the value 1 if a name is female and black sounding and 0, otherwise

```
Initialize a new variable called femaleblackname
resume$femaleblackname <- NA
Assign a 1 to our new variable where sex is female and race is black
resume$femaleblackname[resume$sex == "female" & resume$race == "black"] <- 1
Assign a 0 if sex is not female OR if race is not black
resume$femaleblackname[resume$sex != "female" | resume$race != "black"] <- 0
```

We can check our work

```
table(name = resume$firstname, femaleblack = resume$femaleblackname)
```

	femaleblack
name	0 1
Aisha	0 180
Allison	232 0
Anne	242 0
Brad	63 0
Brendan	65 0

Brett	59	0
Carrie	168	0
Darnell	42	0
Ebony	0	208
Emily	227	0
Geoffrey	59	0
Greg	51	0
Hakim	55	0
Jamal	61	0
Jay	67	0
Jermaine	52	0
Jill	203	0
Kareem	64	0
Keisha	0	183
Kenya	0	196
Kristen	213	0
Lakisha	0	200
Latonya	0	230
Latoya	0	226
Laurie	195	0
Leroy	64	0
Matthew	67	0
Meredith	187	0
Neil	76	0
Rasheed	67	0
Sarah	193	0
Tamika	0	256
Tanisha	0	207
Todd	68	0
Tremayne	69	0
Tyrone	75	0

Let's say we wanted to know the callback rates for just female black (sounding) names.

```
mean(femaleblack$call)
[1] 0.06627784

mean(resume$call[resume$femaleblackname == 1])
[1] 0.06627784
```

BINGO: two ways to do the same thing.

### 3.8.1 `ifelse` statements

Remember how we created the variable `femaleblack`, well there is another way to do that in R using what are called conditional statements with `ifelse()`.

- Can be read: If this relational statement is TRUE, I assign you A, otherwise I assign you B

```
resume$femaleblackname <- ifelse(resume$sex == "female" &
 resume$race == "black", 1, 0)
```

Can be read: If sex is female and race is black, give the observation in the new variable a 1, otherwise give it a 0.

Like most things, we can also get more complicated here. Let's say we wanted to create a variable that indicated both race and sex.

- Can be read: If this relational statement is TRUE, I assign you A,
- Otherwise if this second relational statement is TRUE, I assign you B,
- Otherwise if this third relational statement is TRUE, I assign you C,
- Otherwise I assign you D

```
resume$racesex <- ifelse(resume$sex == "female" &
 resume$race == "black", "FemaleBlack",
 ifelse(resume$sex == "female" &
 resume$race == "white", "FemaleWhite",
 ifelse(resume$sex == "male" &
 resume$race == "white", "MaleWhite", "MaleBlack"))))
```

Note: what you assign can be numeric or text.

## 3.9 Types of Experiments

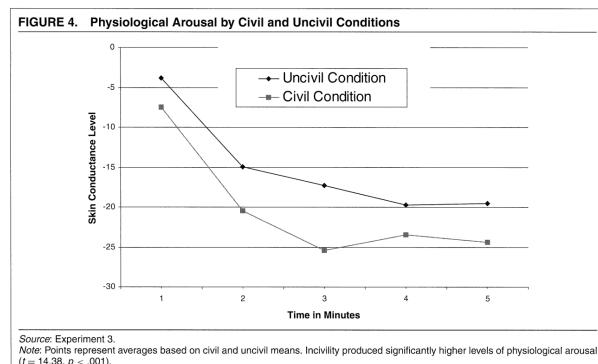
Experiments can vary:

- Setting: Lab, Survey, Field
- Mode: Analog vs. Digital
- And in Validity
  - Internal: were the processes conducted in a correct, reliable way?

- External: can we generalize from the experiment to the real world, or would the results change?
- Context: Would people act the same way outside of the experiment?
- Recruitment: Are the people in our experiment representative of the people we care about?
- Construct
  - \* Treatment: Is the experimental treatment similar to what people see in the real world?
  - \* Outcome: Is the outcome something we care about in the real world? Are we measuring it in a realistic, accurate way?

Review *Bit by Bit* chapter 4 for more examples of social science experiments.

### **Example: Televised Incivility, Trust and Emotions (Mutz and Reeves)**



Participants sat alone in a room with electrodes attached to their hands to measure skin conductance. Subjects viewed 20 minutes of a political debate created for the experiment, which varied in civility and politeness. Results showed respondents had more of an emotional response to the uncivil condition and expressed less trust in politicians.

### **Example: Online Survey Experiment**

#### *Audience Costs (Tomz)*

A country sent its military to take over a neighboring country. The attacking country was led by a [dictator, who invaded **OR** democratically elected government, which invaded] [to get more power and resources **OR** because of a longstanding historical feud].

The attacking country had a [strong military, so it would **OR** weak military, so it would not] have taken a major effort for the United States to help push them out.

A victory by the attacking country would [hurt **OR** not affect] the safety and economy of the United States.

- Participants provided a different version of the vignette above, and a reaction by the president

- Presidential approval varies depending on the president's response and the nature of the situation

### ***Example: Digital Field Experiments in Campaigns***

Example: A/B Testing in Campaigns

			Kamala 2020	Inbox	Kamala is asking - debate where Kamala will be on stage to share our vision for our America. Then, later this week, Kamala is in South Carolina
			Kamala Harris	Inbox	A quick ask from me before I return to debate prep - , -- Kamala Kamala Harris is running for president to fight for justice for the American people. With fewer than 80 days until
			Kamala Harris	Inbox	I would like your input on our debate strategy - , -- Kamala You can unsubscribe from this mailing list at any time: <a href="http://action.kamalaharris.org/cms/unsubscribe/">http://action.kamalaharris.org/cms/unsubscribe/</a>
			kamalaharris.org	Inbox	Direct - and introduce Kamala to more people than ever before through targeted ads on TV and online. That's where you come in.
			Team Kamala	Inbox	Who, what, when, where, why - them to Kamala and share her message. HOW: Click here to commit to make calls for our Call-In For Iowa weekend of action, and
			Kamala HQ	Inbox	Look at this progress! - Team Kamala You can unsubscribe from this mailing list at any time: <a href="http://action.kamalaharris.org/cms/unsubscribe">http://action.kamalaharris.org/cms/unsubscribe</a>
			Kamala Harris	Inbox	I fully intend to fight and to win with you by my side - , -- Kamala You can unsubscribe from this mailing list at any time: <a href="http://action.kamalaharris.org/cms/unsubscribe">http://action.kamalaharris.org/cms/unsubscribe/</a>

Emails are virtually costless. Very easy to ask: Are people more likely to open them with X subject or Y subject or Z subject?

## **3.10 Wrapping Up Causation with Experiments**

In this section, we have discussed what it means to make a causal claim, why it is essentially impossible to make causal comparisons in real life due to the fundamental problem of causal inferences, and how experiments can help us make comparisons that approximate our causal ideals.

In the next section, we start to examine how to visualize data.

### **3.10.1 Summary of R tools in this section**

Here are some of the R tools we used in this section:

- **table()**: this function summarizes the frequency of observations that take a particular value. The input is one or more variables in your data.
  - E.g., `table(resume$sex)` or `table(resume$sex, resume$call)`
- **tapply()**: this function applies a given operation like `mean` to whichever variable is in the first position, separately or “conditionally” by different values of the variable in the second “index” position.
  - E.g., `tapply(resume$call, INDEX=resume$race, mean)` finds the average callbacks for applicants separately for different races of applicants in the data.
- **== > < >= <= !=**: Relational operators help us set up “logical statements” in R that are evaluated as TRUE or FALSE

- E.g., `resume$race == "black"` evaluates whether for each observation in the race column is “black” in which case the statement is TRUE or not black, in which case the statement is FALSE
- E.g., `resume$call < 1` evaluates whether for each observation in the call column has a value less than one in which case the statement is TRUE or not less than 1, in which case the statement is FALSE
- We can then isolate certain parts of columns using relational operators and the brackets `[]`. For example we can take the mean callbacks for applicants who are black using `mean(resume$call[resume$race == "black"])`
- `&` and `|`: These are boolean operators that allow us to combine multiple relational operators using an AND statement (`&`) or an OR statement `|`. Note the bar is a bar that is usually above your backslash key and not a capitalized i.
  - E.g., `mean(resume$call[resume$race == "black" & resume$sex == "female"])`
- `subset()`: We can subset whole rows of our data using this function. It takes two inputs—the first is the name of the original dataframe, and the second is a relational statement. Usually we store this output in R by assigning the results to a new object, a dataframe that contains only those rows for which the logical statement using the relational operators is true. E.g., `females <- subset(resume, sex == "female")` subsets our data to keep only those rows where applicants were female.

# 4 Visualization

In this section, we discuss a set of tools for data visualization in R.

Goals of data visualization

- Communicate information
  - Transparently (show me the data!)
  - Quickly
  - Simply
  - Accurately
  - And with a little work: beautifully

There are many resources for ideas and best practices for data visualization. See [here](#) and [here](#).

We will cover many types of visuals, each typically designed for a different purpose.

What to communicate?

- Data summary
  - Central tendency (e.g., mean, median)
  - Spread (e.g., standard deviation, IQR)
- Comparison
  - e.g., Callback rates for black vs. white sounding names
- Trend
  - e.g., Economic confidence over time
- Relationship
  - e.g., Correlation

## 4.1 Application: Social Status and Economic Views

We are going to explore different types of visualizations through different social science examples. The first application we visit is a survey experiment.

Thal, A. (2020). The desire for social status and economic conservatism among affluent Americans. *American Political Science Review*, 114(2), 426-442.

In the experiment, affluent Americans are randomly assigned to encounter Facebook posts in which others broadcast their economic success. These posts are designed in a way that encourages affluent respondents to view economic success as a means of achieving social status.

Causal claims

- “I expect that exposure to these posts will cause affluent Americans to become more supportive of conservative economic policies.”
- “I also expect that exposure to these posts will cause especially large increases in economic conservatism among affluent men.”

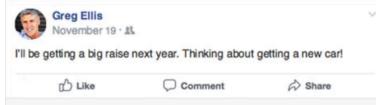
The experiment includes a sample of 2010 affluent Americans— people who report household incomes in the top 10 percent of the U.S. income distribution.

Experiment Ingredients:

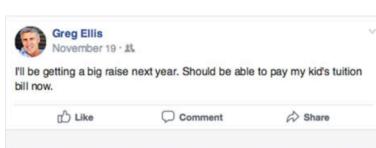
- Causal Question: Does desire for social status influence economic views of affluent Americans?
- Recruitment: Ask affluent Americans to take a survey online
- Randomization: Randomly assign respondents to view different fictional Facebook posts designed to signal different motivations
- Outcome: an index based on respondents’ support for decreasing “taxes on households making \$150,000 or more a year,” support for decreasing the “taxes on money people make from selling investments, also referred to as capital gains,” and support for decreasing “government regulation of business and industry.”
- Comparison: Average economic views between experimental conditions.

Snapshot of status conditions

**TABLE 3. Description of Experimental Conditions**

Condition name and sample size	Variation	Example post
<i>Status I: Social approval</i> Affluent, $n = 375$ Nonaffluent, $n = 205$	Added "Likes" and positive comments from Facebook friends	
<i>Status II: Self-esteem</i> Affluent, $n = 390$ Nonaffluent, $n = 210$	Added emoji and text signaling feelings of self-esteem	
<i>Status III: Conspicuous consumption</i> Affluent, $n = 392$ Nonaffluent, $n = 213$	Added announcement of luxury purchase	

#### Snapshot of Concrete and Placebo comparison conditions

<i>Concrete</i> Affluent, $n = 391$ Nonaffluent, $n = 209$	Added indication of concrete material need	
<i>Placebo</i> Affluent, $n = 394$ Nonaffluent, $n = 208$	Replaced announcement of economic success with announcement of noneconomic success	

Can you put this into the potential outcomes framework?

## 4.2 Boxplots

For a video explainer of the code for boxplots and barplots, see below. The video only discusses the code. Use the notes and lecture discussion for additional context. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)

<https://www.youtube.com/watch?v=QmQr4lfrmUc>

Let's load the data! Here, note that the data file is in a .RData format instead of .csv. This means that instead of using `read.csv`, we should use a function to load the data that is suitable for the .RData format. This will be `load`. That function works the following way:

```
load("status.RData")
```

After running the above code, an object will show up in your R environment.

```
head(status)
```

	condition	male	econcon
2	Concrete	1	0.7500000
3	Self-Esteem	1	1.0000000
4	Placebo	1	0.6666667
5	Self-Esteem	0	0.2500000
6	Self-Esteem	0	1.0000000
7	Social Approval	0	0.8333333

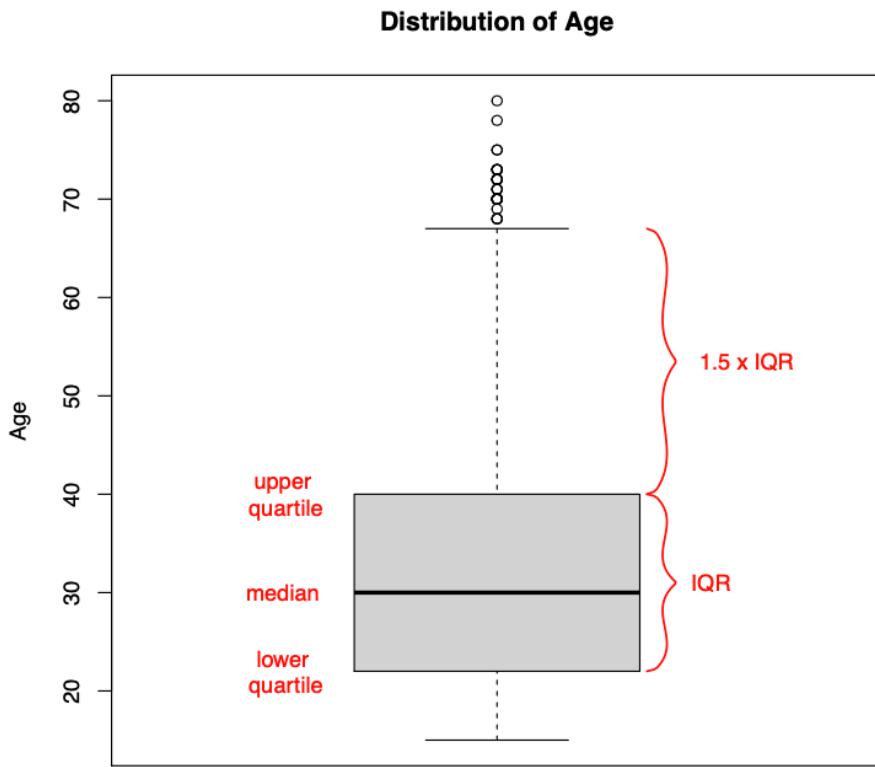
The data include the following variables

- `condition`: Placebo, Concrete, Self-Esteem, Social Approval, Conspicuous Consumption
- `gender`: 1= male; 0= otherwise
- `econcon`: Economic views. Numeric variable from 0 to 1, with higher values reflecting more conservative views

### 4.2.1 Data Summary: Boxplot

Characterize the distributions of continuous numeric variables at once

- Features: box, whiskers, outliers
- We will supply the function with a column in our data, and the boxplot displays the distribution of that variable.



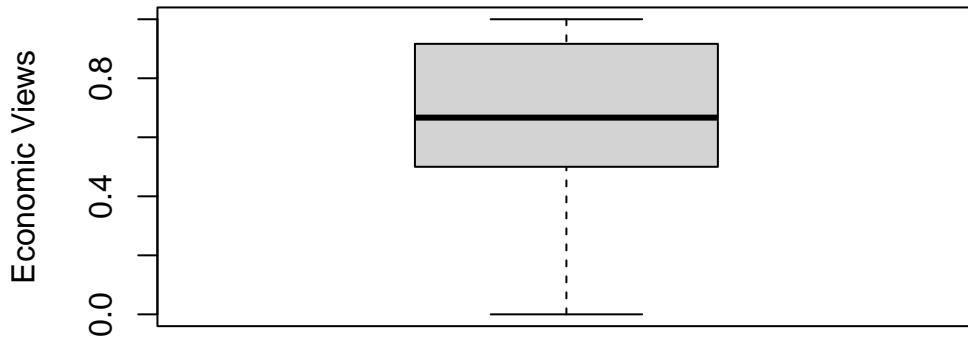
*Figure from Will Lowe*

Here is an example of the boxplot using our econcon variable.

- We have added a title and y-axis label to the plot through the `main` and `ylab` arguments.  
Play around with changing the words in those arguments.

```
boxplot(status$econcon,
 main="Economic Views in the Survey Sample",
 ylab="Economic Views")
```

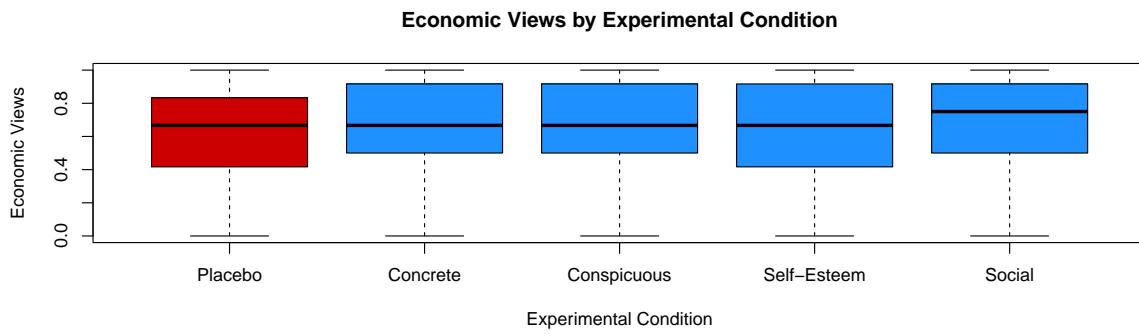
## Economic Views in the Survey Sample



After you execute the plot code, a preview of the plot should appear in the bottom-right window of RStudio.

Boxplots are also useful for data summary across multiple distribution: `boxplot(y ~ x, data = d)`

```
boxplot(econcon ~ condition, data=status,
 main="Economic Views by Experimental Condition",
 ylab="Economic Views",
 names = c("Placebo", "Concrete", "Conspicuous",
 "Self-Esteem", "Social"),
 xlab = "Experimental Condition",
 col = c("red3", rep("dodgerblue", 4)))
```



The additional arguments are just aesthetics. Play around with different settings.

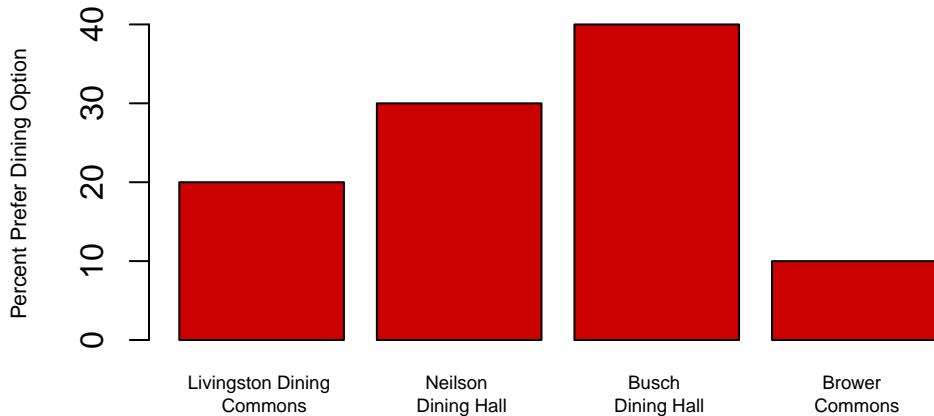
- For example, can you change the code to make the first two boxes red? Colors are supplied as a vector using the `col =` argument.
  - To explore colors in R, run this function `colors()` in your R console.

How should we interpret these results? Does status or social approval motivation, specifically, influence economic views? What about other potential motivations?

### 4.3 Barplots

Comparing frequencies (raw N), proportions, and/or means across categories

## Hypothetical Evaluation of RU Dining



We will use the `barplot()` function.

- In contrast to the boxplot, the barplot function takes a vector of values that will serve as the top of the bars in the plot— it does not summarize a variable from within the function
  - E.g., we could supply it a set of means to plot, not a raw variable
- Many of the other arguments are aesthetics similar to those when working with boxplot.
- This means that barplots are pretty easy to create in R. We can supply it a short vector of any values (e.g., `valuesbar <- c(20, 30, 40, 10)`), and we could also supply it a vector of any names to label those values.

```
Example
valuesbar <- c(20, 30, 40, 10)

namesbar <- c("Livingston Dining \n Commons",
 "Neilson \n Dining Hall",
 "Busch \n Dining Hall",
 "Brower \n Commons")

barplot(valuesbar,
 names=namesbar,
 cex.names = .6,
```

```

main="Hypothetical Evaluation of RU Dining",
ylab="Percent Prefer Dining Option",
cex.lab = .7,
col="red3")

```

- For real applications, this means we could supply a barplot with the output of a `tapply`

For example, in experiments, we may use barplots to compare the mean from the treatment group(s)  $\bar{Y}(1)$  to the control  $\bar{Y}(0)$  on some outcome. Let's do it!

- First, we need the means. Let's find the conditional means of economic views.

```

condmeans <- tapply(status$econcon, status$condition, mean)
condmeans # save as object to supply to the barplot function

```

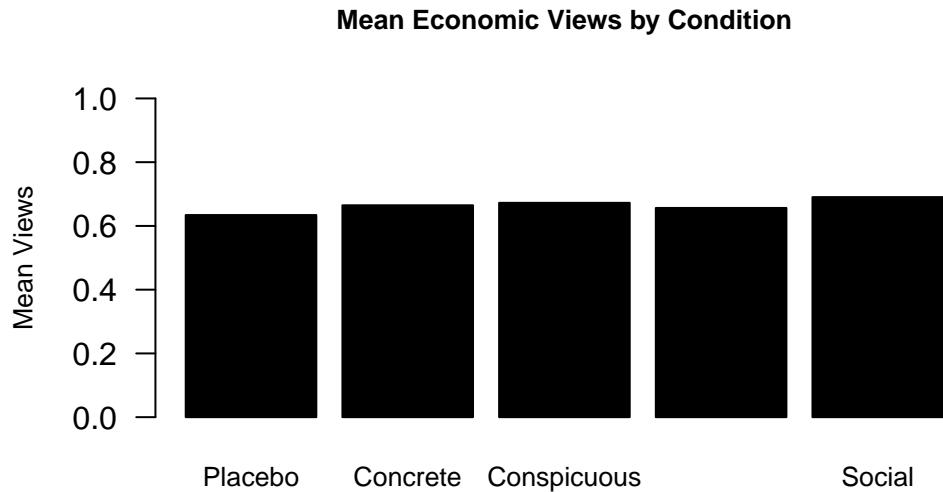
	Placebo	Concrete	Conspicuous	Consumption
	0.6340948	0.6647485		0.6724065
Self-Esteem		Social Approval		
	0.6564103	0.6904444		

The first input is the vector of means/proportions/frequency you want to plot.

```

barplot(condmeans,
 ylim = c(0,1), # y-axis dimensions
 names = c("Placebo", "Concrete", "Conspicuous",
 "Self-Esteem", "Social"),
 col = "black", # color of bars
 main = "Mean Economic Views by Condition", # plot title
 cex.main = .8, # size of plot title
 cex.names = .8, # size of name labels
 ylab = "Mean Views", # yaxis label
 cex.lab = .8, # size of yaxis label
 las = 1) # controls angle of axis labels

```



The remaining arguments alter the look of the plot to make it more informative.

- How could we improve this plot to make the interpretation easier?

#### 4.3.1 Saving Plots

You can save an image of your plot as a `png()` to your working directory. Place `png()` just before your plot with a name in quotations, and then specify the dimensions. Place `dev.off()` at the bottom.

```
png("mybarplot.png", width = 7, height = 4, res=300, units="in")
barplot(condmeans,
 ylim = c(0,1), # y-axis dimensions
 names = c("Placebo", "Concrete", "Conspicuous",
 "Self-Esteem", "Social"),
 col = "black", # color of bars
 main = "Mean Economic Views by Condition", # plot title
 cex.main = .8, # size of plot title
 cex.names = .8, # size of name labels
 ylab = "Mean Views", # yaxis label
 cex.lab = .8,# size of yaxis label
```

```

 las = 1) # controls angle of axis labels
dev.off()

```

Alternatively, you can save it as an image, by going to the plot window in your RStudio environment, and clicking on Export -> Save as Image. Here, you can save it in any file format you would like, as well as change the dimensions.



### 4.3.2 Creating New Variables

The author theorizes that social approval, self-esteem, and conspicuous consumption are all elements of “status motivation.” We could analyze the results by collapsing them into a single category called “status motivation” and compare it to the other experimental groups.

- Create a new variable `conditionnew`
- Code the variable into new categories based on the values in the original `condition` variable
- Check the class of the new variable and convert if necessary

- Verify new variable by exploring values

```

status$conditionnew <- NA # create new variable
Code new variable
status$conditionnew[status$condition == "Placebo"] <- "Placebo"
status$conditionnew[status$condition == "Concrete"] <- "Concrete"
status$conditionnew[status$condition == "Conspicuous Consumption" |
 status$condition == "Self-Esteem" |
 status$condition == "Social Approval"] <- "Status"

class(status$conditionnew) check the class
status$conditionnew <- as.factor(status$conditionnew) # convert

```

Recall, an alternative way to create the new variable is through an `ifelse` statement.

- Can be read: If this relational statement is TRUE, I assign you A, otherwise I assign you B
- This often works best when we change factor variables to character

```

status$conditionnew2 <- as.character(status$condition)
status$conditionnew2 <- ifelse(status$condition == "Conspicuous Consumption" |
 status$condition == "Self-Esteem" |
 status$condition == "Social Approval",
 "Status", status$conditionnew2)
status$conditionnew2 <- as.factor(status$conditionnew2)
table(status$conditionnew2)

```

Concrete	Placebo	Status
391	394	1157

Note: Barplots don't have to display means. We could also display frequencies. For example, let's make a plot of the number of people in each condition using our new variable.

```

freqobs <- table(status$conditionnew)

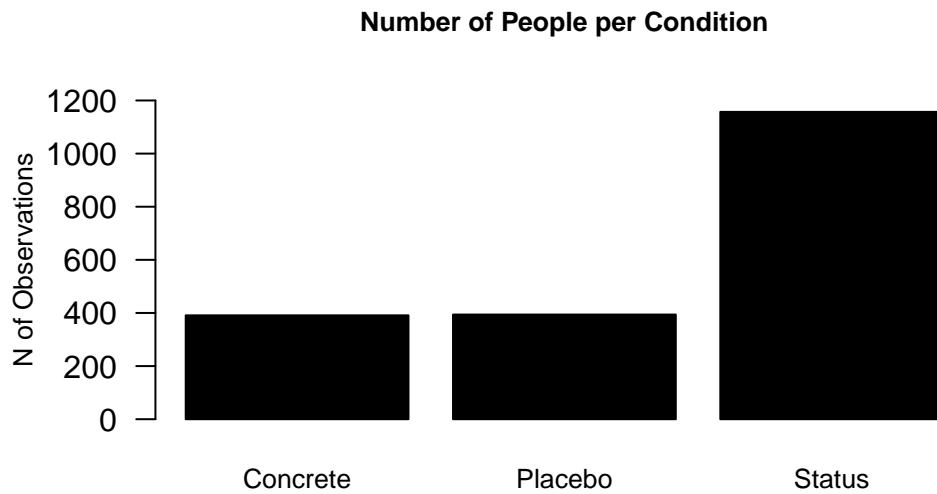
barplot(freqobs,
 ylim = c(0, 1200),
 col = "black", # color of bars
 main = "Number of People per Condition", # plot title
 cex.main = .8, # size of plot title

```

```

cex.names = .8, # size of name labels
ylab = "N of Observations", # yaxis label
cex.lab = .8,# size of yaxis label
las = 1) # controls angle of axis labels

```



## 4.4 Application: Changing Minds on Gay Marriage

We now turn to a study that asks the question

- Research Question Can we effectively persuade people to change their minds?
  - *Contact Hypothesis*: outgroup hostility diminishes through extended positive contact

The authors conduct two randomized control trials in Los Angeles

- *Target population*: voters in Los Angeles
- *Recruitment*: select people from a registered voter list
- *Randomized treatment conditions*:
  - Canvassers have a conversation about same-sex marriage vs.
  - Recycling scripts (placebo)

- Control group: no canvassing
- *Outcome measures:*
  - Feeling towards gay couples (survey responses over multiple waves)
- *Comparison*
  - Compare average change in feelings between treatment conditions

Let's load the data. Data available through QSS. See QSS Chapter 2 for additional discussion.

- **study:** Which study is the data from (1 = Study1, 2 = Study2)
- **treatment:** Five possible treatment assignment options
- **therm1:** Survey thermometer rating of feeling towards gay couples in waves 1 (0–100) (asked before people were canvassed)
- **therm2:** Survey thermometer rating of feeling towards gay couples in waves 2 (0–100) (asked after people were canvassed)

```
marriage <- read.csv("gayreshaped.csv", stringsAsFactors = T)

How many rows and columns
dim(marriage)
```

```
[1] 11948 6
```

```
How many observations in each treatment group, in each study
table(marriage$treatment, marriage$study)
```

	1	2
No Contact	5238	1203
Recycling Script by Gay Canvasser	1046	0
Recycling Script by Straight Canvasser	1039	0
Same-Sex Marriage Script by Gay Canvasser	1151	1238
Same-Sex Marriage Script by Straight Canvasser	1033	0

For a video explainer of the code for the barplot, scatter plot and histogram created with this application, see below. The video only discusses the code. Use the notes and lecture discussion for additional context. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)

<https://www.youtube.com/watch?v=ukexpAulAAk>

Let's focus on study 1 only.

```
marriage1 <- subset(marriage, study == 1)
```

We have to do some work to prepare our outcome and treatment conditions.

In experiments, we compare the mean from the treatment group(s)  $\bar{Y}(1)$  to the control  $\bar{Y}(0)$  on some outcome

- Here are outcome is Change in Support for gay couples: Wave 2 - Wave 1 feeling thermometer scores

```
marriage1$outcome <- marriage1$therm2 - marriage1$therm1
```

#### 4.4.1 Recall: Creating new variables

Let's create a new variable `treatmentnew` that collapses the two Recycling and Same-Sex marriage conditions.

```
marriage1$treatmentnew <- NA
marriage1$treatmentnew[marriage1$treatment == "No Contact"] <- "No Contact"
marriage1$treatmentnew[marriage1$treatment == "Recycling Script by Gay Canvasser" |
 marriage1$treatment ==
 "Recycling Script by Straight Canvasser"] <- "Recycling"
marriage1$treatmentnew[marriage1$treatment == "Same-Sex Marriage Script by Gay Canvasser" |
 marriage1$treatment ==
 "Same-Sex Marriage Script by Straight Canvasser"] <- "Marriage"
marriage1$treatmentnew <- as.factor(marriage1$treatmentnew)

table(marriage1$treatmentnew)
```

Marriage	No Contact	Recycling
2184	5238	2085

#### 4.4.2 Recall: Using `ifelse` to create new variable

An alternative way we could create a variable is to use `ifelse`

Let's try another way using the `ifelse` command.

- Can be read: If this relational statement is TRUE, I assign you A (in this case “No Contact”), otherwise (`ifelse()`)
- if this alternative relational statement is TRUE, I assign you B (in this case “Recycling”), otherwise (`ifelse()`)
- if this alternative relational statement is TRUE, I assign you C (in this case “Marriage”), otherwise
- If all of those were FALSE I assign you D (in this case an NA)

```
marriage1$treatmentnew2 <- ifelse(marriage1$treatment == "No Contact", "No Contact",
 ifelse(marriage1$treatment ==
 "Recycling Script by Gay Canvasser" |
 marriage1$treatment ==
 "Recycling Script by Straight Canvasser",
 "Recycling",
 ifelse(marriage1$treatment ==
 "Same-Sex Marriage Script by Gay Canvasser" |
 marriage1$treatment ==
 "Same-Sex Marriage Script by Straight Canvasser",
 "Marriage",
 NA)))
marriage1$treatmentnew2 <- as.factor(marriage1$treatmentnew2)
```

#### 4.4.3 Calculating the Average Treatment Effect

We now have our outcome and our treatment conditions. In an experiment, we want to look at the difference in means between conditions. Let's calculate the means.

```
outs <- tapply(marriage1$outcome, marriage1$treatmentnew, mean, na.rm=T)
```

Note: Sometimes data include missing cells. In R, these have an NA. To ignore these when calculating a mean, we add `na.rm = T` to the `mean()` or `tapply()` functions.

#### 4.4.4 Visualize means in a barplot

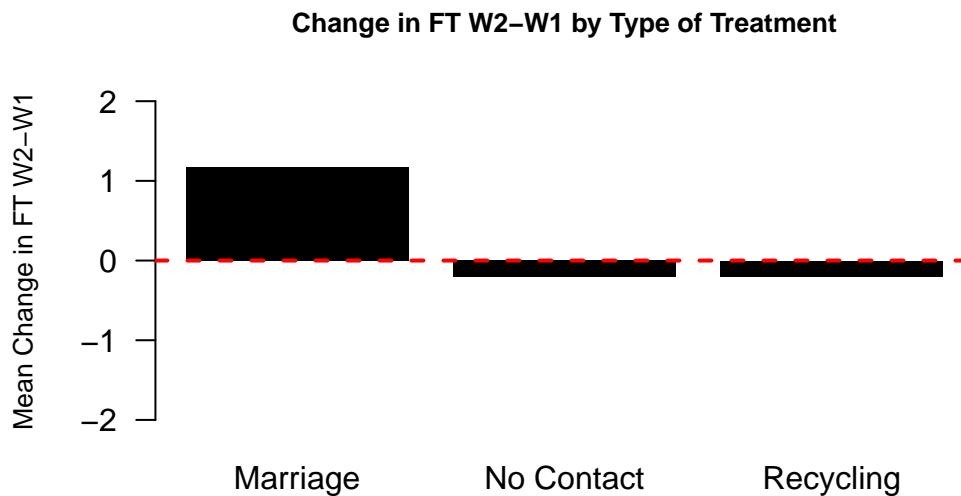
Let's also add a line at 0 using `abline()`

```
barplot(outs,
 col="black",
 ylim = c(-2, 2), # y-axis dimensions
 border = NA, # removes bar borders
```

```

main = "Change in FT W2-W1 by Type of Treatment", # plot title
cex.main = .8, # size of plot title
ylab = "Mean Change in FT W2-W1", # yaxis label
cex.lab = .8, # size of yaxis label
las = 1) # controls angle of axis labels
abline(h=0, lty=2, col = "red", lwd=2) # adds horizontal line at 0 with dashes

```



How should we interpret these results?

- In the Marriage condition, it looks like on average, views toward gay couples became warmer (the bar is positive) after the conversations with canvassers about same-sex marriage.
- In contrast, the views of people in the Recycling or No Contact conditions did not change much and if anything, became slightly colder.
- Comparing between these bars, then, it seems like there is an “average treatment effect” given that the change in the Marriage condition was different from the Recycling and No Contact control groups.

## 4.5 Scatterplots

It turns out that study was completely fabricated, and the article was eventually [retracted](#).

How did people know? Well a team of researchers became suspicious based on exploratory analyses they conducted with the data. Let's do a few of these to learn about scatterplots and histograms.

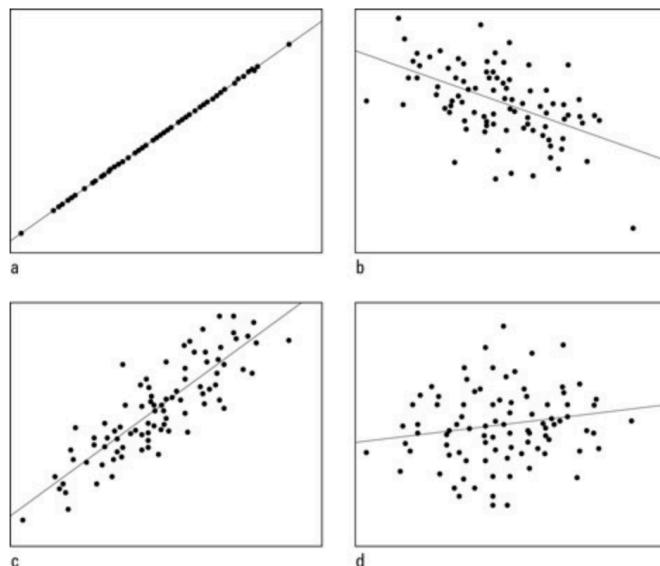
Scatter plots show the relationship between two numeric variables.

A common way to describe and quantify a relationship is through correlation.

- Correlation: When  $x$  changes,  $y$  also changes by a fixed proportion
  - Asks: If you are a certain degree above the mean of  $x$ , are you similarly that much above the mean of  $y$ ?
  - Positive correlation: data cloud slopes up;
  - Negative correlation: data cloud slopes down;
  - High positive or negative correlation: data cluster tightly around a sloped line
  - Not affected by changes of scale: cm vs. inch, etc.

Range of Correlation is between  $-1$  and  $1$

- Look at the graphs below for examples of high and low positive and negative correlations.



Scatterplots with correlations of a)  $+1.00$ ; b)  $-0.50$ ; c)  $+0.85$ ; and d)  $+0.15$ .

Figure 4.1: From *R for Dummies*

The `plot()` function in R works using  $x$  and  $y$  coordinates.

- We have to tell R precisely at which  $x$ - and  $y$ - coordinates to place points (e.g., place a point at  $x=20$  and  $y=40$ )

- In practice, we will generally supply R with a vector of x-coordinates and a vector of corresponding y-coordinates.

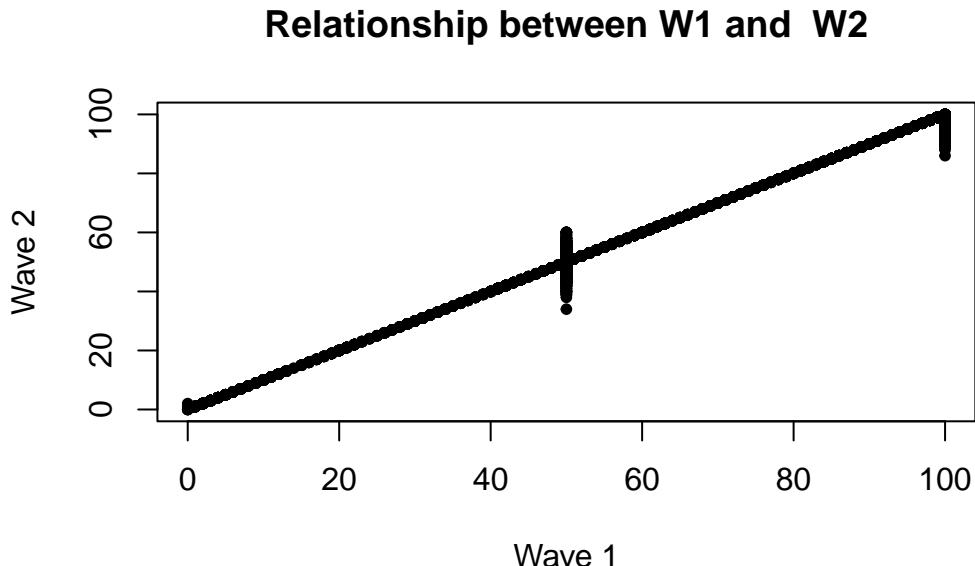
To illustrate a scatterplot, we will examine the relationship between the Wave 1 and Wave 2 feeling thermometer scores in the field experiment, for just the control “No Contact” condition.

```
Subset data to look at control only
controlonly <- subset(marriage1, treatment == "No Contact")
```

In the `plot()`, we supply the x and y vectors.

- `xlim` and `ylim` specify the range of the x and y axis.
- `pch` is the point type. You can play around with that number to view different plot types

```
plot(x=controlonly$therm1, y=controlonly$therm2,
 main = "Relationship between W1 and W2",
 xlab = "Wave 1", xlim = c(0, 100),
 ylab = "Wave 2", ylim = c(0, 100),
 pch = 20)
```



The correlation looks extremely high! It is positively sloped and tightly clustered.

In fact, if we use R's function to quantify a correlation between two variables, we will see it is a correlation above .99, very close to the maximum value.

- By default, R calculate the “pearson” correlation coefficient, a number that will be between -1 and 1. It represents the strength of the linear association between two variables.

```
use = "pairwise" means to use all observations where neither variable has missing NA data
cor(marriage1$therm1, marriage1$therm2, use = "pairwise")
```

```
[1] 0.995313
```

This high correlation was unusual for this type of data.

- Feeling thermometers suffer from low reliability. How a person answers the question at one point in time (perhaps 83) in Wave 1 often differs from the numbers they say when asked again at a future point in time in Wave 2. A person's responses often aren't that stable.
- Because there was such a high correlation, it suggested that the data might not have been generated by real human responses

## 4.6 Histograms

The researchers later discovered the Wave 1 data was suspiciously correlated with an existing survey: 2012 CCAP.

- They believe the researcher likely used CCAP for Wave 1 - used survey responses from real humans that took a real survey – but not the humans that the researcher claimed to interview in the experiment.
- Then the researcher generated the Wave 2 data by adding random noise to the Wave 1 data
- Part of why they believe this has to do with a histogram plot they generated to compare Waves 1 and 2

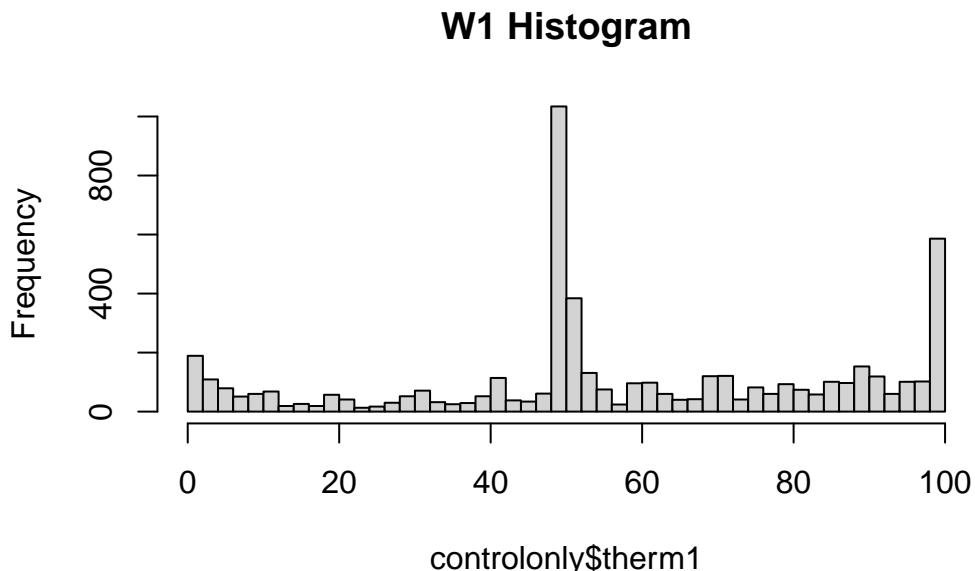
A histogram is a useful plot for summarizing the distribution of a single variable.

- It shows the frequency of observations (e.g., the number of survey respondents) who give an answer within a particular interval of numeric values

Because a histogram is a single variable summary, we just supply R with the numeric variable we want to summarize.

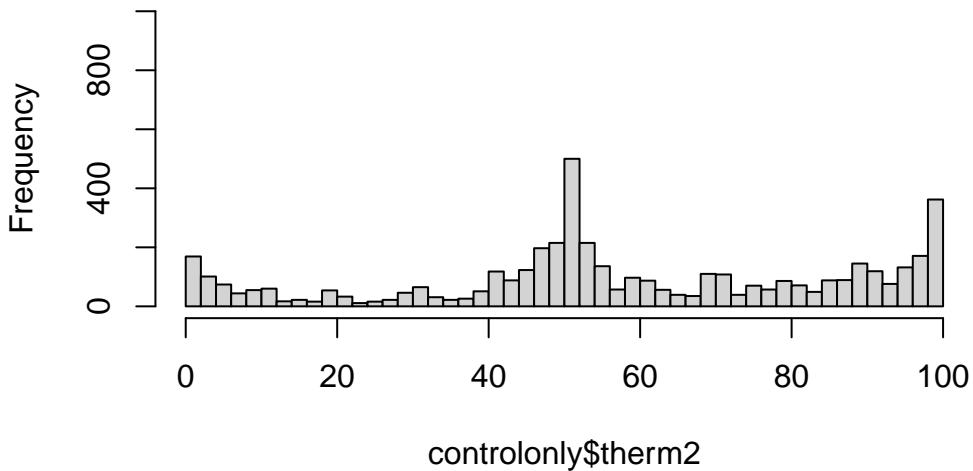
- The new argument here `breaks` tells R how many of the individual rectangles we want. You can play around with that number to see how the plot changes.

```
hist(x=controlonly$therm1, breaks=50,
 main = "W1 Histogram", ylim = c(0,1000))
```



```
hist(x=controlonly$therm2, breaks=50,
 main = "W2 Histogram", ylim = c(0,1000))
```

## W2 Histogram



The researchers noticed that the heaping patterns were different between Wave 1 and Wave 2.

- When real humans answer these types of feeling thermometer questions, we often see heaping (tall spikes) at values of 0, 50, and 100. Humans tend to gravitate toward those nice round numbers to anchor their responses. In addition, often researchers might recode people with missing responses (people who skip a question), as having a score of 50, increasing the number at that point.
  - Wave 1 has a lot of this heaping—look at the higher bars around 0, 50, and 100, suggesting a lot of survey respondents gave those answers.
  - However, Wave 2 has less heaping, particularly at 50. This suggested to the researchers that the Wave 2 data were likely generated by a computer and not real humans

### 4.6.1 Happy research ending

While the original article was retracted

- Researchers who found irregularities received funding to conduct similar studies with real data this time
- Multiple publications suggest the canvassing approach was effective:

- Broockman and Kalla. 2016. “Durably reducing transphobia: A field experiment on door-to-door canvassing” *Science* 352 no. 6282.
- Broockman and Kalla. 2020. “Reducing exclusionary attitudes through interpersonal conversation: evidence from three field experiments.” *American Political Science Review*
- Kalla and Broockman. 2021. “Which narrative strategies durably reduce prejudice? Evidence from field and survey experiments supporting the efficacy of perspective-taking.” *American Journal of Political Science*. Forthcoming.

## 4.7 Line Plots

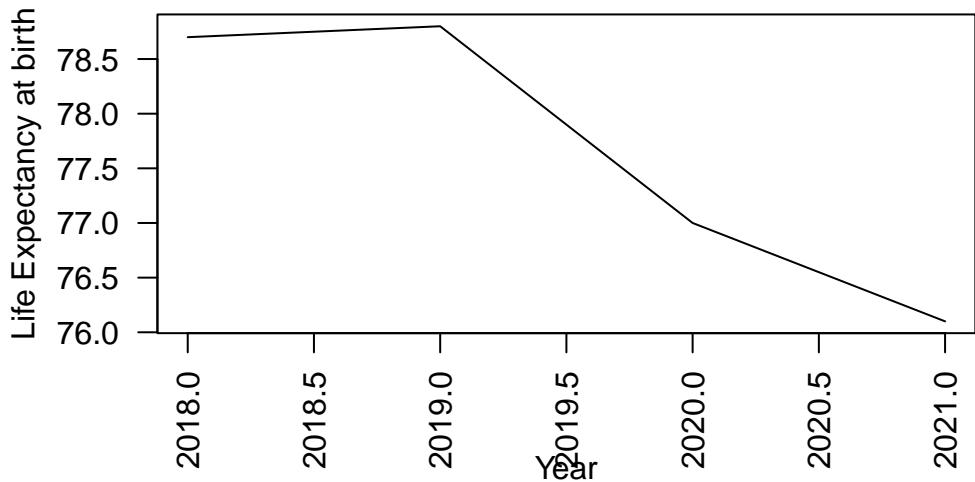
In this application, we will create a line plot in R. Line plots are built very similarly to scatterplots. We provide R an input of values for the x-axis and corresponding values on the y-axis.

For example, if we wanted to plot the US life expectancy for the past few years from 2018-2021, we could do the following based on data from the National Center for Health Statistics:

```
years <- c(2018, 2019, 2020, 2021)
yvalues <- c(78.7, 78.8, 77, 76.1)

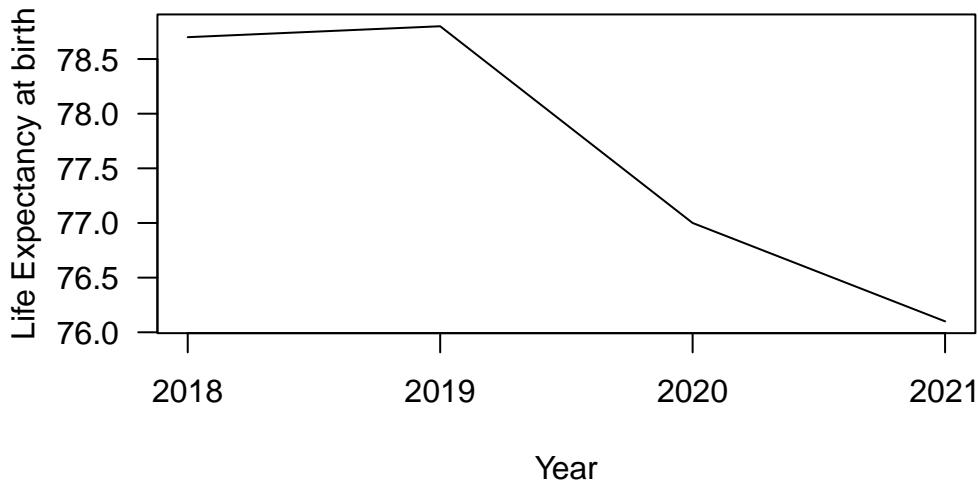
Key to making this a line plot is type="l"
plot(x=years,
 y=yvalues,
 type="l",
 main="US Life Expectancy by year",
 xlab = "Year",
 ylab = "Life Expectancy at birth",
 las=2) # orientation of axis labels
```

## US Life Expectancy by year



```
Alternate approach is to customize axis
plot(x=1:4,
 y=yvalues,
 type="l",
 main="US Life Expectancy by year",
 xlab = "Year",
 ylab = "Life Expectancy at birth",
 las=2, # orientation of axis labels
 xaxt="n") # remove original axis
axis(1, at=1:4, labels = years) # label points only at relevant ticks
```

## US Life Expectancy by year



The two vectors of values may come from variables in your data or may come from calculations you make using those variables.

Let's do a more complex example and break this down step by step.

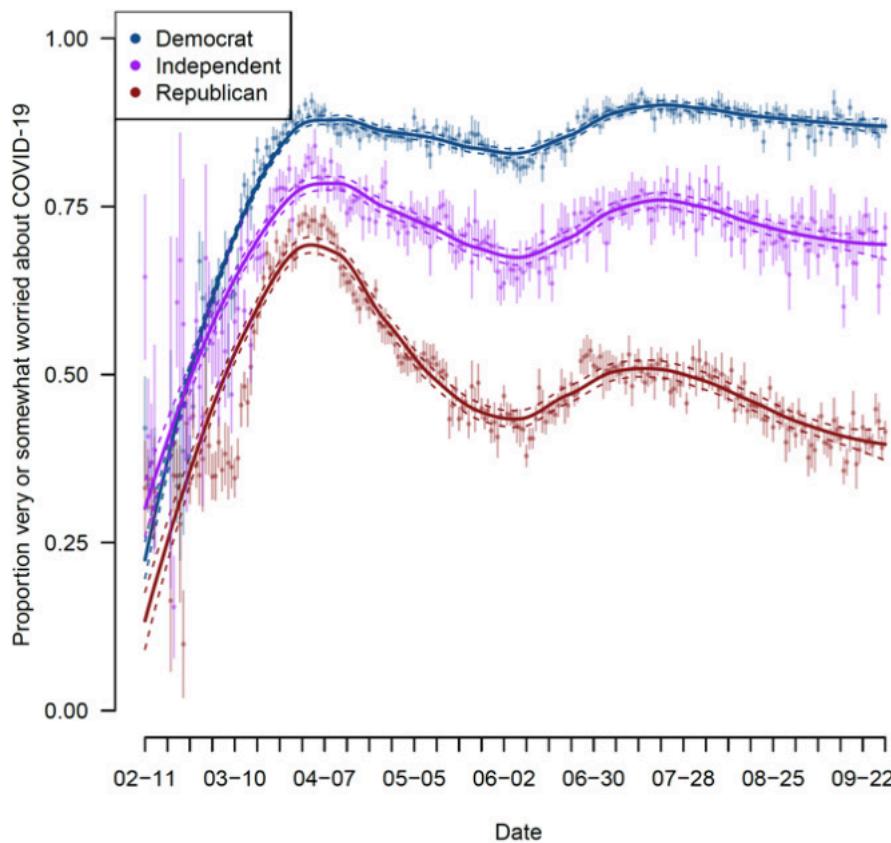
### 4.8 Application: Trends during COVID

Since the onset of the pandemic in 2020, researchers have evaluated attitudinal and behavioral responses to policy changes, political messages, and COVID case/hospitalization/death rates.

- Survey data on attitudes and self-reported behavior
- Health care provider administrative data
- Mobile phone data to track locations
- Social media data to track attitudes and mobility

*Example: Using Survey data from over 1.1 million responses to measure concern about the coronavirus over time.*

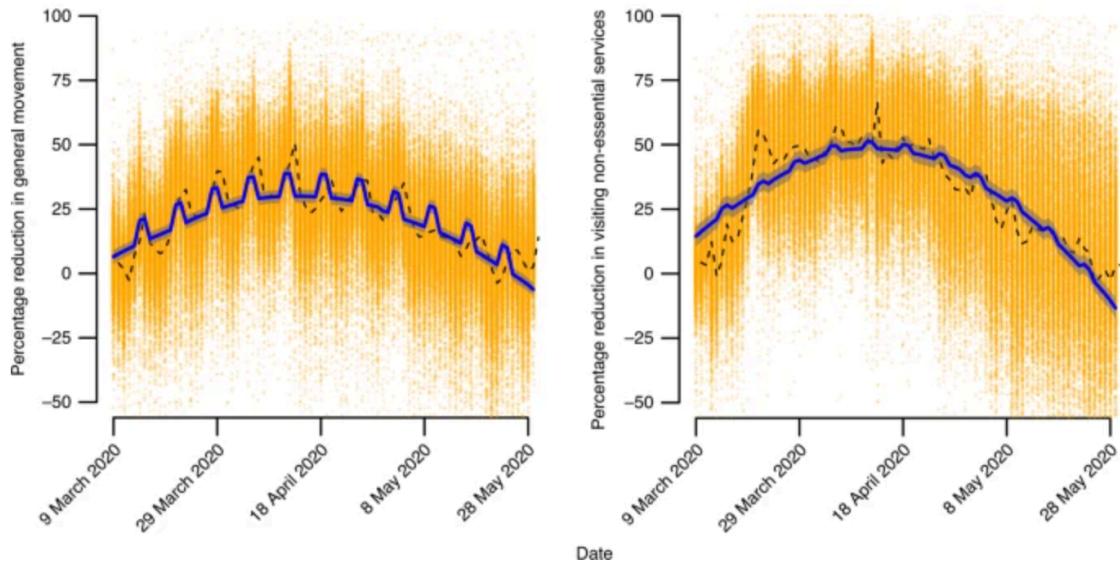
- Clinton, Joshua, et al. “[Partisan pandemic: How partisanship and public health concerns affect individuals' social mobility during COVID-19](#).” Science advances 7.2 (2021): eabd7204.



*Example: Using the geotracking data of 15 million smartphones per day to compute percentage reduction in general movement and visiting non-essential services relative to before COVID-19 (before March 9).*

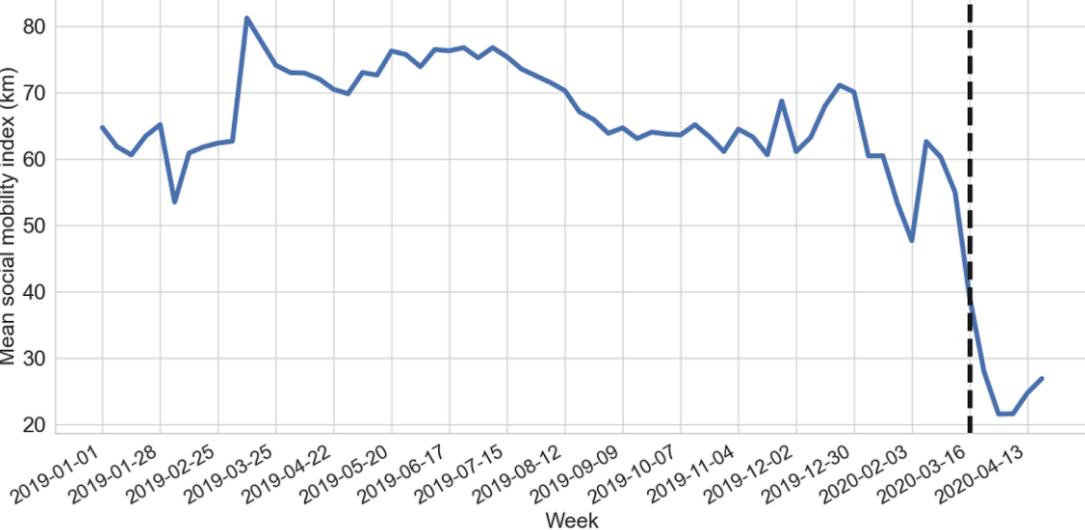
- Gollwitzer, Anton, et al. “[Partisan differences in physical distancing are linked to health outcomes during the COVID-19 pandemic](#).” Nature human behaviour 4.11 (2020): 1186-1197.

**Fig. 1: Physical distancing as a function of time (9 March to 29 May 2020).**



*Example:* **Using Twitter geolocation data** to track how much movement users have by looking at the distances from all locations where a given user has tweeted.

- Paiheng Xu, Mark Dredze, David A Broniatowski. “[The Twitter Social Mobility Index: Measuring Social Distancing Practices from Geolocated Tweets](#).” Journal of Medical Internet Research (JMIR), 2020.



**Figure 1.** Mean social mobility index (kilometers) in United States from January 1, 2019, to April 27, 2020. Weeks with missing data are excluded from the figure.

We will use the Twitter social mobility index to study how the movement of geo-located Twitter users changed from 2019 into April 2022.

- We will compare this movement for users located in the Northeast vs. South

Each row of the dataset represents a week of the year. Each column represents a particular geography for which social mobility was calculated by the researchers.

- **Dates** indicates the date
- **Northeast**: social mobility data for those in the northeast of the U.S.
- **South**: social mobility data for those in the south of the U.S.

```
Load the data from the author Mark Dredze's website
covid <- read.csv("https://raw.githubusercontent.com/mdredze/covid19_social_mobility.github.io/master/covid19_social_mobility.csv")
```

Just like we have encountered numeric, factor, and character variables, R also has the ability to treat variables specifically as dates. We will want R to treat the date variable we read in as a date, and not as raw text or some other variable type. To do this, we will use the `as.Date` function.

```
Date variable original format and class
head(covid$Dates)
```

```
[1] "2019-01-01" "2019-01-07" "2019-01-14" "2019-01-21" "2019-01-28"
```

```
[6] "2019-02-04"
```

```
class(covid$Dates)
```

```
[1] "character"
```

```
Convert to class Date
covid$Dates <- as.Date(covid$Date)
head(covid$Dates)
```

```
[1] "2019-01-01" "2019-01-07" "2019-01-14" "2019-01-21" "2019-01-28"
[6] "2019-02-04"
```

```
class(covid$Dates)
```

```
[1] "Date"
```

The researchers continue to add to these data. Let's look at the portion of data from 2019 to April 2022.

- Note the use of `as.Date` again to make sure R knows our text should be treated as a date
- Note the use of the greater than or equal to `>=` and less than or equal signs `<=` to specify which rows we want to keep in the data. We want rows that are in dates after January 1, 2019 and (`&`) on or before April 25, 2022.

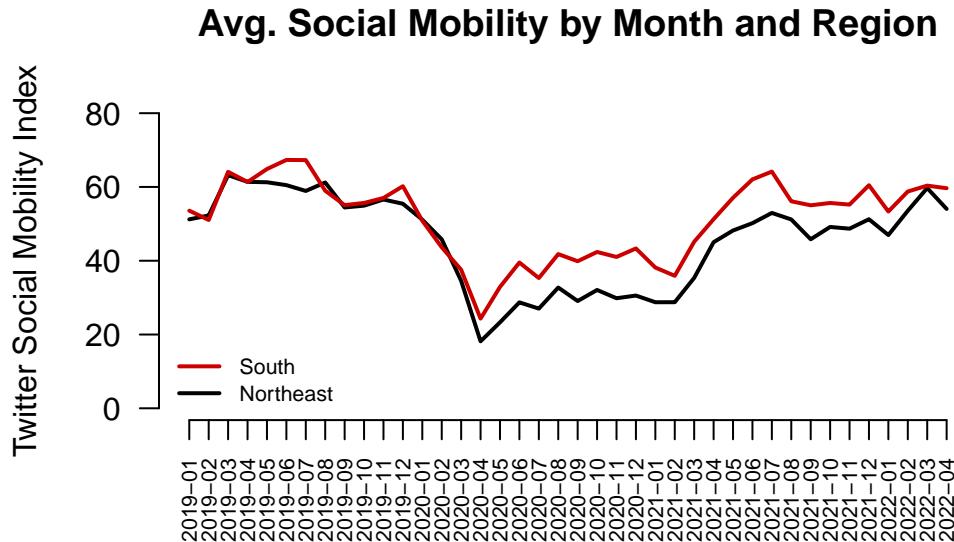
```
covidsub <- subset(covid, Dates >= as.Date("2019-01-01") &
 Dates <= as.Date("2022-04-25"))
```

These data are collected by week. That is very detailed. While that may be useful, let us create another variable that contains just the month and year, which will allow us to calculate the average per month. With a date variable, we can use the `format` function to change the format to just year and month.

```
covidsub$monthyear <- format(covidsub$Dates, "%Y-%m")
range(covidsub$monthyear)
```

```
[1] "2019-01" "2022-04"
```

Where we are going ...



Starting from the bottom ...

- Let's first create a scatterplot by providing R with our two variables
- In a trend/line plot, we want each month on the x-axis
- We want our outcome on the y-axis, in this case, average social mobility by month
- Ultimately we will want to compare the Northeast with the South. We will plot one line at a time, starting with the Northeast

We first need to find the average by month. Recall our `tapply()` function.

```
mobilitybymonthNE <- tapply(covidsub$Northeast, covidsub$monthyear, mean,
 na.rm=T)

mobilitybymonthSO <- tapply(covidsub$South, covidsub$monthyear, mean,
 na.rm=T)
```

Let's look at the output for the Northeast. Each value is what we ultimately want on the y-axis— the average social mobility in a given month.

```
mobilitybymonthNE
```

```
2019-01 2019-02 2019-03 2019-04 2019-05 2019-06 2019-07 2019-08
```

```

51.22066 52.26420 63.20130 61.38417 61.27622 60.49753 58.91779 61.20730
2019-09 2019-10 2019-11 2019-12 2020-01 2020-02 2020-03 2020-04
54.44546 54.93814 56.59830 55.44538 51.12414 45.80660 34.55917 18.15076
2020-05 2020-06 2020-07 2020-08 2020-09 2020-10 2020-11 2020-12
23.29190 28.71901 27.02149 32.73828 29.07536 32.07877 29.83641 30.56208
2021-01 2021-02 2021-03 2021-04 2021-05 2021-06 2021-07 2021-08
28.75507 28.76227 35.35340 45.02537 48.19897 50.18401 52.96105 51.19241
2021-09 2021-10 2021-11 2021-12 2022-01 2022-02 2022-03 2022-04
45.81695 49.15654 48.69051 51.24941 46.96813 53.55241 59.70933 54.04312

```

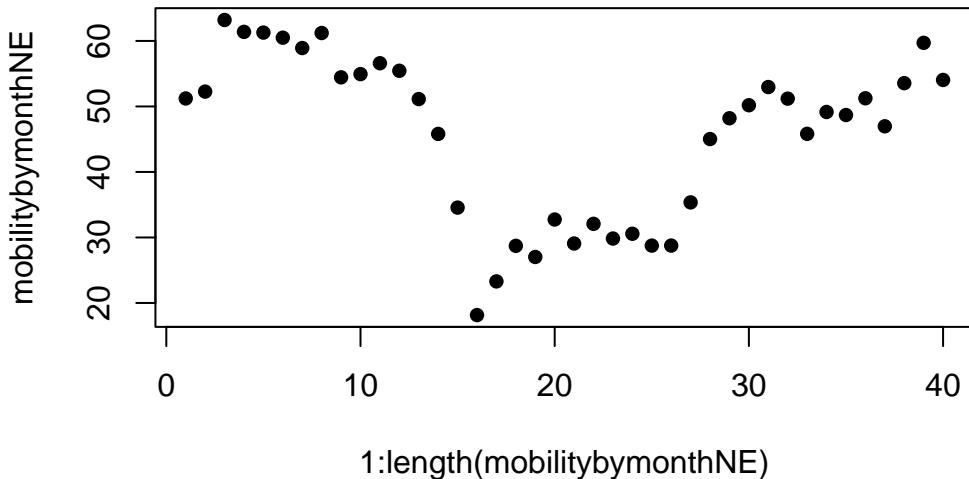
We want to plot them each at their own point on the x-axis, from the first month to the last month. We can start by creating a vector of the same length as we have months:

```
1:length(mobilitybymonthNE)
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```

These become our two inputs in the plot.

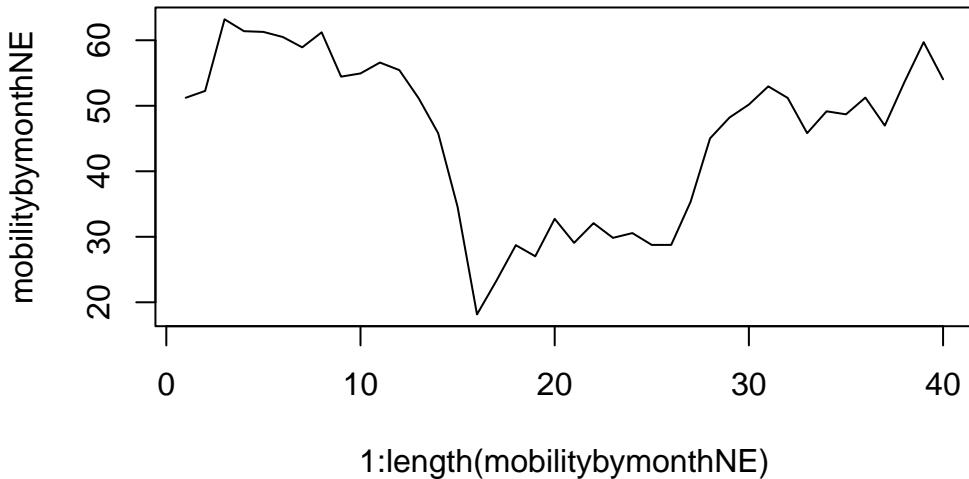
```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE, pch=16) # pch is point type
```



We now transform it to a line by specifying `type="l"`

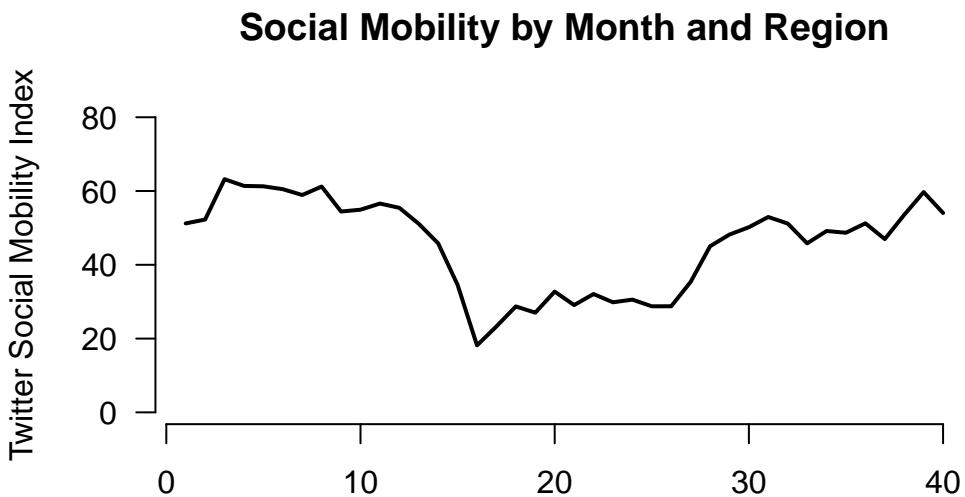
- By default, R creates a plot with `type=p` for points. R also has `type=b` which has both a line and points.

```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE, type="l") # makes it a line
```



Let us change the aesthetics a bit by adding labels and removing the border with `bty="n"`.

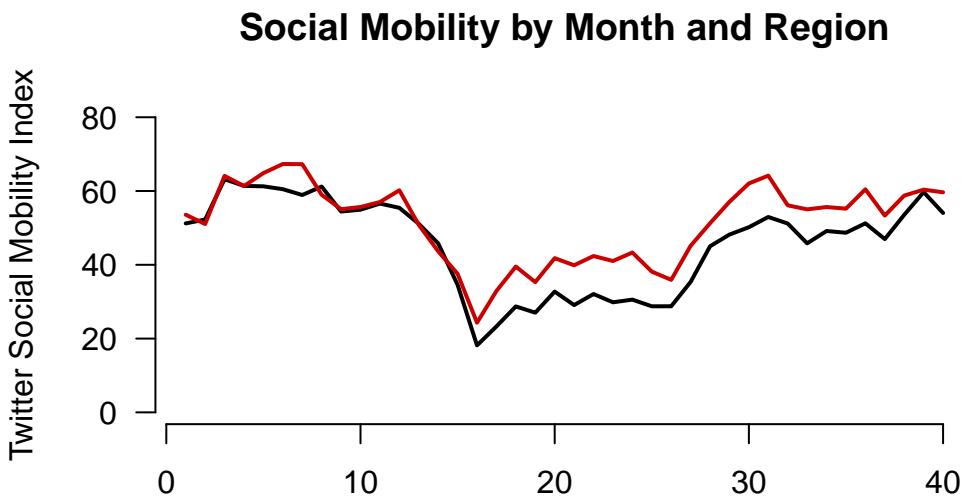
```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80), # y-axis limits
 las=1, # orientation of axis labels
 lwd=2, # line width
 bty="n") # removes border
```



Let's add a comparison line with the `lines()` function to look at trends for the south.

- Note that this is outside of the `plot()` function, but the inputs are very similar. We supply a set of x and y coordinates.

```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80), # y-axis limits
 las=1, # orientation of axis labels
 lwd=2, # line width
 bty="n") # removes border
Add line to the plot
lines(x=1:length(mobilitybymonthSO),
 y=mobilitybymonthSO, col="red3", lwd=2)
```



Let's create our own axis for the plot to add detail. To do this, we add `xaxt` to the `plot` function and then use `axis()` below the function.

The labels we will add are the actual months in the data. These happen to be the labels or `names` of our vectors:

```
names(mobilitybymonthNE)

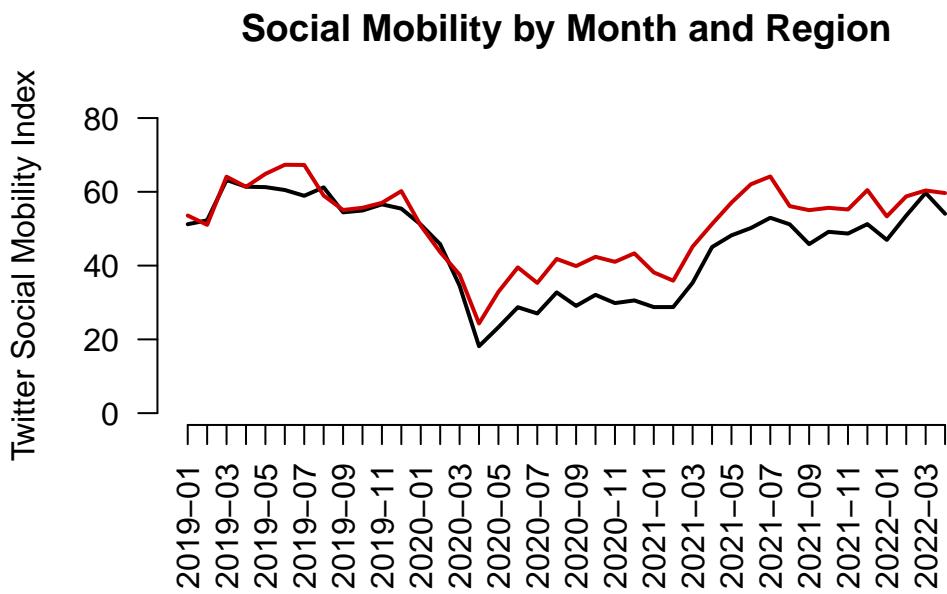
[1] "2019-01" "2019-02" "2019-03" "2019-04" "2019-05" "2019-06" "2019-07"
[8] "2019-08" "2019-09" "2019-10" "2019-11" "2019-12" "2020-01" "2020-02"
[15] "2020-03" "2020-04" "2020-05" "2020-06" "2020-07" "2020-08" "2020-09"
[22] "2020-10" "2020-11" "2020-12" "2021-01" "2021-02" "2021-03" "2021-04"
[29] "2021-05" "2021-06" "2021-07" "2021-08" "2021-09" "2021-10" "2021-11"
[36] "2021-12" "2022-01" "2022-02" "2022-03" "2022-04"

plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80),
```

```

 las=1,
 lwd=2,
 bty="n",
 xaxt="n") # removes original x-axis
Add line to the plot
lines(x=1:length(mobilitybymonthSO),
 y=mobilitybymonthSO, col="red3", lwd=2)
add the axis the "1" means x-axis. A "2" would create a y-axis
axis(1, at = 1:length(mobilitybymonthNE),
 labels=names(mobilitybymonthNE), las=2)

```



Finally, let's add a `legend()`. Now we're here!

```

plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80),
 las=1,

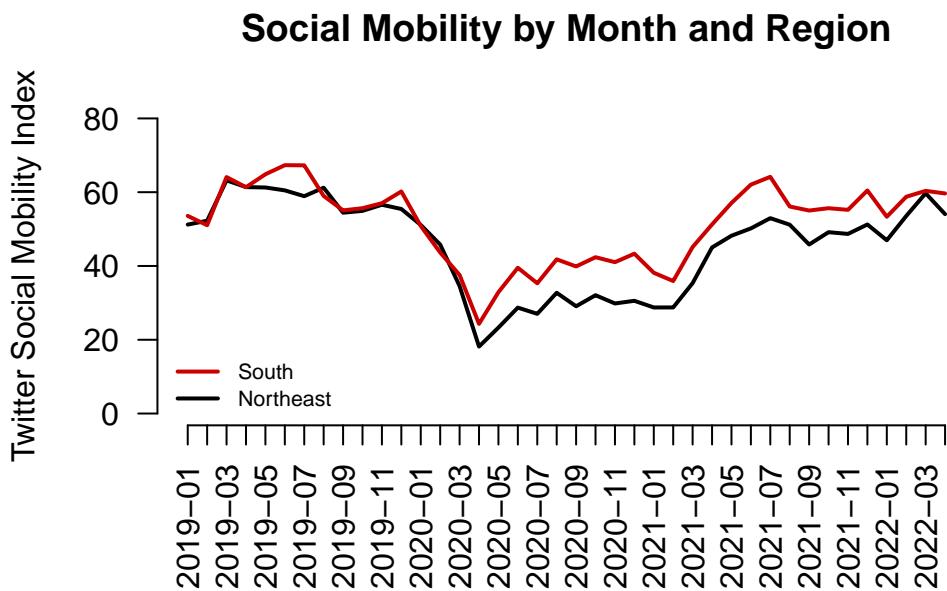
```

```

lwd=2,
bty="n",
xaxt="n") # removes original x-axis
Add line to the plot
lines(x=1:length(mobilitybymonthS0),
 y=mobilitybymonthS0, col="red3", lwd=2)
add the axis the "1" means x-axis. A "2" would create a y-axis
axis(1, at = 1:length(mobilitybymonthNE),
 labels=names(mobilitybymonthNE), las=2)

Add legend, "bottomleft" indicates where on the plot to locate it
Could use "topright" instead, for example
legend("bottomleft", col=c("red3", "black"),
 c("South", "Northeast"),
 cex = .7, # size of legend
 lwd=2,
 bty="n")

```



## 4.9 Visual tips and tricks

Recall we said the goals of visualization are to communicate information

- Transparently (show me the data!)
- Quickly
- Simply
- Accurately
- And with a little work: beautifully

What NOT to communicate?



Claus Wilke provides an overview of rules of thumb to fall when creating a data visualization on the Serial Mentor [website](#).

An example is below

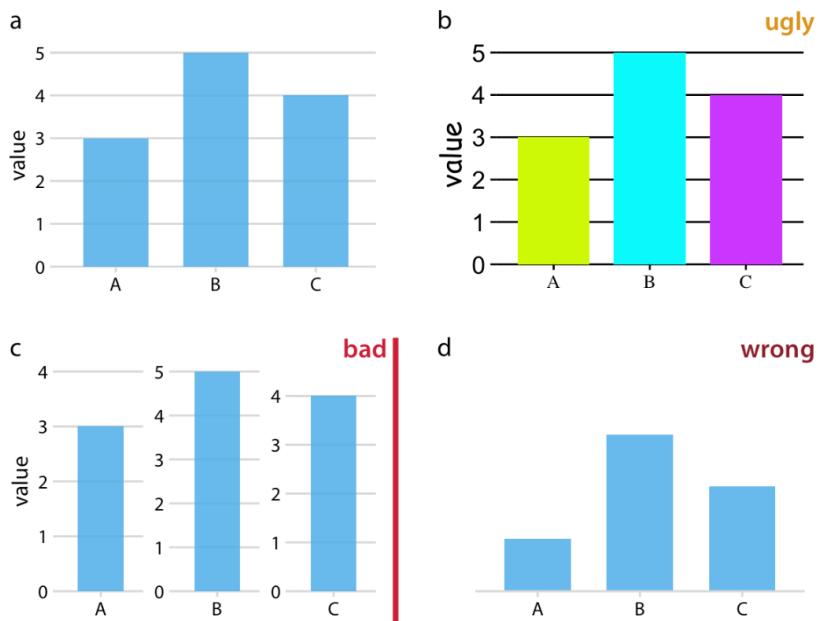


Figure 1.1: Examples of ugly, bad, and wrong figures. (a) A bar plot showing three values ( $A = 3$ ,  $B = 5$ , and  $C = 4$ ). This is a reasonable visualization with no major flaws. (b) An ugly version of part (a). While the plot is technically correct, it is not aesthetically pleasing. The colors are too bright and not useful. The background grid is too prominent. The text is displayed using three different fonts in three different sizes. (c) A bad version of part (a). Each bar is shown with its own y-axis scale. Because the scales don't align, this makes the figure misleading. One can easily get the impression that the three values are closer together than they actually are. (d) A wrong version of part (a). Without an explicit y axis scale, the numbers represented by the bars cannot be ascertained. The bars appear to be of lengths 1, 3, and 2, even though the values displayed are meant to be 3, 5, and 4.

Overall, the best thing to do is to look at your visual from a consumer's [point of view](#). You want your visuals to be intuitive enough for a viewer to be able to interpret it without too much help from you or explanatory text elsewhere in a paper or presentation. Our goal is to help consumers of our data understand the main takeaways of our research easily and accurately.

- We want to make sure our visuals always have informative labels that a lay person can understand (instead of technical variable names, we can use plain language)
  - We may need to add a legend or additional text to a visual to help with this
- We want to choose colors to convey information. We want to avoid colors that are hard to see or might distract consumers.
- The axis dimensions should not be misleading. If the goal is to compare two or more plots to each other, we would want them to have similar axes, for example.

## 4.10 Common R plotting functions and arguments

Here is a refresher of several of the functions and arguments we have come across.

Create a plot

- `plot()`: for scatterplots and trend plots
- `barplot()`: for barplot comparisons across categories
- `boxplot()`: boxplot for summaries of numeric variables
- `hist()`: for histogram summaries of a single numeric variable

Aesthetic arguments within a plot

- `main` =: Specifies the main title of the plot. Supply text (e.g., `main = "my title"`)
- `ylab` =: Specifies the title of the y-axis. Supply text (e.g., `ylab = "Mean of variable"`)
- `xlab` =: Specifies the title of the x-axis. Supply text (e.g., `xlab = "X variable name"`)
- `ylim` =: Specifies the range of the y-axis. Supply vector of two numbers (e.g., `ylim = c(0, 100)`)
- `xlim` =: Specifies the range of the x-axis. Supply vector of two numbers (e.g., `xlim = c(0, 100)`)
- `bty="n"`: Removes the border box around the plot
- `cex, cex.main, cex.names, cex.lab, cex.axis`: Changes the size of different elements of a plot. Default is 1, so a value of .8 would be smaller than default, and 1.2 would be bigger than normal.
- `type` =: Specifies the type of plot (e.g., `type="l"` is a line plot, `type="b"` is a plot with points and lines connecting them)
- `lwd`=: Specifies the width of a line on a plot. Default is 1. E.g., `lwd=3` makes a line much thicker
- `pch`=: Specifies the point type. E.g., `pch=15`
- `lty`=: Specifies the line type. E.g., `lty=2` is a dashed line
- `col`=: Specifies the color of the central element of the plot. Can take a single color or vector of colors. Use `colors()` in the console to see all R colors.
- `names`: Specifies a set of labels in a barplot

Ways to annotate a plot (generally added below the initial plotting function)

- `abline()`: Adds a line to the plot at a particular point on the x- or y- intercept, either horizontal, vertical, or of a particular slope
  - Example: Adding a horizontal line at a particular at a y value of 2 `abline(h=2)`
  - Example: Adding a vertical line at a particular at a x value of 2 `abline(v=2)`
- `lines(x=, y=)`: Adds a line connecting pairs of x- and y-coordinates. We used this to add the South line to the social mobility plot.

- `axis()`: Used to replace the default x- or y- axis that R will create with a customized axis
  - To create an original y-axis, use `axis(2, vectorofvalues, labels)` and specify `yaxt="n"` inside the plotting function to remove the original y-axis.
  - To create an original x-axis, use `axis(1, vectorofvalues, labels)` and specify `xaxt="n"` inside the plotting function to remove the original x-axis.
- `legend()`: Adds a legend to a plot. Can specify the location as the first argument (e.g., `"bottomleft"` or `"topright"`)
- `text()`: Adds text to a plot at specific x- and y- locations. (E.g., `text(x=3, y=4, "Here is a point")`). The x and y arguments can be single numbers or a vector of numbers. x and y need to be the same length.
- `points()`: Adds points to a plot at specific x- and y- locations. Inputs are much like `plot`

## 4.11 A note on `ggplot`

R has a number of open-source packages that people can use to expand the set of capabilities for visualization and analysis. These can be installed through RStudio. We will look at one of these packages: `ggplot2`.

*Using ggplot will be extra-credit at this point in the course. We may return to it later in the semester as part of the main curriculum. Reviewing this section of the notes is optional.*

The “gg” in `ggplot2` stands for the “Grammar of Graphics.” This program provides another framework for creating figures in R. According to Hadley Wickham, “`ggplot2` provides beautiful, hassle-free plots that take care of fiddly details like drawing legends.”

Practically speaking, `ggplot()` is another tool to plot the same types of figures we have been making in class. Some people prefer `ggplot2` because they find the logic of building figures more intuitive using this framework and/or more aesthetically pleasing. However, both `ggplot()` and the plots we have been making in class can accomplish the same ultimate goals of data visualization— to communicate information transparently, quickly, accurately, simply, and beautifully. Which types of plots you may prefer is up to your own taste.

Think of packages like apps on a smartphone.

- If RStudio is our smartphone, we install a package like you install an app on the phone. You only have to do this once, though occasionally you may want or need to update the installation to a new version.

```
Run this line in your R console
install.packages("ggplot2")
```

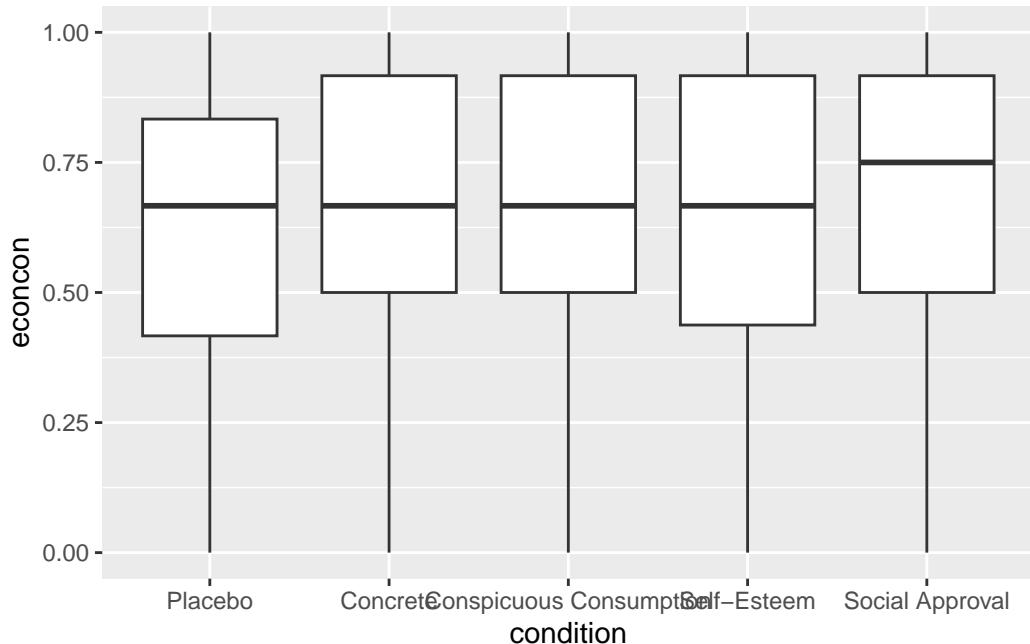
- On a smartphone, every time you want to use an app after you have installed it, you have to open the app. Similarly, every time we want to open a package in RStudio, we have to open it by using the `library()` command

```
Add and run this line in your R script, above the code where you will use functions from ggplot2
```

The main plotting function in `ggplot2` is the `ggplot()` function. It will give you access to barplots, boxplots, scatterplots, histograms, etc.

- The syntax within this package is a little different from the base R plotting functions. We will investigate below. For now, here is an example of using `ggplot` to create a boxplot using the experiment on social status from earlier in this section.

```
ggplot(data=status, mapping = aes(x=condition, y=econcon)) +
 geom_boxplot()
```



The three primary components of a `ggplot()` are a dataframe (`data =`), a set of mapping aesthetics (`aes()`), and `geoms` (e.g., `geom_boxplot`, `geom_bar`, `geom_point`, `geom_line`, etc.).

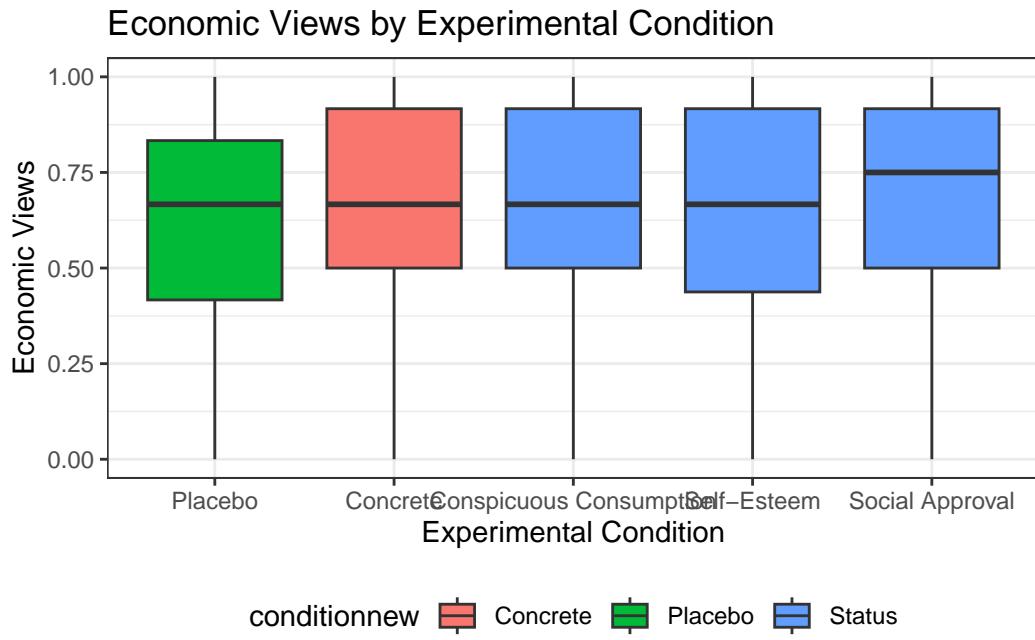
- The function `ggplot()` first takes a dataframe that includes the values you would like to plot (e.g., `data = status`).
- The aesthetics then include the variable names that you want to plot on the x and y axis (e.g., `aes(x=condition, y=econcon)`)
  - Additional mapping aesthetics can be specified. For example, a third variable (or a repeat of a previous variable) can also be specified (e.g., `fill =`, `colour =`, `shape =`), which acts as a grouping variable. If this is specified, `ggplot()` will create a corresponding legend for the plot and will color/make different shapes for different groups within this third variable (See the boxplot below for an example of grouping by condition).
- After closing out the first `ggplot()` parentheses, you then annotate the plot by adding (+) a geometric layer. This is essentially where you specify the type of plot (though it is possible to have multiple geometric layers).
- Just like with the other plotting functions in R, you can also specify a number of other arguments to make your plot more informative and aesthetically pleasing. Here, you do this by adding (+) additional arguments. See examples below (e.g., `ggtitle`, `xlab`, `ylab` for titles, `ylim` for y-axis limits, etc.)
- Likewise, just like with the other plotting functions, you can save your plots as a pdf or png. To do so here, you include the line `ggsave()` just below your plot.

There are many more possibilities for plotting with `ggplot()`, but these should get you started. For additional resources on all that is gg, I recommend the [R Graphics Cookbook](#).

Here is a second version of the boxplot with more aesthetics specified.

- We will color in the boxes based on the collapsed condition variable.

```
ggplot(data=status, mapping = aes(x=condition, y=econcon, fill=conditionnew)) +
 ## Specifies plot type. E.g., also have geom_point(), geom_bar()
 geom_boxplot()+
 ## Note many arguments are similar to other R functions but the syntax is a little different
 ggtitle("Economic Views by Experimental Condition")+
 ylab("Economic Views")+
 xlab("Experimental Condition")+
 ylim(0,1)+
 ## Changes the overall theme (i.e., color scheme, borders, etc.)
 theme_bw()+
 theme(legend.position="bottom")
```

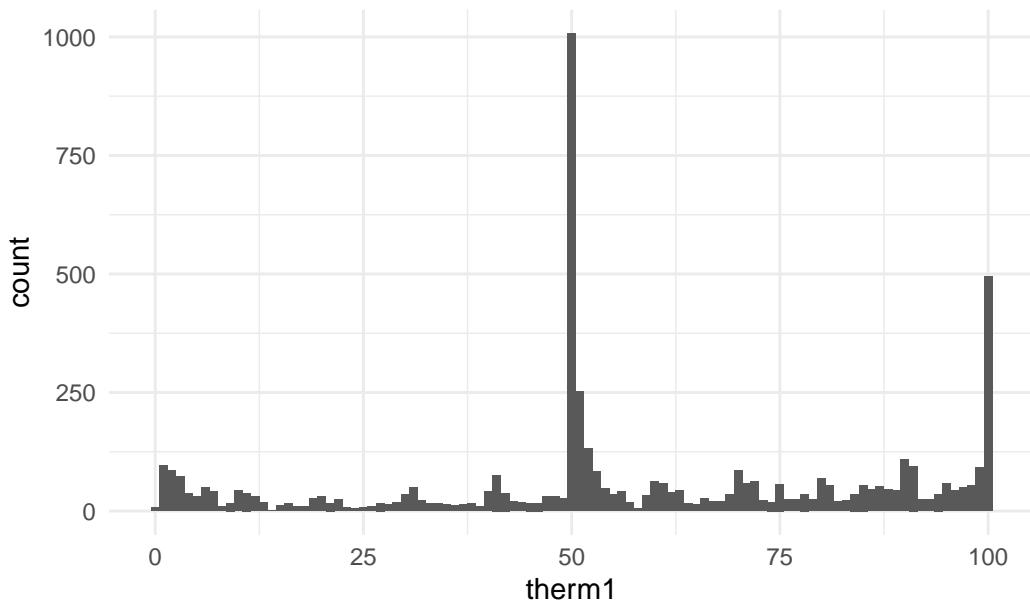


```
ggsave("myboxplot.pdf", width=7, height=5)
```

Here is an example of a histogram from the application on views toward gay couples.

```
ggplot(controlonly, aes(x=therm1)) +
 geom_histogram(binwidth = 1) +
 ggtitle("W1 Histogram") +
 theme_minimal()
```

## W1 Histogram

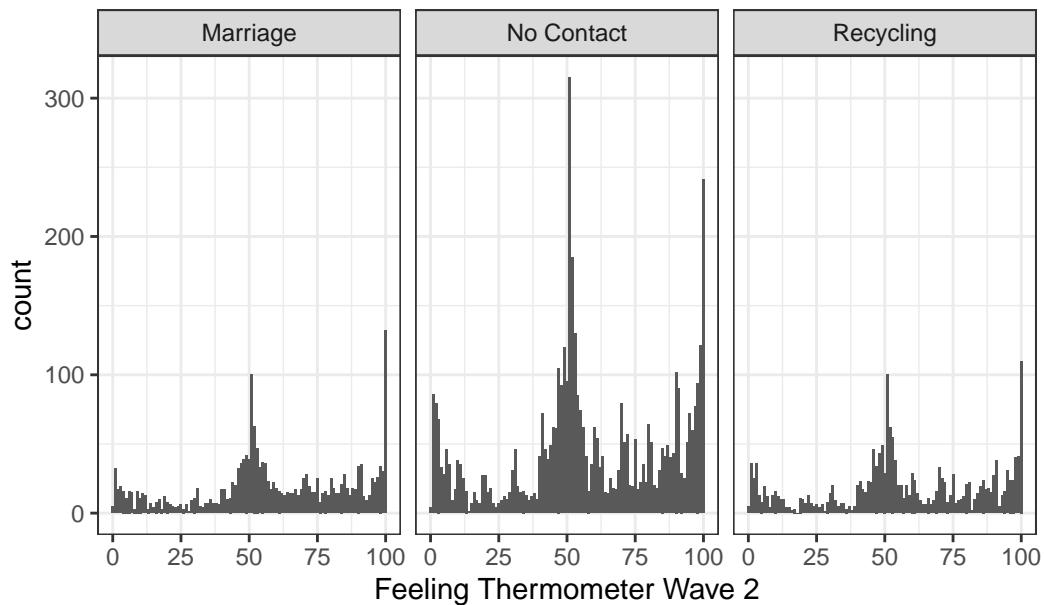


Instead of displaying multiple categories through different shapes or colors, we could also create multiple mini plots instead. This is done through `facet`. Let's look at a histogram for each condition for the thermometers in wave 2.

```
ggplot(marriage1, aes(x=therm2)) +
 geom_histogram(binwidth = 1) +
 ggtitle("W2 Histogram by Condition") +
 xlab("Feeling Thermometer Wave 2") +
 theme_bw() +
 facet_wrap(~treatmentnew)
```

Warning: Removed 1042 rows containing non-finite values (`stat\_bin()`).

## W2 Histogram by Condition

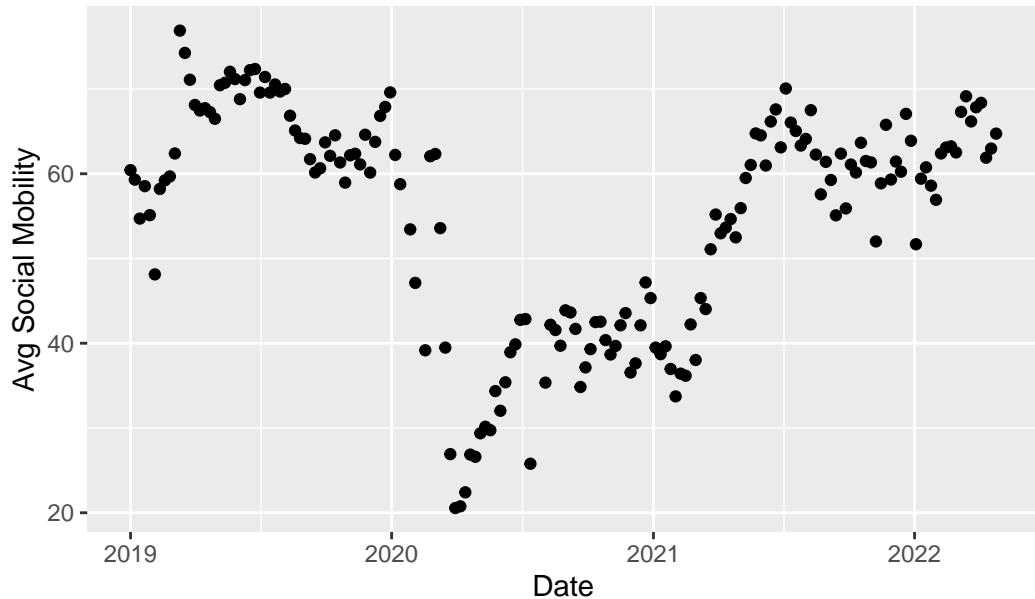


We can similarly create a scatter and line plot. Let's use the social mobility data. Here we see `geom_point` and `geom_line`.

```
Scatterplot
ggplot(covidsub, aes(x=Dates, y=avg_USA)) +
 geom_point() +
 ggtitle("Average Social Mobility in US") +
 xlab("Date")+
 ylab("Avg Social Mobility")
```

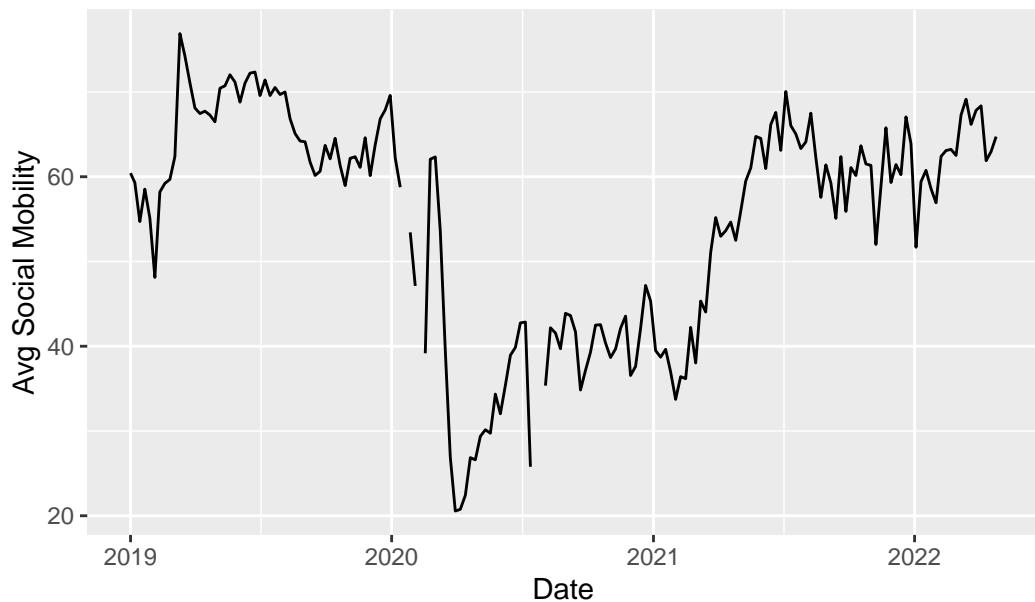
Warning: Removed 4 rows containing missing values (`geom\_point()`).

## Average Social Mobility in US



```
Line plot
ggplot(covidsub, aes(x=Dates, y=avg_USA)) +
 geom_line() +
 ggtitle("Average Social Mobility in US") +
 xlab("Date")+
 ylab("Avg Social Mobility")
```

## Average Social Mobility in US



# 5 Causality with Non-Experimental Data

In this section, we continue to evaluate causal claims, but this time we will not have the benefit of experiments.

*Recall: Why do we use experiments?*

We want to evaluate causal claims:

- Does manipulating one factor (a “treatment”) cause a change in an outcome? ( $Y_i(1) - Y_i(0)$ )
  - But we have a problem: the fundamental problem of causal inference
  - (Can’t simultaneously both be treated and untreated - e.g., you can’t simultaneously be contacted and not contacted by a campaign)
  - So instead, we randomly assign some units to receive a treatment, and some not to, and then compare their average outcomes in an experiment

And because of random assignment of the treatment, we can be confident that the groups are similar EXCEPT for the treatment

- Therefore, any difference between the two groups in average outcomes can be attributed to the treatment

*But what if we can't randomize the treatment?*

## 5.1 Why can't we always experiment?

Example: Do political leaders tend to matter for democracy?

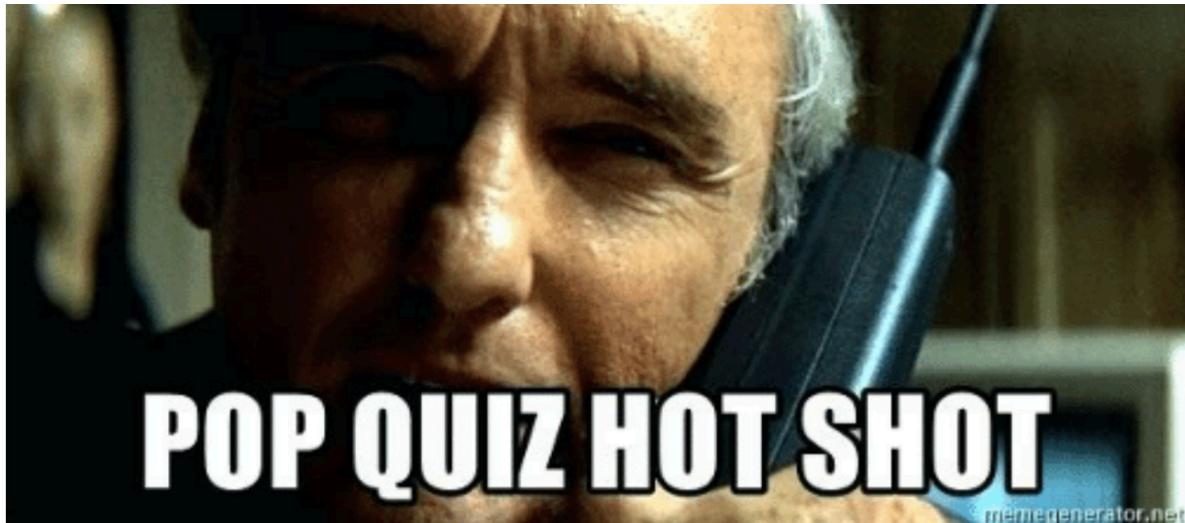
- Our outcome: how democratic nations are
- Our causal effect of interest:
  - On average, how democratic nations are with their current leaders -
  - On average, how democratic nations would be with different leaders
- Possible Experimental Designs to randomly assign half of countries to receive a different political leader
  - Rig elections (I.e., Election fraud- Illegal, unethical)

- Forcibly remove half from office (Probably illegal)
- Assassinations (Illegal, Immoral, Unethical, etc.)

Again, we have problems!!

### **5.1.1 What can we do instead?**

Let's say we want to make a causal claim about the effect of one variable on an outcome, but we can't think of an experimental design that will help us estimate this.



What do you do?

## **5.2 Causal Identification Strategies**

Our goal: Try to “identify” the causal effect of one variable on an outcome. As Montell Jordan once said, this is how we do it:

- Use data we have (that exist out in the world)
- Compare those who are “treated” to a relevant comparison group who is not treated

However, we can't randomize treatment so....

- We do our best to try to choose a good comparison (one very similar to the treatment group, but happens not to be treated)

We want to rule out all possible confounding variables and “alternative explanations” for the outcomes we observed.

### 5.2.1 Example: Travis Kelce Jersey Sale and Instagram Gains

Why did Travis Kelce experience an increase in social media followers?

- Let's use the `plot()` function to visualize this
  - Kelce saw an increase in followers on Fri, Sept 22 of 8786
  - Kelce saw an increase in followers on Sat, Sept 23 of 7242
  - Kelce saw an increase in followers on Sun, Sept 24 of 27249
- When visualizing a trend, we put time on the x-axis

```
time <- c("Fri, Sept 22", "Sat, Sept 23", "Sun, Sept 24")
```

- We put the values of the outcome on the y-axis

```
followers <- c(8786, 7242, 27249)
```

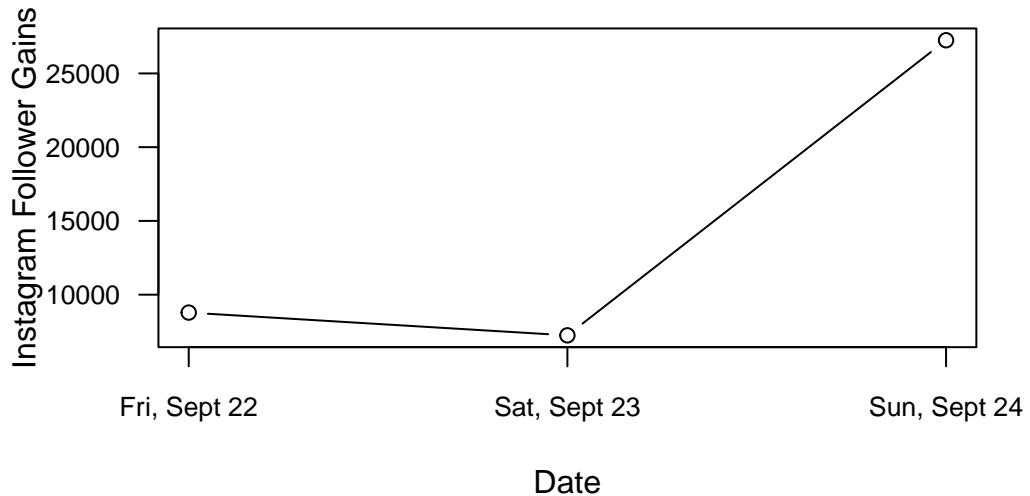
#### 5.2.1.1 Line Plots

To make it a line plot, we add `type = "l"` or `type = "b"`.

- Note: Because our time is text-based, we cannot add it to the plot directly. Instead, we use a placeholder `1:length(time)`.
- Instead, we add `xaxt="n"` to remove the default x-axis and add `axis()` below the plot code to add our own custom axis. By adding the “1”, we are indicating it should be drawn on the x-axis

```
plot(x=1:length(time), y=followers,
 type="b",
 main = "Travis Kelce Instagram Follower Gains over Time",
 ylab="Instagram Follower Gains",
 xlab="Date",
 xaxt="n",
 las=2, cex.axis=.8)
axis(1, at=1:length(time), labels=time, cex.axis=.8)
```

## Travis Kelce Instagram Follower Gains over Time



YOUR TURN: Make a causal claim about the increase in Kelce's followers

- What is the outcome? the number of followers
- What is the treatment? what do you think caused the increase
- What are the two counterfactual states of the world under treatment vs. not under treatment?

Is this a Taylor Swift effect?

- How could we prove it? What are possible confounders?
  - Maybe it's just the effect of playing a game on Sunday?
  - Maybe all NFL players experienced a similar increase?
  - Maybe Kelce had a particularly good game relative to other players?

# Taylor Swift effect: Travis Kelce jersey sales spike nearly 400%



ESPN News Services

Sep 26, 2023, 10:16 AM ET

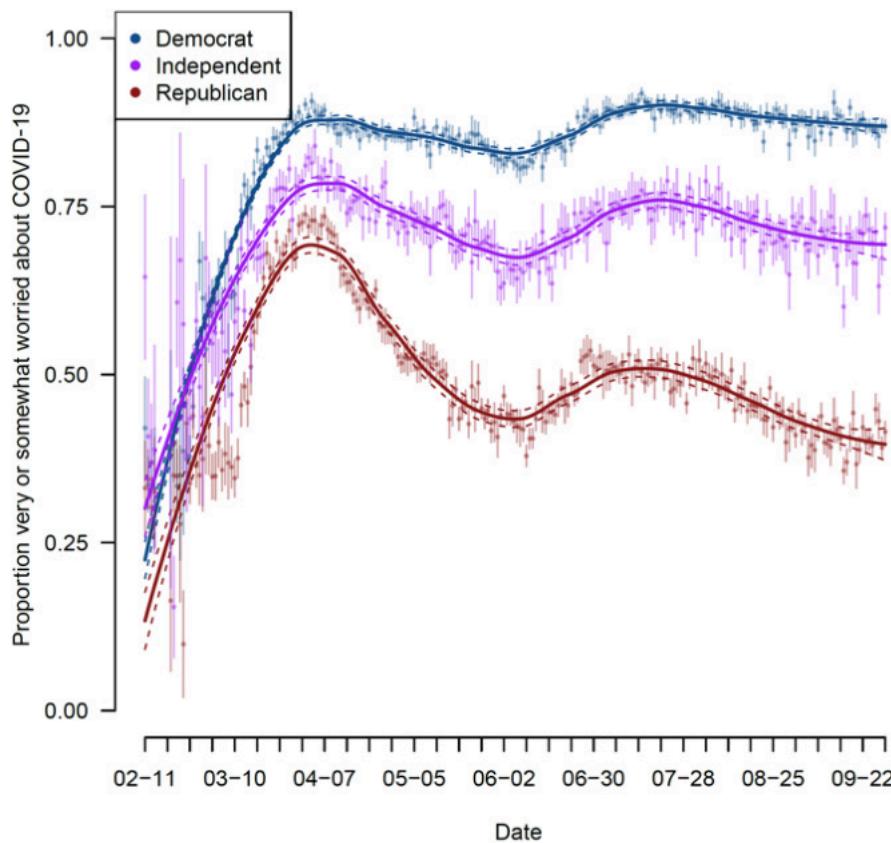
## 5.2.2 Example: Social Mobility Data

Since the onset of the pandemic in 2020, researchers have evaluated attitudinal and behavioral responses to policy changes, political messages, and COVID case/hospitalization/death rates.

- Survey data on attitudes and self-reported behavior
- Health care provider administrative data
- Mobile phone data to track locations
- Social media data to track attitudes and mobility

*Example: Using Survey data from over 1.1 million responses to measure concern about the coronavirus over time.*

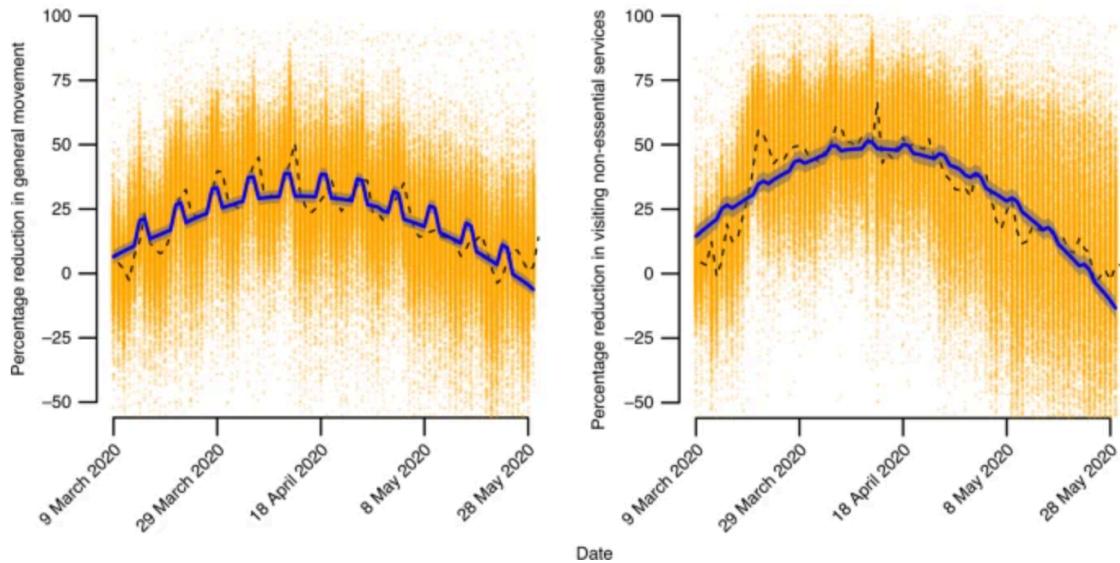
- Clinton, Joshua, et al. “[Partisan pandemic: How partisanship and public health concerns affect individuals’ social mobility during COVID-19](#).” Science advances 7.2 (2021): eabd7204.



*Example: Using the geotracking data of 15 million smartphones per day to compute percentage reduction in general movement and visiting non-essential services relative to before COVID-19 (before March 9).*

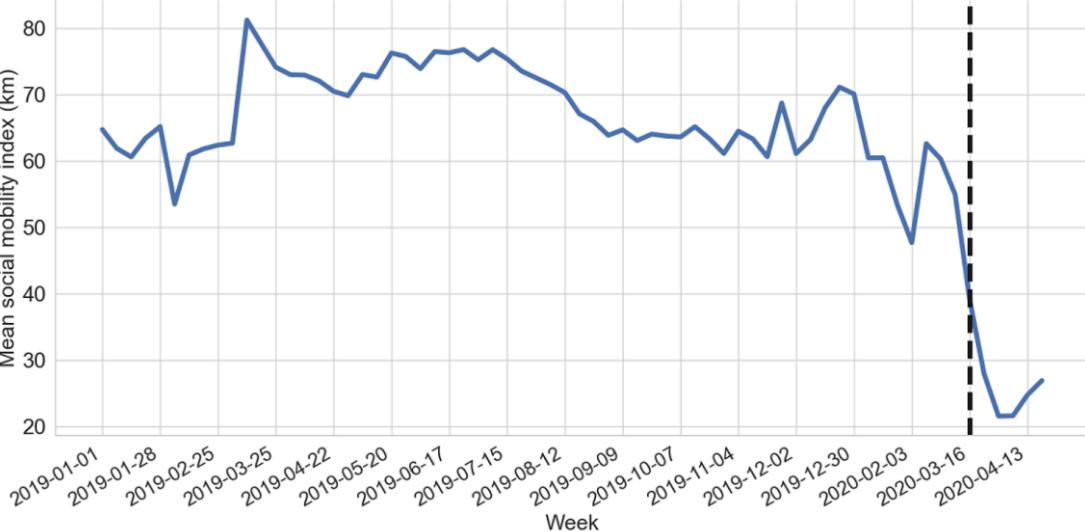
- Gollwitzer, Anton, et al. “[Partisan differences in physical distancing are linked to health outcomes during the COVID-19 pandemic](#).” Nature human behaviour 4.11 (2020): 1186-1197.

**Fig. 1: Physical distancing as a function of time (9 March to 29 May 2020).**



*Example: Using Twitter geolocation data* to track how much movement users have by looking at the distances from all locations where a given user has tweeted.

- Paiheng Xu, Mark Dredze, David A Broniatowski. “[The Twitter Social Mobility Index: Measuring Social Distancing Practices from Geolocated Tweets](#).” Journal of Medical Internet Research (JMIR), 2020.



**Figure 1.** Mean social mobility index (kilometers) in United States from January 1, 2019, to April 27, 2020. Weeks with missing data are excluded from the figure.

We will use the Twitter social mobility index to study how the movement of geo-located Twitter users changed from 2019 into April 2022.

- We will compare this movement for users located in the Northeast vs. South

Each row of the dataset represents a week of the year. Each column represents a particular geography for which social mobility was calculated by the researchers.

- **Dates** indicates the date
- **Northeast**: social mobility data for those in the northeast of the U.S.
- **South**: social mobility data for those in the south of the U.S.

```
Load the data from the author Mark Dredze's website
covid <- read.csv("https://raw.githubusercontent.com/mdredze/covid19_social_mobility.github.io/master/covid19_social_mobility.csv")
```

Just like we have encountered numeric, factor, and character variables, R also has the ability to treat variables specifically as dates. We will want R to treat the date variable we read in as a date, and not as raw text or some other variable type. To do this, we will use the `as.Date` function.

```
Date variable original format and class
head(covid$Dates)
```

```
[1] "2019-01-01" "2019-01-07" "2019-01-14" "2019-01-21" "2019-01-28"
```

```
[6] "2019-02-04"
```

```
class(covid$Dates)
```

```
[1] "character"
```

```
Convert to class Date
covid$Dates <- as.Date(covid$Date)
head(covid$Dates)
```

```
[1] "2019-01-01" "2019-01-07" "2019-01-14" "2019-01-21" "2019-01-28"
[6] "2019-02-04"
```

```
class(covid$Dates)
```

```
[1] "Date"
```

The researchers continue to add to these data. Let's look at the portion of data from 2019 to April 2022.

- Note the use of `as.Date` again to make sure R knows our text should be treated as a date
- Note the use of the greater than or equal to `>=` and less than or equal signs `<=` to specify which rows we want to keep in the data. We want rows that are in dates after January 1, 2019 and (`&`) on or before April 25, 2022.

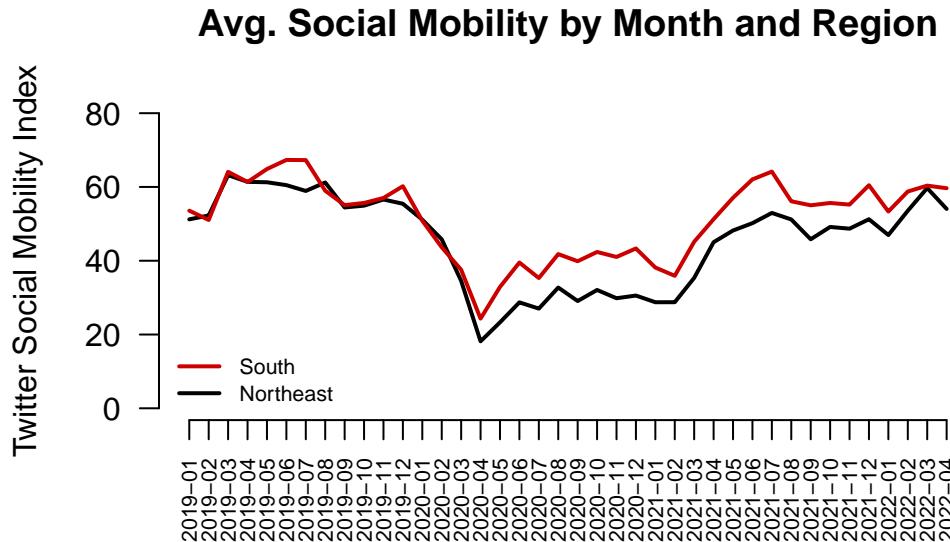
```
covidsub <- subset(covid, Dates >= as.Date("2019-01-01") &
 Dates <= as.Date("2022-04-25"))
```

These data are collected by week. That is very detailed. While that may be useful, let us create another variable that contains just the month and year, which will allow us to calculate the average per month. With a date variable, we can use the `format` function to change the format to just year and month.

```
covidsub$monthyear <- format(covidsub$Dates, "%Y-%m")
range(covidsub$monthyear)
```

```
[1] "2019-01" "2022-04"
```

Where we are going ...



Starting from the bottom ...

- Let's first create a scatterplot by providing R with our two variables
- In a trend/line plot, we want each month on the x-axis
- We want our outcome on the y-axis, in this case, average social mobility by month
- Ultimately we will want to compare the Northeast with the South. We will plot one line at a time, starting with the Northeast

We first need to find the average by month. Recall our `tapply()` function.

```
mobilitybymonthNE <- tapply(covidsub$Northeast, covidsub$monthyear, mean,
 na.rm=T)

mobilitybymonthSO <- tapply(covidsub$South, covidsub$monthyear, mean,
 na.rm=T)
```

Let's look at the output for the Northeast. Each value is what we ultimately want on the y-axis— the average social mobility in a given month.

```
mobilitybymonthNE
```

2019-01 2019-02 2019-03 2019-04 2019-05 2019-06 2019-07 2019-08

```

51.22066 52.26420 63.20130 61.38417 61.27622 60.49753 58.91779 61.20730
2019-09 2019-10 2019-11 2019-12 2020-01 2020-02 2020-03 2020-04
54.44546 54.93814 56.59830 55.44538 51.12414 45.80660 34.55917 18.15076
2020-05 2020-06 2020-07 2020-08 2020-09 2020-10 2020-11 2020-12
23.29190 28.71901 27.02149 32.73828 29.07536 32.07877 29.83641 30.56208
2021-01 2021-02 2021-03 2021-04 2021-05 2021-06 2021-07 2021-08
28.75507 28.76227 35.35340 45.02537 48.19897 50.18401 52.96105 51.19241
2021-09 2021-10 2021-11 2021-12 2022-01 2022-02 2022-03 2022-04
45.81695 49.15654 48.69051 51.24941 46.96813 53.55241 59.70933 54.04312

```

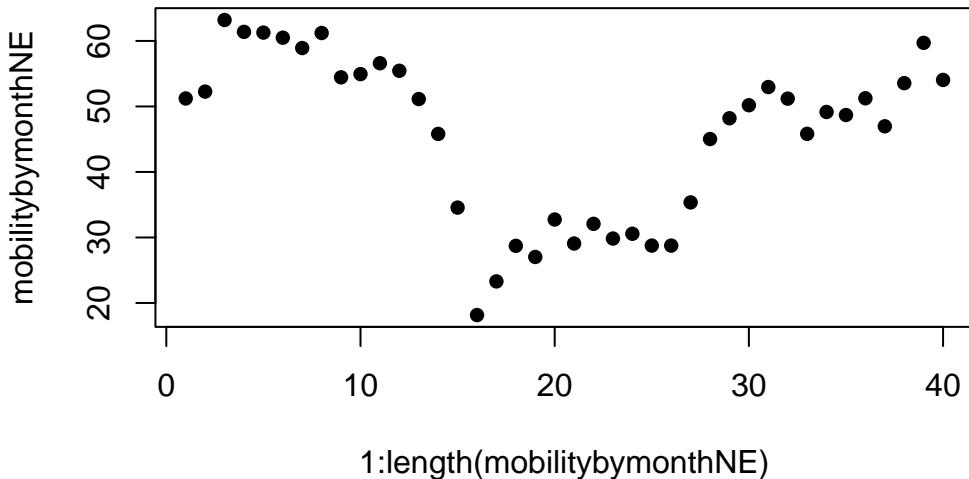
We want to plot them each at their own point on the x-axis, from the first month to the last month. We can start by creating a vector of the same length as we have months:

```
1:length(mobilitybymonthNE)
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```

These become our two inputs in the plot.

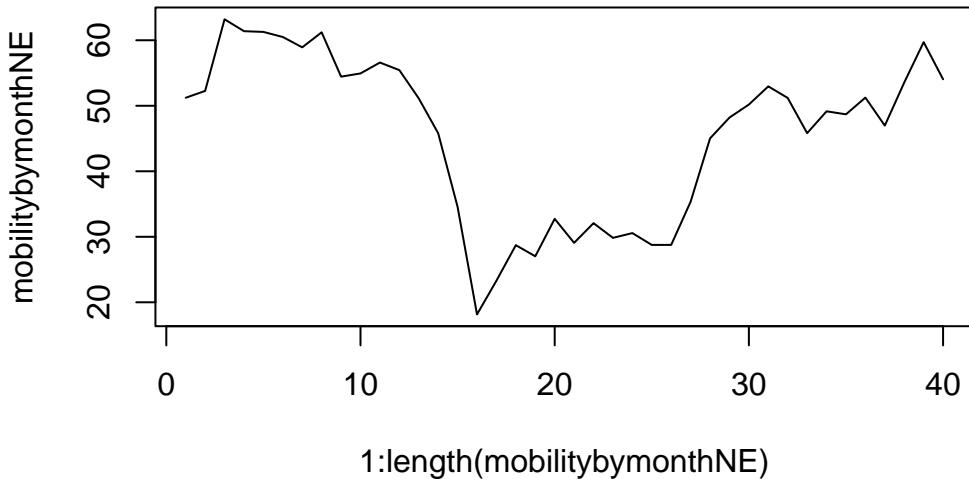
```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE, pch=16) # pch is point type
```



We now transform it to a line by specifying `type="l"`

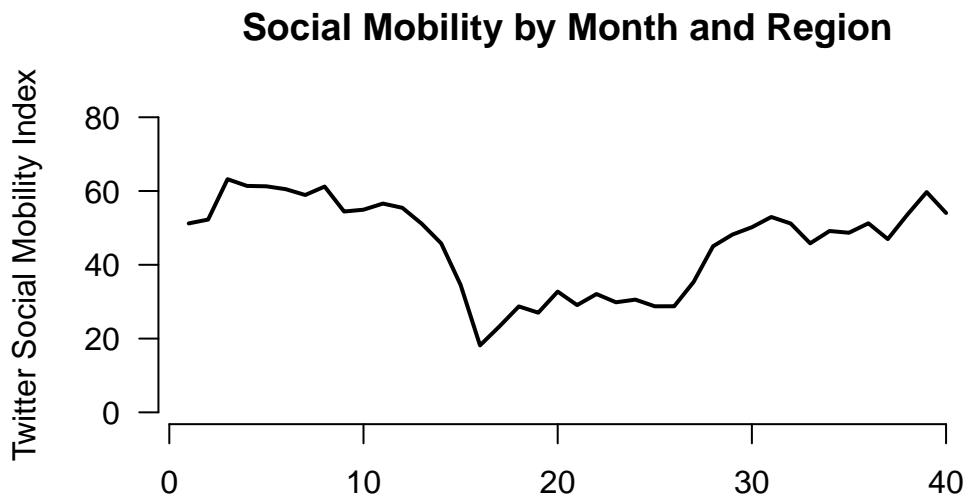
- By default, R creates a plot with `type=p` for points. R also has `type=b` which has both a line and points.

```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE, type="l") # makes it a line
```



Let us change the aesthetics a bit by adding labels and removing the border with `bty="n"`.

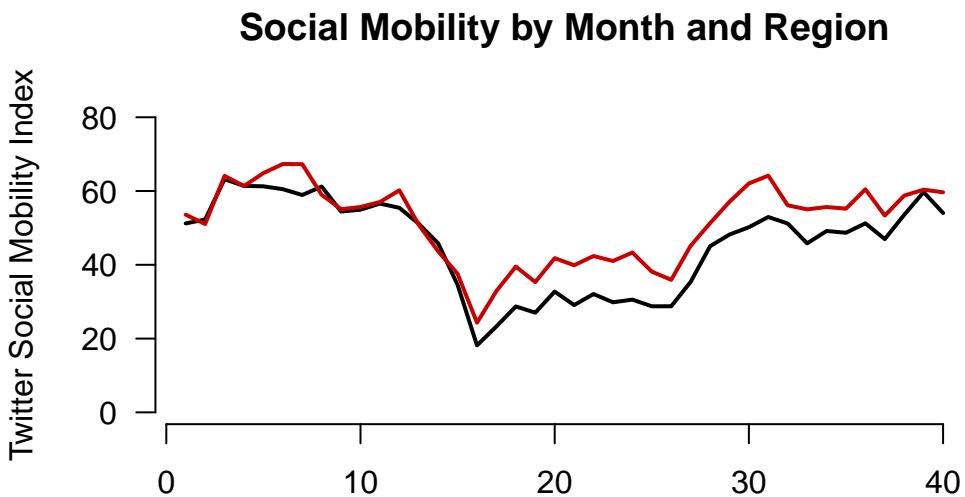
```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80), # y-axis limits
 las=1, # orientation of axis labels
 lwd=2, # line width
 bty="n") # removes border
```



Let's add a comparison line with the `lines()` function to look at trends for the south.

- Note that this is outside of the `plot()` function, but the inputs are very similar. We supply a set of x and y coordinates.

```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80), # y-axis limits
 las=1, # orientation of axis labels
 lwd=2, # line width
 bty="n") # removes border
Add line to the plot
lines(x=1:length(mobilitybymonthSO),
 y=mobilitybymonthSO, col="red3", lwd=2)
```



Let's create our own axis for the plot to add detail. To do this, we add `xaxt` to the `plot` function and then use `axis()` below the function.

The labels we will add are the actual months in the data. These happen to be the labels or `names` of our vectors:

```
names(mobilitybymonthNE)

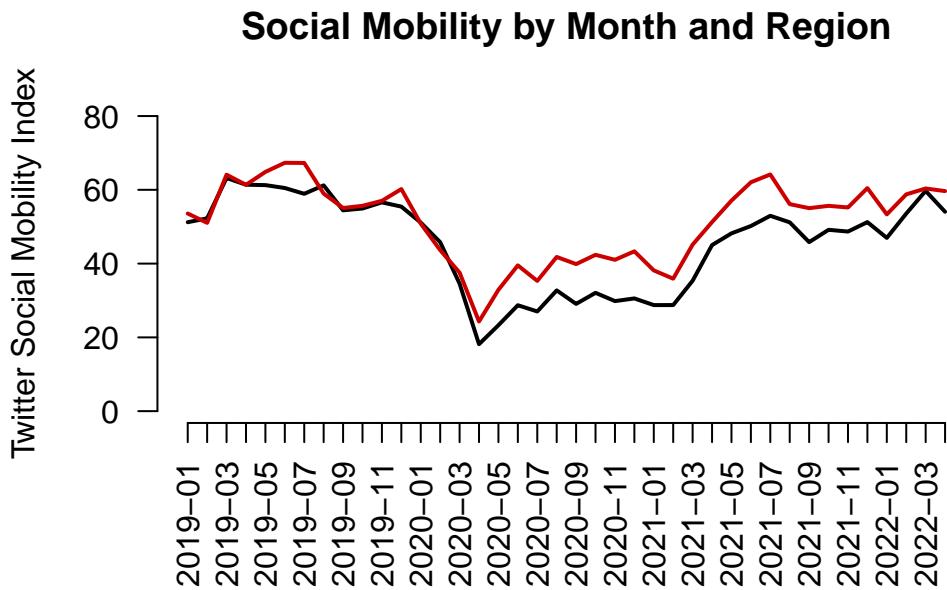
[1] "2019-01" "2019-02" "2019-03" "2019-04" "2019-05" "2019-06" "2019-07"
[8] "2019-08" "2019-09" "2019-10" "2019-11" "2019-12" "2020-01" "2020-02"
[15] "2020-03" "2020-04" "2020-05" "2020-06" "2020-07" "2020-08" "2020-09"
[22] "2020-10" "2020-11" "2020-12" "2021-01" "2021-02" "2021-03" "2021-04"
[29] "2021-05" "2021-06" "2021-07" "2021-08" "2021-09" "2021-10" "2021-11"
[36] "2021-12" "2022-01" "2022-02" "2022-03" "2022-04"

plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80),
```

```

 las=1,
 lwd=2,
 bty="n",
 xaxt="n") # removes original x-axis
Add line to the plot
lines(x=1:length(mobilitybymonthSO),
 y=mobilitybymonthSO, col="red3", lwd=2)
add the axis the "1" means x-axis. A "2" would create a y-axis
axis(1, at = 1:length(mobilitybymonthNE),
 labels=names(mobilitybymonthNE), las=2)

```



Finally, let's add a `legend()`. Now we're here!

```

plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80),
 las=1,

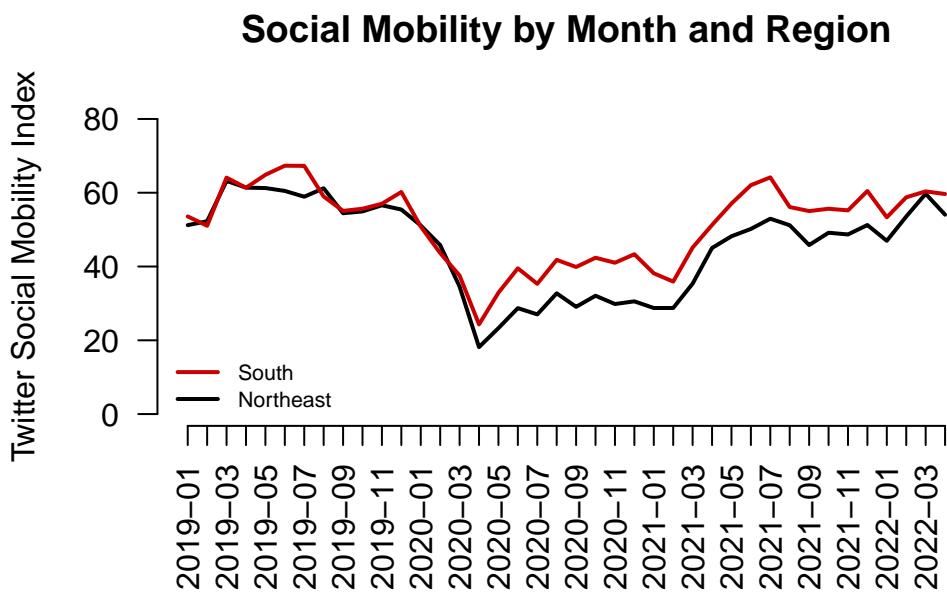
```

```

lwd=2,
bty="n",
xaxt="n") # removes original x-axis
Add line to the plot
lines(x=1:length(mobilitybymonthS0),
 y=mobilitybymonthS0, col="red3", lwd=2)
add the axis the "1" means x-axis. A "2" would create a y-axis
axis(1, at = 1:length(mobilitybymonthNE),
 labels=names(mobilitybymonthNE), las=2)

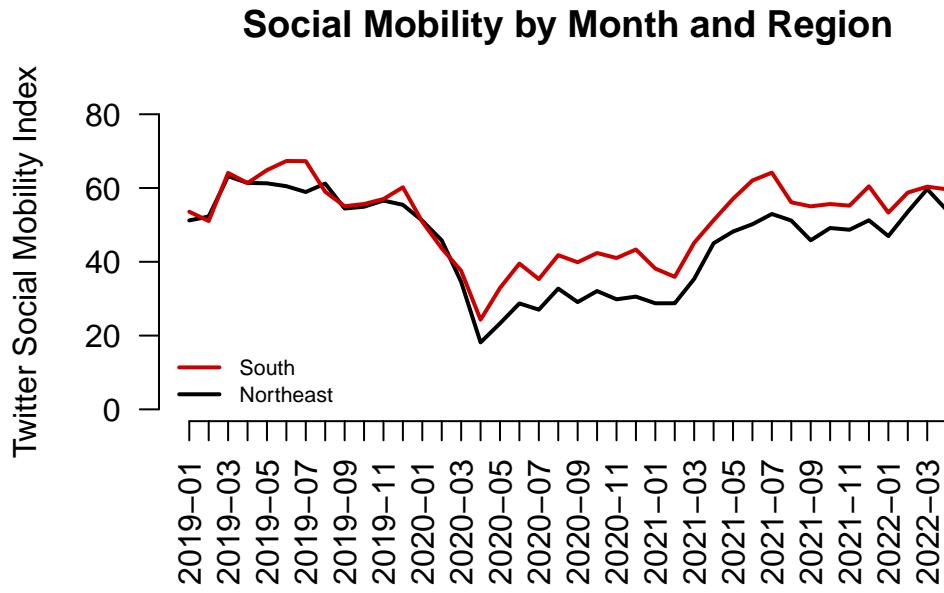
Add legend, "bottomleft" indicates where on the plot to locate it
Could use "topright" instead, for example
legend("bottomleft", col=c("red3", "black"),
 c("South", "Northeast"),
 cex = .7, # size of legend
 lwd=2,
 bty="n")

```



#### 5.2.3 Causal claims from before vs. after comparisons

*What types of research questions could these trends generate?*



What would you want to know about how movement has changed over time. Think about examples of causal claims you might make:

- Example: X caused mobility to decline
- Example: Z caused mobility to decrease
- Example: W caused mobility to increase at different rates across different regions.

So what can we do to test causal claims?

- What is the fundamental problem of causal inference in this case?
- Can we do an experiment?
- Researchers try to form comparison groups, in a strategic way, with the data they have (i.e., “observational” or “non-experimental” data).
- Because they cannot randomly assign two different experiences of the world, instead they choose two cases or two groups of cases that
  - Seem extremely similar except
  - One has the treatment of interest, and one does not

#### Example: Before vs. After Comparison

Let's examine social mobility just before vs. just after the federal announcement of social distancing guidelines to stop the spread of COVID-19.

- To do so, we will draw a vertical line at March 2020

- Note we use `abline(v=)` to indicate a vertical line at a location to cross the x-axis

This is the 15th entry in our vector, which means at point 15 on the x-axis.

```
mobilitybymonthNE["2020-03"]
```

```
2020-03
34.55917
```

```
mobilitybymonthNE[15]
```

```
2020-03
34.55917
```

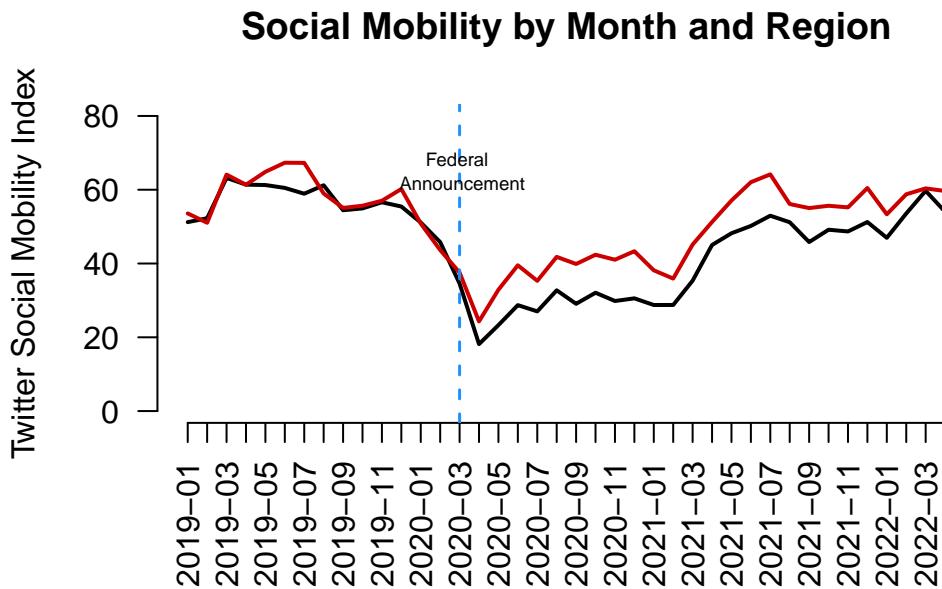
- We will also add text to inform views what that line represents
  - Note we use `text(x= , y=, labels)` to indicate where to put text

```
plot(x=1:length(mobilitybymonthNE),
 y=mobilitybymonthNE,
 type="l",
 main="Social Mobility by Month and Region",
 ylab="Twitter Social Mobility Index",
 xlab="",
 ylim = c(0, 80),
 las=1,
 lwd=2,
 bty="n",
 xaxt="n") # removes original x-axis
Add line to the plot
lines(x=1:length(mobilitybymonthS0),
 y=mobilitybymonthS0, col="red3", lwd=2)

add the axis the "1" means x-axis. A "2" would create a y-axis
axis(1, at = 1:length(mobilitybymonthNE),
 labels=names(mobilitybymonthNE), las=2)
add dashed blue vertical line
abline(v=15, lty=2, col="dodgerblue", lwd=1.5)

add text near the line
the \n breaks the text into different lines
```

```
text(x=15, y=65, labels = "Federal \n Announcement", cex=.6)
```



We see mobility does appear to be lower after the announcement relative to before the announcement. Is this causal?

- Assumption: We would want to be able to argue that social mobility in the weeks following the announcement (after time period) would look similar to social mobility in the weeks prior to the announcement (before period) **if not for the federal announcement**
  - That the before vs. after time periods would be similar in any meaningful way if not for the presence of the treatment in the after period.

Does this seem like a plausible argument? Could other things (confounders) occurring around the time of the federal announcement also have caused the steep decline in social mobility?

- If we think something else happened around the same time that might have caused mobility to go down anyway, then we may be doubtful that this is a *causal* effect.

## 5.3 Three Common Identification Strategies

Example: Does drinking Sprite make a person a better basketball player? (Inspired by 1990s commercial where a kid believes drinking Sprite will cause him to play basketball better.)

<https://www.youtube.com/watch?v=zbavu2Al-ME>

- **Cross-section comparison:** Compare Grant Hill (who drinks Sprite) to others (who don't)
- **Before-and-after:** Compare Grant Hill after he started drinking Sprite to Grant Hill before
- **Difference-in-differences:** Compare Grant Hill before and after drinking Sprite and subtract from this the difference for some other person (who never drank Sprite) during the same two periods

(Note: “drinking Sprite” is our treatment.)

### 5.3.1 Threats to Cross-Section Designs

**Assumption:** Must assume no confounders and any alternative explanations related to differences between the treated and control subjects that also relate to the outcome. The Threat: Your two groups may differ in ways beyond the “treatment” in ways that are relevant to the outcome you care about.

- Compare Grant Hill, a tall NBA player who currently drinks Sprite (treatment group) to
- Yourself, assuming you and they do not drink Sprite (control group)
- Compare your basketball skill levels (the outcome).
- Suppose Grant Hill is better (a positive treatment effect).
  - Can we conclude Sprite *causes* a person to be a better player?

Nope, because other things that affect basketball talent differ between you and Grant Hill, and these things, not Sprite, may explain the difference in basketball talent.

Moreover, even if we compared just among NBA players (Grant Hill vs. non-Sprite drinking players of his era), it's possible that Sprite targeted all-stars to recruit to drink Sprite. In this way, pre-existing basketball talent (a *confounder*) both explains why Grant Hill drank Sprite (relates to the treatment) and explains his higher level of basketball talent (relates to the outcome) in the time period after drinking Sprite.

- For a cross-sectional comparison to be plausible, we need to choose a very similar comparison in order to isolate the treatment as the main variable that is causing a change in an outcome.

### 5.3.2 Threats to Before-After Designs

*Assumption:* Must assume no confounding time trend. Threat: Something else may be changing over time, aside from the treatment, that is affecting your outcome.

- Compare Grant Hill in the years after he started drinking Sprite (treated) to
- Grant Hill the years before he started drinking Sprite (control)
- Compare his basketball skill levels (outcome).
- Suppose Grant Hill after Sprite is better (a positive treatment effect).
- Can we conclude Sprite causes a person to be a better player?

Not if something else Grant Hill started doing during that time period made him better (e.g., maybe during that time the NBA provided higher quality coaches and trainers, and everyone (including Grant Hill) got better).

- You want your treatment to be the only thing relevant to basketball talent changing over time.

### 5.3.3 Threats to Diff-in-Diff Designs

*Assumption:* Must assume parallel trends: That in the absence of treatment, your treatment group would have changed in the same way as your control

- Compare Grant Hill in the years before vs. after he started drinking Sprite to Grant Hill's teammate, who never drank sprite, in the same two time periods (before Hill drinks Sprite vs. after Hill drinks Sprite)
- Compare the **change in each player's basketball skill levels**. Suppose Grant Hill's skills increased to a greater degree than his teammate's over the same time period.
- Can we conclude Sprite causes a person to be a better player?

If we are confident that Grant Hill did not have a unique (non-Sprite) advantage over that time period relative to other players, then our assumption might be plausible— that Grant Hill and other players would have experienced a similar growth in their skills if not for Grant Hill getting the extra benefit of Sprite.

Instead, if, for example, Grant Hill got a new trainer during this period AND his teammate did not, then we might have expected Grant Hill to see more improvement even if he didn't start drinking Sprite. A violation of the parallel trends assumption!

- Causality is hard!

## 5.4 Application: Economic Effects of Basque Terrorism

Research Question: What is the economic impact of terrorism?

- Factual ( $Y(1)$ ): Economy given Basque region hit with terrorism in early 1970s
  - From 1973 to late 1990s, ETA killed almost 800 people
  - Activity localized to Basque area
- Counterfactual ( $Y(0)$ ): How would Basque economy have fared in the absence of the terrorism?
  - Basque was the 3rd richest region in Spain at onset
  - Dropped to the 6th position by late 1990s
  - Would this fall have happened in the absence of terrorism?

Problem: We can't observe the counterfactual. We can't go back in time to manipulate the experience of terrorism.

### 5.4.1 Applying 3 Identification Strategies

- Compare Basque to others after 1973 (Cross-section comparison)
- Compare Basque before and after 1973 (Before-and-after)
- Compare others before and after 1973 and subtract the difference from Basque's difference (Difference-in-differences)

For a video explainer of the code for this application, see below. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)

<https://www.youtube.com/watch?v=E4PqZgcv5IQ>

```
basque <- read.csv("basque.csv")
```

```
head(basque)
```

	region	year	gdpcap
1	Andalucia	1955	1.688732
2	Andalucia	1956	1.758498
3	Andalucia	1957	1.827621
4	Andalucia	1958	1.852756
5	Andalucia	1959	1.878035
6	Andalucia	1960	2.010140

## Variables

- `region`: 17 regions including Basque
- `year`: 1955 – 1997
- `gdpcap`: real GDP per capita (in 1986 USD, thousands)

## Subset Basque Data into Four Groups

```
Basque before terrorism
basqueBefore <- subset(basque, (year < 1973) &
 (region == "Basque Country"))

Basque after terrorism
basqueAfter <- subset(basque, (year >= 1973) &
 (region == "Basque Country"))

others before terrorism
othersBefore <- subset(basque, (year < 1973) &
 (region != "Basque Country"))

others after terrorism
othersAfter <- subset(basque, (year >= 1973) &
 (region != "Basque Country"))
```

What is the economic impact of terrorism?

Cross-section comparison

```
mean(basqueAfter$gdpcap) - mean(othersAfter$gdpcap)
```

```
[1] 1.132917
```

Before-and-after design

```
mean(basqueAfter$gdpcap) - mean(basqueBefore$gdpcap)
```

```
[1] 2.678146
```

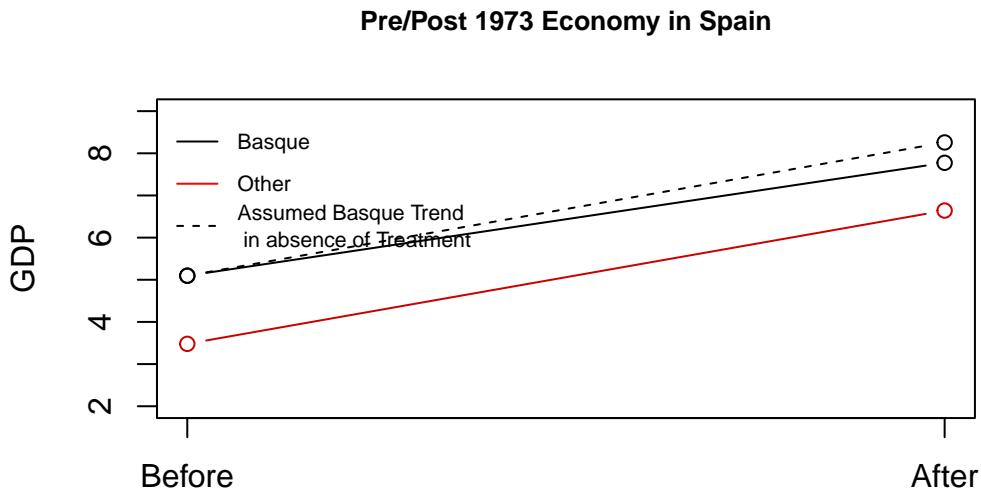
Difference-in-Differences design

```
treatDiff <- mean(basqueAfter$gdpcap) -
 mean(basqueBefore$gdpcap)
controlDiff <- mean(othersAfter$gdpcap) -
 mean(othersBefore$gdpcap)
```

```
treatDiff = controlDiff
```

```
[1] -0.48316
```

Here is a way to visualize this difference-in-differences. Our estimated causal effect is the difference between the observed growth in the Basque region and what we assume the growth would have been in the absence of terrorism (the treatment).



What should we conclude from each approach?

- Each approach resulted in a different estimate of the impact of terrorism on the economy. We should choose the approach for which we think the underlying assumptions are most plausible.

## 5.5 Placebo Tests

Which Results Should We Believe? Role of Placebo Tests

Cross-section comparison

```
were there pre-existing differences between the groups?
mean(basqueBefore$gdpcap) - mean(othersBefore$gdpcap)
```

```
[1] 1.616077
```

Before-and-After design

```
was there a change in a group we don't think should have changed?
mean(othersAfter$gdpcap) - mean(othersBefore$gdpcap)
```

```
[1] 3.161306
```

What about the Difference-in-Differences design?

```
here we go back in time even further to examine "pre-treatment" trends
we want them to be similar
(basqueBefore$gdpcap[basqueBefore$year == 1972] -
 basqueBefore$gdpcap[basqueBefore$year == 1955]) -
 (mean(othersBefore$gdpcap[othersBefore$year == 1972]) -
 mean(othersBefore$gdpcap[othersBefore$year == 1955]))
```

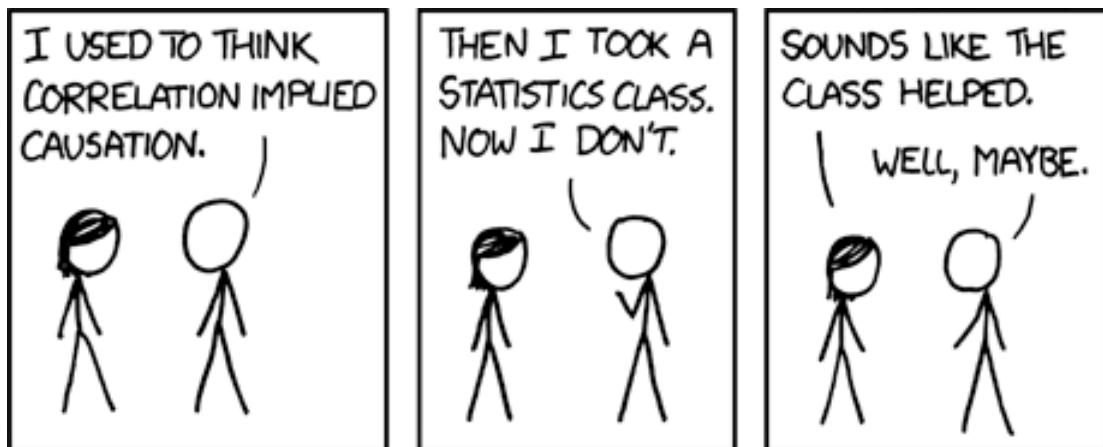
```
[1] 0.07147071
```

These “placebo” checks are closest to zero for diff-in-diff, so we may believe that the most.

*Thanks to Will Lowe and QSS for providing the foundations for this example*

## 5.6 Wrapping Up Causality

Do you get this joke?



# 6 Loops in R

In this brief section, we will go over conducting loops in R.

Loops are a tool in R that are useful for situations where we want to do something over and over and over and ... over again, where we just change something small each time.

A quick example using the Basque data from the previous section:

```
basque <- read.csv("basque.csv", stringsAsFactors = T)
```

Let's say I wanted to know the GDP for each region for the earliest year they are in the data.

```
regionsubset <- subset(basque, region == "Andalucia")
regionsubset$gdpcap[regionsubset$year == min(regionsubset$year)]
```

```
[1] 1.688732
```

```
Repeat for a new region
```

```
regionsubset <- subset(basque, region == "Aragon")
regionsubset$gdpcap[regionsubset$year == min(regionsubset$year)]
```

```
[1] 2.288775
```

```
Repeat for a new region
```

```
regionsubset <- subset(basque, region == "Baleares")
regionsubset$gdpcap[regionsubset$year == min(regionsubset$year)]
```

```
[1] 3.143959
```

Ughhh can we automate this? we have 17 regions!!!

```
unique(basque$region)
```

```

[1] Andalucia Aragon Principado De Asturias
[4] Baleares Canarias Cantabria
[7] Castilla Y Leon Castilla-La Mancha Cataluna
[10] Comunidad Valenciana Extremadura Galicia
[13] Madrid Murcia Navarra
[16] Basque Country Rioja Rioja
17 Levels: Andalucia Aragon Baleares Basque Country Canarias ... Rioja

```

Where we will be going by the end of this section:

```

gdpmoneyear <- rep(NA, 17) # empty "container" vector
regions <- unique(basque$region) # what we iterate through
names(gdpmoneyear) <- unique(basque$region) # labels for our output

for(i in 1:17){
 regionsubset <- subset(basque, region == regions[i])
 gdpmoneyear[i] <- regionsubset$gdpcap[regionsubset$year ==
 min(regionsubset$year)]
}
head(gdpmoneyear) # output

```

Andalucia	Aragon	Principado De Asturias
1.688732	2.288775	2.502928
Baleares	Canarias	Cantabria
3.143959	1.914382	2.559412

We got all of the answers with just one chunk of code!

## 6.1 The anatomy of a loop

*A short video introduction to the anatomy of a loop* <https://www.youtube.com/watch?v=RDsnhVpM1To>

In many situations, we want to repeat the same calculations with different inputs. Loops allow you to avoid writing many similar code chunks.

- The function `for(i in X){}` will create a loop in your programming code where `i` is a counter and
- `X` is a placeholder for a vector for the possible values of the counter.

We use the following syntax:

```
for (i in X) {
 command1...
 command2...
 ...
}
```

to indicate we want to repeat command1 and command2 and .... as many commands as we want, for each *i* in the set of possible values for *i* stored in *X*.

### 6.1.1 The key parts of a loop

The meat: the command or set of commands you want to do over and over.

```
the meat
result <- 6 + 2
result <- 8 + 2
result <- 4 + 2
result <- 7 + 2
result <- 11 + 2
```

Note the pattern: we take some number and + 2 each time.

- It is the number that is changing -> what we will iterate.

For a loop, you want to:

1. The Meat: Write down the code for one version.

```
result <- 6 + 2
```

2. The Bread: Embed this code in the loop syntax (`for(i in X){}`)

```
for(i in X){
 result <- 6 + 2
}
```

3. Create a vector that contains the values you want to loop through

```
somenumbers <- c(6, 8, 4, 7, 11)
```

4. Create a storage vector that will contain the results

```
result <- rep(NA, length(somenumbers))
```

5. Modify the meat and bread to iterate by using [i], and replace X.

```
for(i in 1:length(somenumbers)){
 result[i] <- somenumbers[i] + 2
}
```

where `1:length(somenumbers)` reflects possible values `i` will take

```
1:length(somenumbers)
```

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

### 6.1.2 A short example

Let's put these parts together:

Suppose we want to add 2 to a set of numbers `c(6, 8, 4, 7, 11)`

```
somenumbers <- c(6, 8, 4, 7, 11) # iteration vector
result <- rep(NA, length(somenumbers)) # container vector

for(i in 1:length(somenumbers)){
 result[i] <- somenumbers[i] + 2
}
result
```

```
[1] 8 10 6 9 13
```

How does this work? Every iteration, the value of i changes.

- For example, when `i` is 1, we take the first value in our `somenumbers` vector `somenumbers[1]`, add 2 to it, and store it in the first position of our container vector `result[1]`. When `i` is 2, we switch the number in the brackets to 2, corresponding to the second entry in each vector, and so on.

```
Suppose i is 1
result[1] <- somenumbers[1] + 2
result[1]
```

```
[1] 8
```

```
Suppose i is 2
result[2] <- somenumbers[2] + 2
result[2]
```

```
[1] 10
```

```
Suppose i is 3
result[3] <- somenumbers[3] + 2
result[3]
```

```
[1] 6
```

### 6.1.3 Troubleshooting a loop

The inside part of the loop should run if we set `i` to a particular value.

```
i <- 1
result[i] <- somenumbers[i] + 2
```

If you get an error here, there is something wrong with the meat! (and not necessarily the loop)

```
result[i]
```

```
[1] 8
```

For example, if we had a typo, we'd get an error. Try running the below!

```
i <- 1
result[i] <- somenumberz[i] + 2
```

#### 6.1.4 Your turn

Using a loop, for each value in our poll results, add 10 and divide by 100. Store in a vector called `adjustedpollresults`.

```
pollresults <- c(70, 40, 45, 60, 43, 80, 23)
```

Remember the steps:

1. The Meat: Write down the code for one version.
2. The Bread: Embed this code in the loop syntax (`for(i in X){}`)
3. Create a vector that contains the values you want to loop through (here it's `pollresults`)
4. Create a storage vector that will contain the results (here it's `adjustedpollresults`)
5. Modify the meat and bread to iterate by using `[i]` and replace `X`.

Try on your own, then expand for the solution.

```
pollresults <- c(70, 40, 45, 60, 43, 80, 23)
adjustedpollresults <- rep(NA, length(pollresults))

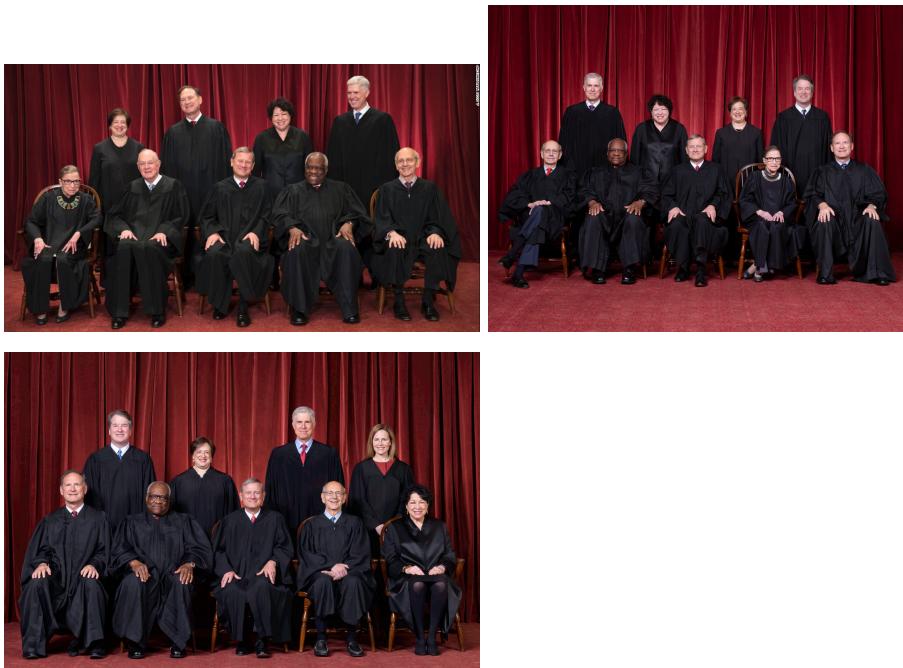
for(i in 1:length(pollresults)){
 adjustedpollresults[i] <- (pollresults[i] + 10)/100
}
adjustedpollresults
```

```
[1] 0.80 0.50 0.55 0.70 0.53 0.90 0.33
```

## 6.2 Application: U.S. Supreme Court

*A video explainer of the loop in this section using a similar dataset that goes through 2020. We now have data going through 2021! [https://www.youtube.com/watch?v=Ij9YWgUQg\\_s](https://www.youtube.com/watch?v=Ij9YWgUQg_s)*

The Court has changed a lot recently in its composition (and will continue to do so next term).



Ideology on the U.S. Supreme Court: With Kennedy out, Kavanaugh in, did the Court have a Conservative shift? What about with the death of Ruth Bader Ginsburg and confirmation of Amy Coney Barrett?

Many people predicted it would. See this [FiveThirtyEight article](#) as an example. The graph from the article shows Kavanaugh's predicted ideology.

OCT. 6, 2018, AT 4:01 PM

## How Kavanaugh Will Change The Supreme Court

By [Oliver Roeder](#)

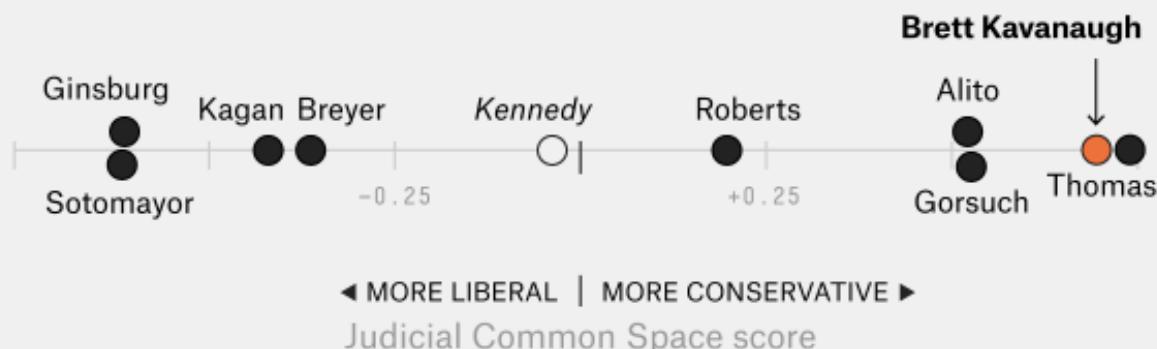
Filed under [Supreme Court](#)



Brett Kavanaugh before the Senate Judiciary Committee on Sept. 6. DREW ANGERER / GETTY IMAGES

### How Kavanaugh compares

The ideologies of the current justices on the Supreme Court — and that of President Trump's nominee



Supreme Court justice scores are from the 2016 term. Kavanaugh's score is based on his nomination process and not his votes.

FiveThirtyEight

SOURCES: LEE EPSTEIN, CHAD WESTERLAND, THE JOURNAL OF LAW, ECONOMICS, AND ORGANIZATION

We will explore how the Court has changed ideologically, with a focus on how the location of

the median U.S. Supreme Court Justice shifted over time.

Why does the median matter? A refresher on the Court

- President nominates the justice. Senate must confirm.
- Justices serve lifetime appointments.
- Trump nominated Gorsuch, following Scalia death, confirmed 2017.
- Trump nominated Kavanaugh, following Kennedy retirement, confirmed 2018.
- The Court typically has 9 justices, so whichever justice is the median in terms of ideology, can act as the “swing” vote in cases where the Court is divided
  - Anthony Kennedy was often the “swing” justice for a decade.
- With Kennedy out, the prediction was that the Court would return to similar balance as when O’Connor was the median.
- In 2020, Ruth Bader Ginsburg died and Amy Coney Barrett was confirmed to Court late that year, likely shifting the Court again.
- We can continue to update the data to examine what happened. We don’t yet have data on changes to the Court that have come about since Justice Ketanji Brown Jackson was confirmed following the retirement of Justice Breyer.

Let’s load and explore our data.

- `term`: is the year of the SC (1991-2021 except for 2005),
- `justiceName`: contains the name of the Justice, and
- `post_mn`: includes the “ideal point”— this is the estimated ideology

Martin-Quinn Scores assess ideology based on how judges “cluster” together in their voting patterns. Every Justice gets an ideology score, and this score can change each SC term (year) they are on the Court. Higher scores are more conservative justices, and lower, more liberal. More information is available at the [MQScores website](#)

```
justices <- read.csv("justices23.csv", stringsAsFactors = T)

alternative
justices <- read.csv("https://raw.githubusercontent.com/ktmccabe/teachingdata/main/justice")
```

We are going to make the name variable a character class. This will make R treat the names as raw text rather than valued categories. This will be useful later on in the application.

```
justice Name as character
justices$justiceName <- as.character(justices$justiceName)
```

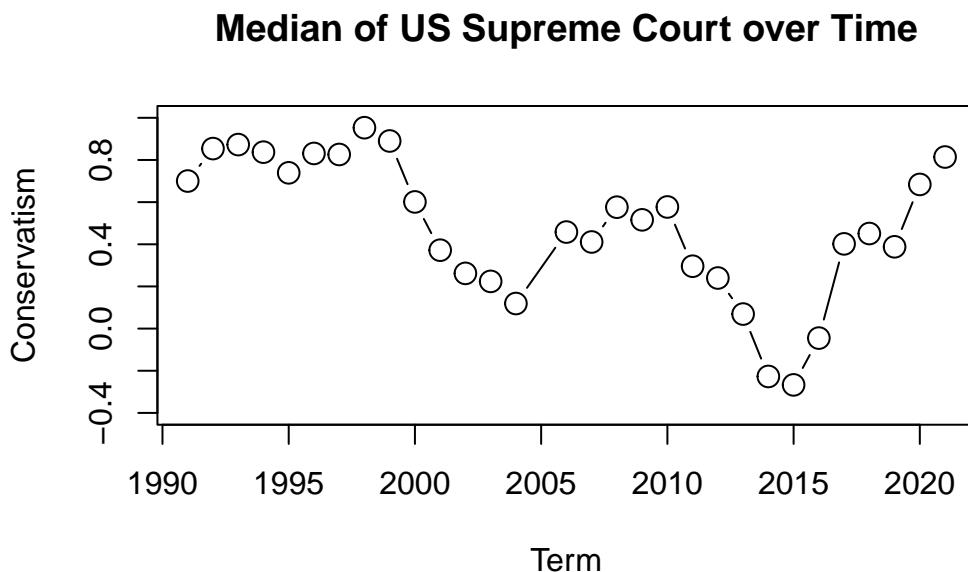
We can use `tapply()` to see the median “ideal point” (ideology score) each term in our data.

```

Note: we use tapply like before but replace mean with median
medians <- tapply(justices$post_mn, justices$term, median)

plot(x = names(medians),
 y= medians,
 ylim = c(-.4, 1),
 type = "b",
 cex=1.5,
 ylab="Conservatism",
 xlab="Term",
 main="Median of US Supreme Court over Time")

```



We see a conservative shift at the end of the plot. However, we cannot tell whether this represents a shift within a particular justice's ideology or a shift in which justice has become the median, due perhaps, to the change in the Court's composition.

***We need to find which justice is the median!***

Loops to the rescue!

We will start our process by defining the meat of the operation.

- We want to find the median SC Justice for each term. To get started, let's pretend we only have to find the median Supreme Court Justice for one term.

```
SCterms <- sort(unique(justices$term))
SCterms

[1] 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2006
[16] 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021
```

Note that where you have a vector where some entries in the vector are repeated (such as terms on the Supreme Court), you can extract the unique elements of that vector using the `unique()` function. You can also `sort()` them in numeric or alphabetical order. This won't be necessary most times.

First, let's think about how we would do this for just one of the Supreme Court terms. Well we would first subset our data frame to contain only that one Supreme Court term.

```
Example for the first term
SCterms[1]
```

```
[1] 1991
```

```
Subset data to include only rows from 1991
subterm <- subset(justices, term == 1991)
```

Then, we would take the median of these ideal points

```
median.ip <- median(subterm$post_mn)
```

Finally, we would figure out which justice has this median.

```
result <- subterm$justiceName[subterm$post_mn == median.ip]
result
```

```
[1] "SDOConnor"
```

Now let's put it into our loop syntax

```
for(i in . . .){
subterm <- subset(justices, term == 1991)
median.ip <- median(subterm$post_mn)
result <- subterm$justiceName[subterm$post_mn == median.ip]
```

```
#}
```

Now, we need our container vector and iteration vectors.

```
SCterms <- sort(unique(justices$term))
results <- rep(NA, length(SCterms))
names(results) <- SCterms
```

Finally, we would modify our loop syntax with `i` and `[i]`

```
for(i in 1:length(SCterms)){
 subterm <- subset(justices, term == SCterms[i])
 median.ip <- median(subterm$post_mn)
 results[i] <- subterm$justiceName[subterm$post_mn == median.ip]
}
```

Did it work?

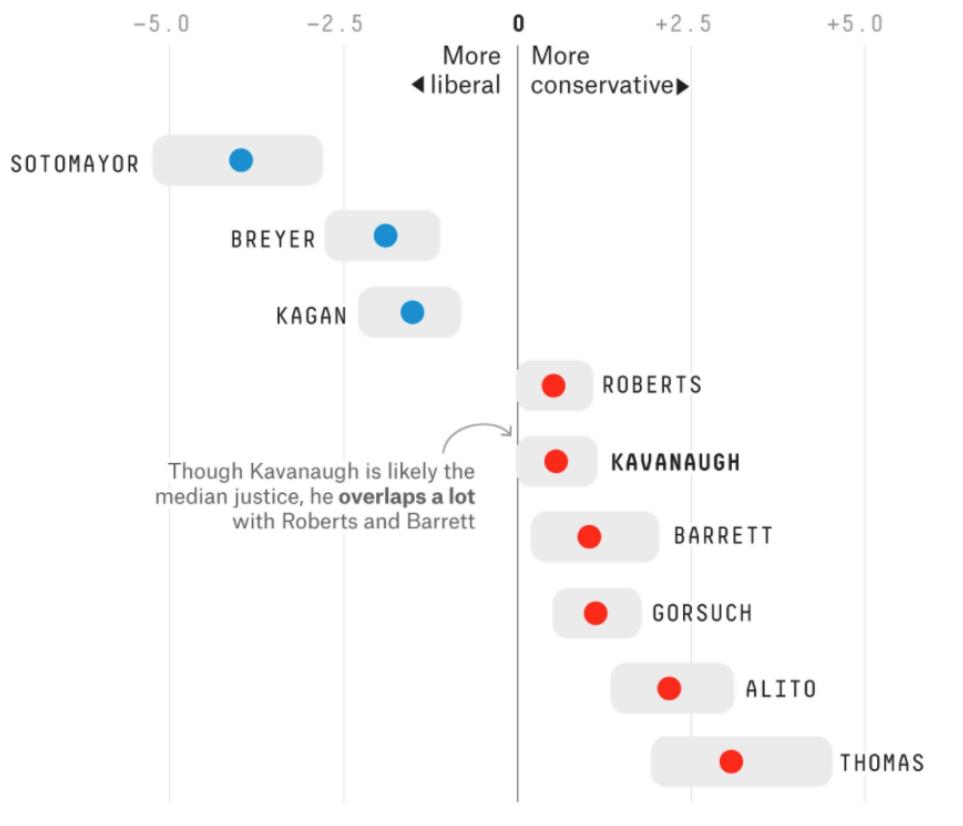
```
results
```

1991	1992	1993	1994	1995
"SDOConnor"	"SDOConnor"	"AMKennedy"	"SDOConnor"	"AMKennedy"
1996	1997	1998	1999	2000
"AMKennedy"	"AMKennedy"	"AMKennedy"	"SDOConnor"	"SDOConnor"
2001	2002	2003	2004	2006
"SDOConnor"	"SDOConnor"	"SDOConnor"	"SDOConnor"	"AMKennedy"
2007	2008	2009	2010	2011
"AMKennedy"	"AMKennedy"	"AMKennedy"	"AMKennedy"	"AMKennedy"
2012	2013	2014	2015	2016
"AMKennedy"	"AMKennedy"	"AMKennedy"	"AMKennedy"	"AMKennedy"
2017	2018	2019	2020	2021
"AMKennedy"	"JGRoberts"	"JGRoberts"	"BMKavanaugh"	"BMKavanaugh"

Our evidence aligns with others:

## Kavanaugh was likely the median justice

Estimated ideologies of Supreme Court justices in the October 2020 term



FiveThirtyEight

SOURCE: MARTIN-QUINN SCORES

### 6.2.1 Troubleshooting the loop

Recall, the inside part of the loop should run if we set *i* to a particular value.

```
i <- 1
subterm <- subset(justices, term == SCterms[i])
median.ip <- median(subterm$post_mn)
results[i] <- subterm$justiceName[subterm$post_mn == median.ip]

results[i]
```

```
1991
"SDOConnor"
```

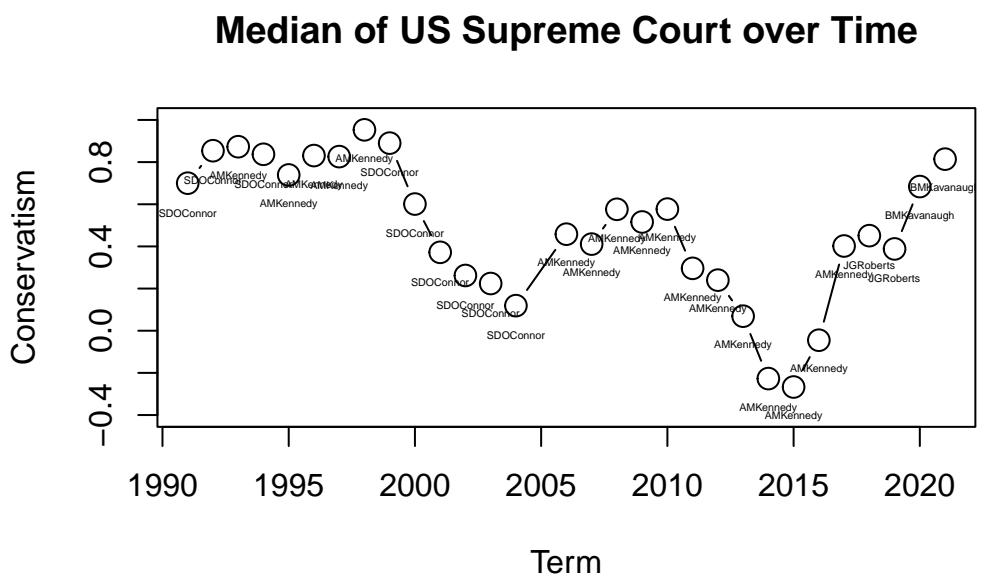
We are in good shape! If we had a typo, we'd get an error message there, and that would be a sign that we need to work on the inside part of the code before putting it back into the loop structure.

### 6.2.2 Visualizing the Results

To get a bit more practice with plots, let's visualize the results and make our interpretations.

```
medians <- tapply(justices$post_mn, justices$term, median)
plot(x =names(medians),
 y= medians,
 ylim = c(-.4, 1),
 type = "b",
 cex=1.5,
 ylab="Conservatism",
 xlab="Term",
 main="Median of US Supreme Court over Time")

Add the names to the plot
Note: we want to make sure medians and results are in the same order for this to work
text(x=names(results), y=(medians - .14), labels=results, cex=.35)
```



We have now used the `text()` function. Similar to plot, the `text()` takes a set of x and y coordinates that tells R the location of where you want to add a piece(s) of text to the plot. The third input is the actual text.

Why did the Court shift more conservative at the end of the time trend?

- Well we see that Justice Roberts and then Justice Kavanaugh became the median!

## The Supreme Court's Conservative Supermajority Is Just Beginning To Flex Its Muscles

By [Laura Bronner](#) and [Elena Mejia](#)  
Filed under [Supreme Court](#)  
Published Jul. 2, 2021

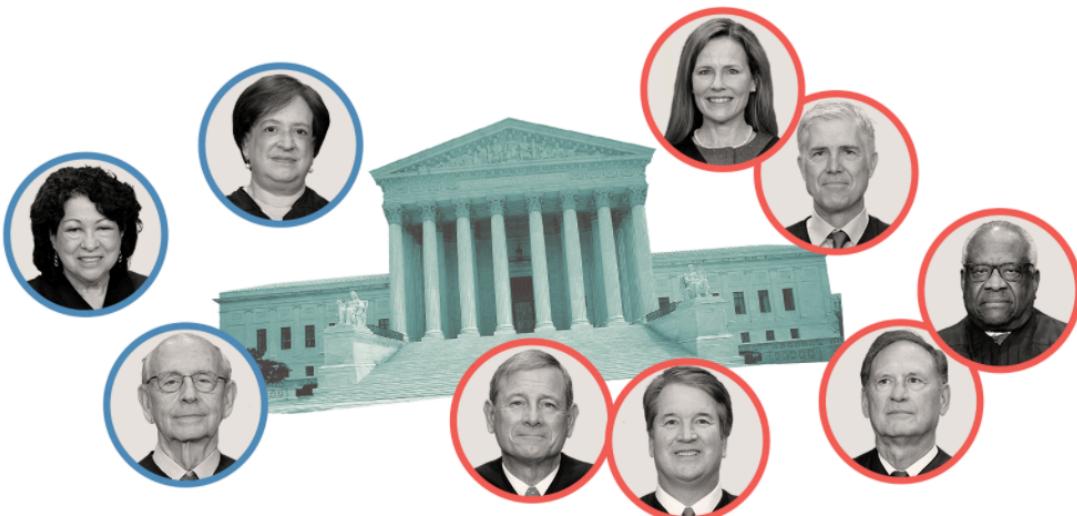
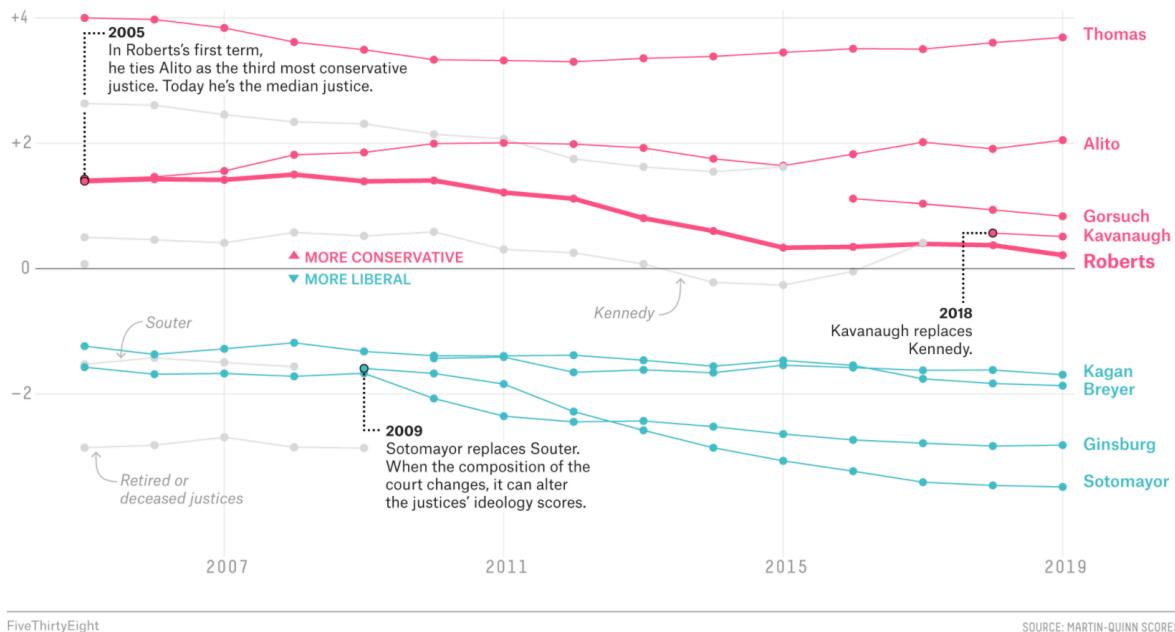


PHOTO ILLUSTRATION BY FIVETHIRTYEIGHT / GETTY IMAGES

Figure 6.1: FiveThirtyEight

As [FiveThirtyEight](#) notes, just because Justice Roberts is the new median, does not mean he has become more liberal. The Court composition is shifting, and the MQ scores also depend on the issues being heard before the Court.

**Chief Justice Roberts is increasingly pivotal to the court**  
 Estimated ideologies of Supreme Court justices since Roberts joined the court in 2005



FiveThirtyEight

SOURCE: MARTIN-QUINN SCORES

Recall, the Martin-Quinn scores measure justice ideology based on voting patterns. What are the strengths and weaknesses of using this type of information to score the ideology of a justice?

### 6.2.3 Enhancing the plot

Let's make the plot more beautiful by color coding.

```
medians <- tapply(justices$post_mn, justices$term, median)
plot(x =names(medians),
 y= medians,
 ylim = c(-.4, 1),
 type = "b",
 ylab="Conservatism",
 xlab="Term",
 main="Median of US Supreme Court over Time",
 xaxt="n", ## removes the x-axis
 las=1)

Adds text
```

```

text(x=names(results), y=(medians - .14), results, cex=.35)

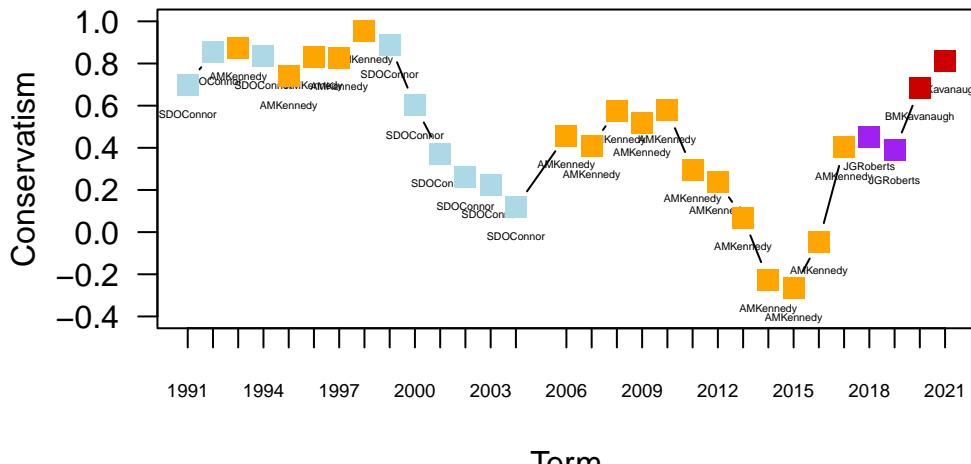
Adds color-coded points on top of existing points
points(x =names(medians),
 y= medians,

Adds colors according to how results is coded
col= ifelse(results == "AMKennedy", "orange",
 ifelse(results == "SDOConnor", "light blue",
 ifelse(results == "JGRoberts", "purple",
 "red3"))),
 pch=15, # point type- squares
 cex=1.5) # size of points

Adds custom x-axis at the specific years included in names(medians)
axis(1, names(medians), cex.axis=.6)

```

## Median of US Supreme Court over Time



We have used the `points()` function. This adds an additional layer of points to a plot. It works much like the `plot` function in that it takes a set of x and y coordinates.

We could change the look of the plot even more by adding a legend and altering the borders and look of the plot.

```

medians <- tapply(justices$post_mn, justices$term, median)
plot(x =names(medians),
 y= medians,
 ylim = c(-.4, 1),
 type = "b",
 ylab="Conservatism",
 xlab="Term",
 main="Median of US Supreme Court over Time",
 xaxt="n", # removes x axis
 las=1, # changes the orientation of the axis labels
 lwd=2, # increases the thickness of the lines
 tick=F, # removes the tick marks from the axis
 bty="n") # removes the plot border

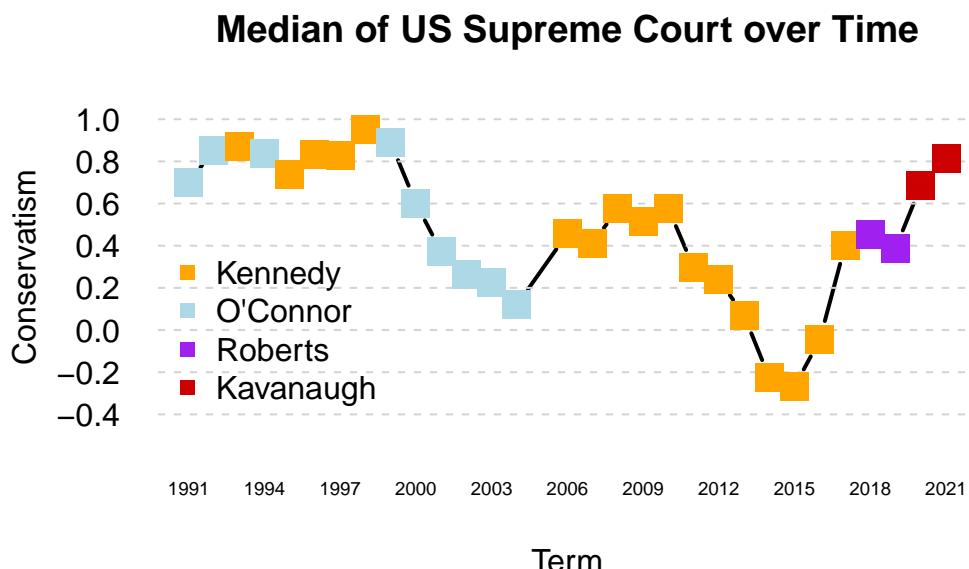
adds horizontal dashed gray lines
abline(h=seq(-.4, 1, .2), lty=2, col="light gray")

Adds a legend
legend("bottomleft", pch=15, col = c("orange", "light blue", "purple", "red3"),
 c("Kennedy", "O'Connor", "Roberts", "Kavanaugh"), bty="n")

Adds the color-coded points
points(x =names(medians), y= medians,
 col= ifelse(results == "AMKennedy", "orange",
 ifelse(results == "SDOConnor", "light blue",
 ifelse(results == "JGRoberts", "purple",
 "red3"))),
 pch=15, cex=2)

Adds our custom x-axis
axis(1, names(medians), cex.axis=.6, tick=F)

```



#### 6.2.4 Wrapping Up

We have calculated and visualized how the median U.S. Supreme Court Justice and Justice's ideology has changed over the past three decades.

- This gave us additional practice with loops and visualization
- We also gained exposure to an example of how political scientists take a large amount of information—votes on all Supreme Court cases—and try to summarize it using a single number that represents how liberal or conservative a justice is

This type of information can be used for many social science goals: 1) To describe trends in the Court 2) To help explain why the Court has voted a particular way on recent cases 3) To predict how the Court will vote in the future as new justices arrive.

With Amy Coney Barrett now on the Court for multiple terms and Ketanji Brown Jackson beginning to decide cases, the MQ scores will continue to be updated to allow for future exploration of these dynamics.

FEB. 25, 2022, AT 11:04 AM

# How Ketanji Brown Jackson Could Change The Supreme Court

By [Amelia Thomson-DeVeaux](#)

Graphics by [Elena Mejía](#)

Filed under [Supreme Court](#)



KEVIN LAMARQUE / POOL / AFP / GETTY IMAGES

Figure 6.2: FiveThirtyEight

# 7 Prediction

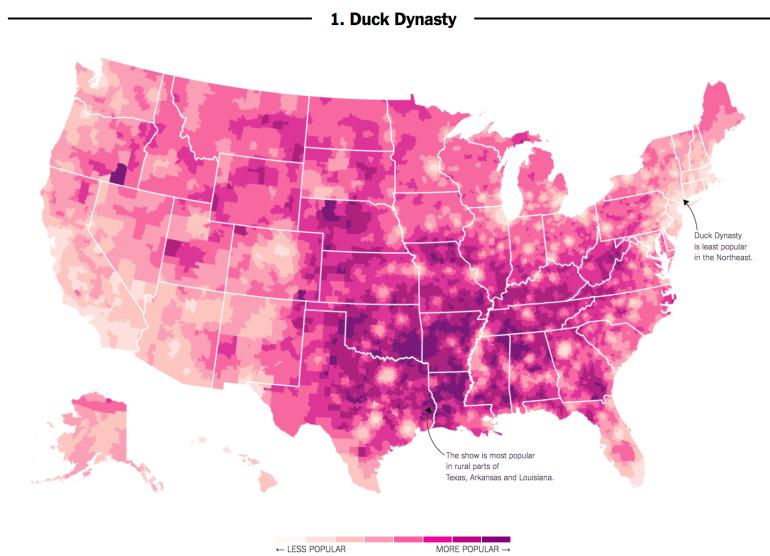
In this section, we move to our next social science goal

- Describe
- Explain, evaluate, and recommend → Causality
- **Predict**
- Discover

Most of the tools we have been working on thus far have focused on first describing our data and then conducting tests through different types of comparisons and visualizations, in order to assess a deductive hypothesis, explaining the relationship between two variables.

Now we turn to a different goal.

Recall the difference between Correlation vs. Causality using our graphic showing the popularity of *Duck Dynasty* in different parts of the country. In 2016, researchers at the [NY Times](#) noticed that areas in the country where the television show Duck Dynasty was popular also tended to support Donald Trump at higher rates.



For those used to working with the goal of explanation, shifting to prediction and classification may mean we need to shift what types of information we think is important.

- Correlation: Areas that watch Duck Dynasty are more likely to support Trump (degree to which two variables “move together”)
- Causality: Watching Duck Dynasty (vs. not watching) causes you to support Trump.

If we were interested in the goal of explaining voting decisions (what causes someone to vote a certain way?), we might not care if someone watches the show. However, if we were just interested in predicting vote share or voting decisions, a strong correlation could still be useful. Without spending a single dollar on surveying a community, we might have a general sense of their support for a candidate.

## 7.1 Prediction Overview

Our goal: Predict (estimate/guess) some unknown using information we have as accurately and precisely as possible

- Prediction could involve estimating a numeric outcome. Alternatively, prediction also involves classification— predicting a categorical outcome (e.g., prediction of who wins vs. who loses).

Some political science examples of this might include

1. Categorizing comments on social media as being toxic/nasty/uncivil
2. Detecting Fake news and misinformation
3. Forecasting election results

NICHOLAS THOMPSON BUSINESS 06.29.2017 89:00 AM

## Instagram Unleashes an AI System to Blast Away Nasty Comments

The social media site wants to turn itself into the friendliest place on the internet.

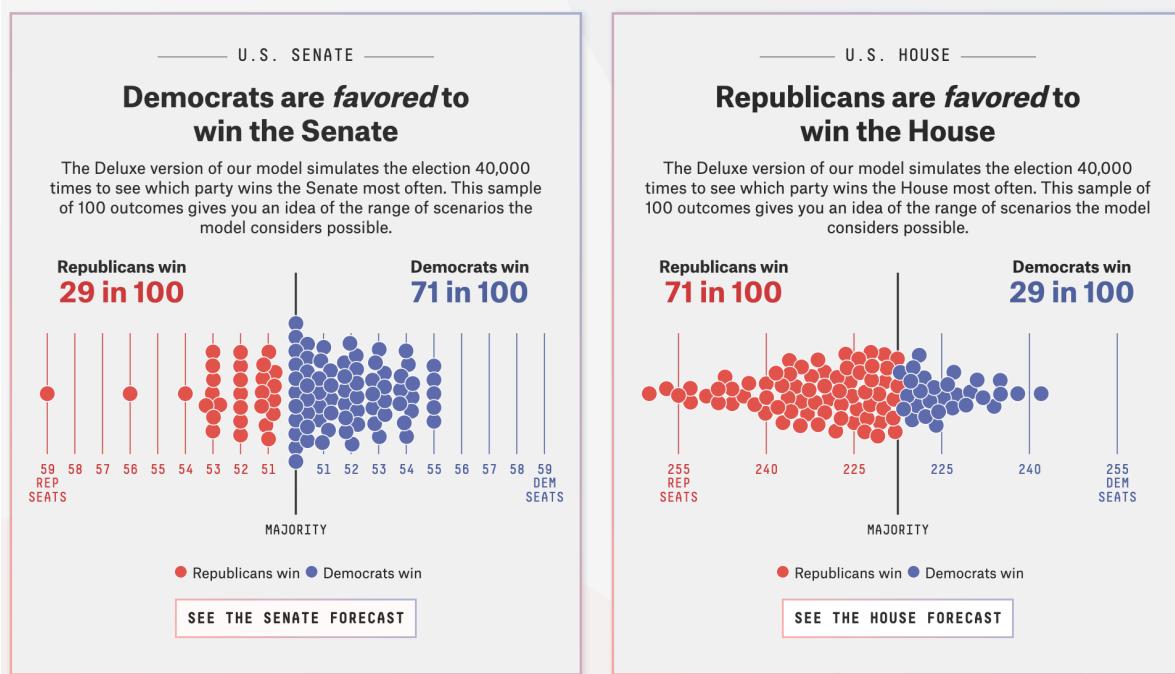


Figure 7.1: Wired



Figure 7.2: PBS

UPDATED SEP. 16, 2022, AT 8:57 PM



Other examples

- Trying to detect hate speech online
- Predicting where or when an attack might occur
- Trying to classify a large amount of text into subject or topic categories for analysis

What other types of things might we try to predict or classify in political science?

## 7.2 Process of Prediction

Predict (estimate/guess) some unknown using information we have – and do so as accurately and precisely as possible.

1. Choose an approach
  - Using an observed (known) measure as a direct proxy to predict an outcome
  - Using one or more observed (known) measures in a regression model to predict an outcome
  - (Beyond the course) Using a statistical model to select the measures to use for predicting an outcome

## 2. Assess accuracy and precision

- Prediction error:  $Prediction - Truth$
- Bias: Average prediction error:  $\text{mean}(Prediction - Truth)$ 
  - A prediction is ‘unbiased’ if the bias is zero (If the prediction is on average true)
- Root-mean squared error:  $\sqrt{\text{mean}((Prediction - Truth)^2)}$ 
  - Like ‘absolute’ error – the average magnitude of the prediction error
  - the typical distance the prediction is from the truth
- Confusion Matrix
  - A cross-tab of predictions you got correct vs. predictions you got wrong (misclassified)
  - Gives you true positives and true negatives vs. false positives and false negatives

## 3. Iterate to improve the prediction/classification

- Often, we repeat steps 1-3 until we are confident in your method for predicting.
4. Danger Zone: Eventually, after you have tested the approach and are satisfied with the accuracy, you may start applying it to new data for which you do not know the right answer.

## 7.3 Example: Forecasting 2020 US Election based on 2016 Results

Let’s try to predict the 2020 election results using just the 2016 results.

*For a video explainer of the code for this application, see below. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)*

<https://www.youtube.com/watch?v=zWDxZogRwOs>

```
results2020 <- read.csv("elecresults2020.csv", stringsAsFactors = T)
```

Variables

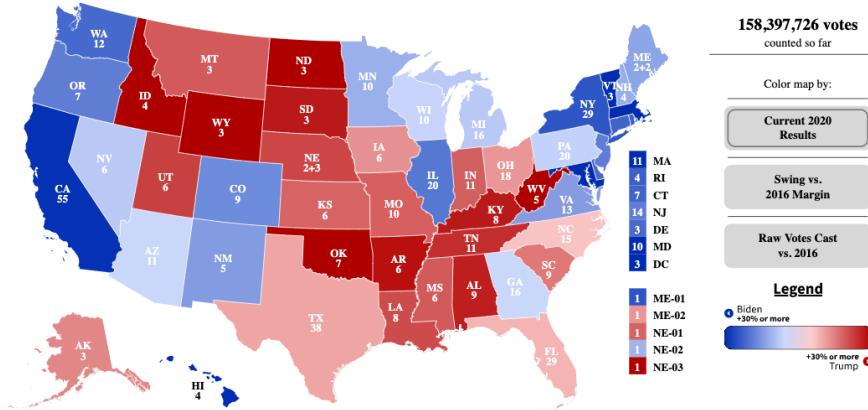
- **state**: state or state and district
- **called**: result of 2020 election
- **margin2016**: two-party margin in 2016. Positive values indicate Democratic win, negative indicate Republican win
- **EV**: Electoral votes associated with a state/ district of a state



**Joe Biden** ✓  
306 Electoral Votes  
81,282,916 votes  
51.3%

**Donald Trump**  
232 Electoral Votes  
74,223,369 votes  
46.9%

The  
Cook  
POLITICAL  
REPORT



```
sum(results2020$EV[results2020$called == "R"])
```

[1] 232

```
sum(results2020$EV[results2020$called == "D"])
```

[1] 306

### 7.3.1 Choose Approach

- 1) Choose an approach: Using an observed (known) measure as a direct proxy to predict an outcome
  - Let's use the 2016 result as a direct proxy to predict 2020.

```
results2020$predicted2020 <- ifelse(results2020$margin2016 < 0, "R", "D")
results2020$predicted2020 <- as.factor(results2020$predicted2020)
```

### 7.3.2 Assess Accuracy

2) Assess accuracy

What proportion of states did we get correct?

```
mean(results2020$predicted2020 == results2020$called)
```

```
[1] 0.8928571
```

Classification

We want to correctly predict the winner of each state

Prediction of binary outcome variable = classification problem

- true positive: correctly predicting Biden to be the winner
- false positive: incorrectly predicting Biden to be the winner (misclassification)
- true negative: correctly predicting Biden to be the loser
- false negative: incorrectly predicting Biden to be the loser (misclassification)

We define one outcome as the “positive” and one as the “negative.” Here we will say a Biden win is the positive and a Trump win is the negative. You could flip this and make a Trump win the positive and a Biden win the negative. This terminology comes from settings where there is a more objective positive vs. negative result (e.g., a positive medical test result) than most social science settings. The key thing is that we are trying to identify different types of correct classifications vs. misclassifications.

Confusion Matrix: Tells us how we went right, how we went wrong.

```
table(predicted=results2020$predicted2020, actual = results2020$called)
```

	actual	
predicted	D	R
D	22	0
R	6	28

Which states did we get wrong?

```
results2020$state[results2020$predicted2020 != results2020$called]
```

```
[1] Arizona Georgia Michigan
[4] Nebraska 2nd District Pennsylvania Wisconsin
56 Levels: Alabama Alaska Arizona Arkansas California Colorado ... Wyoming
```

### 7.3.3 Iterate to improve predictions

Start back at step one. We continue to repeat steps 1 and 2 until we are confident in our predictions.

How could we improve our predictions of elections? What other information could we use?

## 7.4 Example: Using polls to predict the 2020 election results

*For a video explainer of the code for this application, see below. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)*

<https://www.youtube.com/watch?v=zWDxZogRwOs>

Many forecasters use pre-election polls in their models to predict election outcomes. In 2016 and 2020, polling-based forecasts received a lot of criticism

Prior to the 2016 elections, forecasts that used polls seemed confident that Hillary Clinton would win the election. [Political analysts](#) also seemed to think the polls were favorable to Clinton.

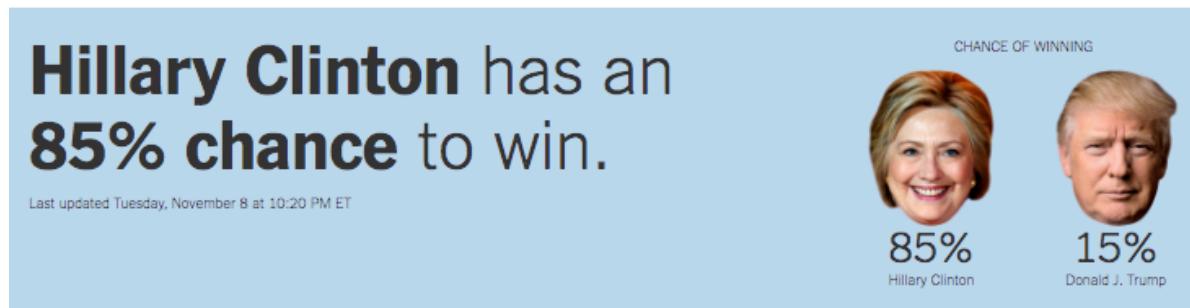


Figure 7.3: NY Upshot

We all know that afterwards, Clinton did not win.

This led public opinion scholars and practitioners to do a deep investigation into the quality of pre-election polling. Like 2016, following the 2020 election, a similar team investigated the

## Why 2016 election polls missed their mark

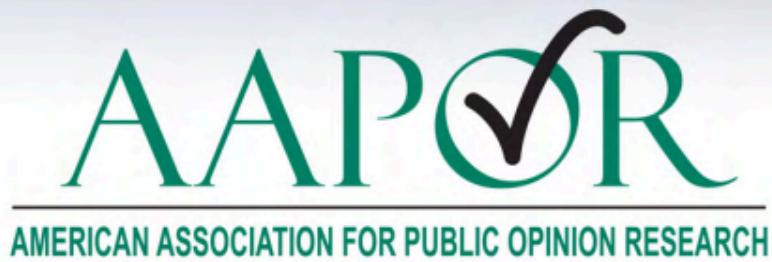
BY ANDREW MERCER, CLAUDIA DEANE AND KYLEY MCGEENEY



Supporters of presidential candidate Hillary Clinton watch televised coverage of the U.S. presidential election at Comet Tavern in the Capitol Hill neighborhood of Seattle on Nov. 8. (Photo by Jason Redmond/AFP/Getty Images)

Figure 7.4: Pew

quality of pre-election polling in 2020. Here, while many polls pointed to a favorable outcome for Biden, the results seemed closer than one might have anticipated.



# Task Force on 2020 Pre-Election Polling: An Evaluation of the 2020 General Election Polls

The results of these findings are in the [AAPOR report](#).

## 7.4.1 Choose an approach: Let's analyze some polls

We are going to do our own analysis of pre-election polls as a prediction of the 2020 election results. We will use a large number of state polls conducted from May-November 2020 that were made available to the public on FiveThirtyEight.

```
polls2020 <- read.csv("pollsandresults2020.csv", stringsAsFactors = T)
```

Variables

- TrumpPoll, BidenPoll: Poll-based vote share for Biden or Trump
- TrumpResult, BidenResult: Actual vote share for Biden or Trump
- EV: Electoral votes associated with state/CD
- days\_to\_election: Days until Election Day
- stateid: state abbreviation
- fte\_grade: FiveThirtyEight Pollster grade
- sample\_size: Poll sample size

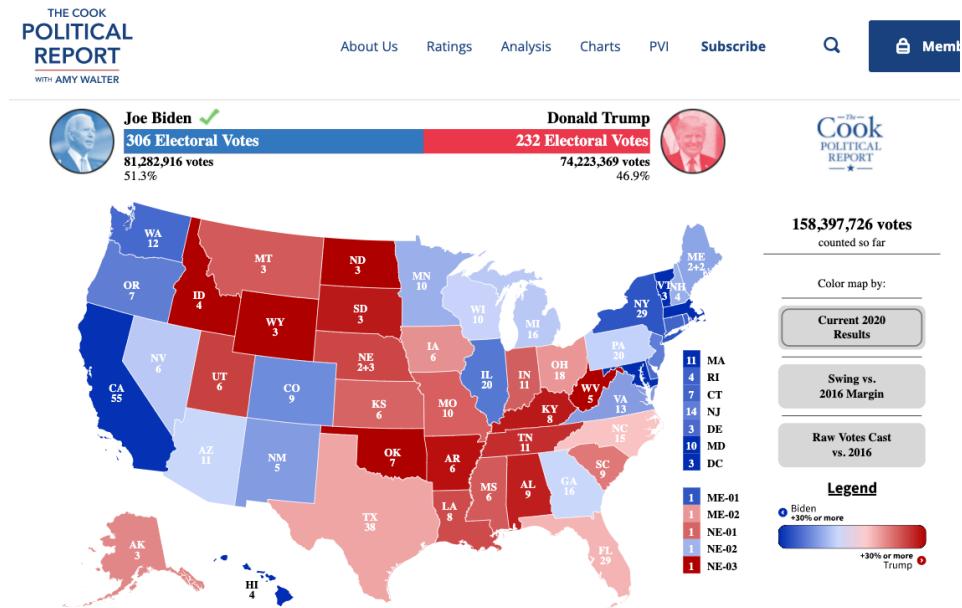
Can we predict the outcome of an election using polls?

Let's create our outcome variables.

```
Biden's margin of victory (or defeat) in the polls
polls2020$polldiff <- polls2020$BidenPoll - polls2020$TrumpPoll
```

```
Biden's margin of victory (or defeat) in the actual election result
polls2020$resultdiff <- polls2020$BidenResult - polls2020$TrumpResult
```

Positive numbers mean Biden was ahead/won. Negative mean Trump was ahead/won.



Let's predict the amount of electoral votes for Biden based on polls in each state close to Election Day.

Let's start with 1 state.

- Let's grab all polls within 2 weeks of the election or the most recent day polled (for areas that did not have recent polls)

```
Iteration vector
states <- unique(polls2020$stateid)
states[1]
```

[1] AL  
55 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA ... WY

```

Subset to just Alabama
subdata <- subset(polls2020, stateid == states[1])

Further subset to the "latest polls"
subdata <- subset(subdata, days_to_election < 15 |
 days_to_election == min(subdata$days_to_election))

```

Now let's extract the actual margin for Biden, the poll-based predicted margin, and finally, let's assign electoral votes based on our prediction.

```

Find the margin for the actual result
result.marginAL <- mean(subdata$resultdiff)
result.marginAL

```

[1] -25.4

```

Find the margin for our prediction
polls.marginAL <- mean(subdata$polldiff)
polls.marginAL

```

[1] -21.16667

```

Allocate votes for Biden according to the margin
bidenvotesAL <- ifelse(mean(subdata$polldiff) > 0,
 unique(subdata$EV), 0)
bidenvotesAL

```

[1] 0

We predicted Biden would lose Alabama because the `polls.marginAL` is negative. Therefore, we assigned Biden 0 electoral votes in this example.

#### 7.4.2 Loop through all states

```

Iteration vector
states <- unique(polls2020$stateid)
Container vector

```

```

polls.margin <- result.margin <- bidenvotes <-
 rep(NA, length(states))

names(polls.margin) <- names(result.margin) <-
 names(bidenvotes) <-as.character(unique(states))

Loop
for(i in 1:length(states)){
 subdata <- subset(polls2020, stateid == states[i])
 subdata <- subset(subdata, days_to_election < 15 |
 days_to_election == min(subdata$days_to_election))
 result.margin[i] <- mean(subdata$resultdiff)
 polls.margin[i] <- mean(subdata$polldiff)
 bidenvotes[i] <- ifelse(mean(subdata$polldiff) > 0,
 unique(subdata$EV), 0)
}
sum(bidenvotes) # predicted

```

[1] 351

### 7.4.3 Check Accuracy

#### 7.4.3.1 Quantitative Measures of Accuracy

Let's calculate two common measures of prediction error: bias (the average prediction error) and root-mean-squared error (a typical magnitude of the prediction error).

```

Calculate Bias (Predicted Biden - True Biden)
predictionerror <- polls.margin -result.margin
bias <- mean(predictionerror)

Root Mean Squared Error
sqrt(mean((predictionerror)^2))

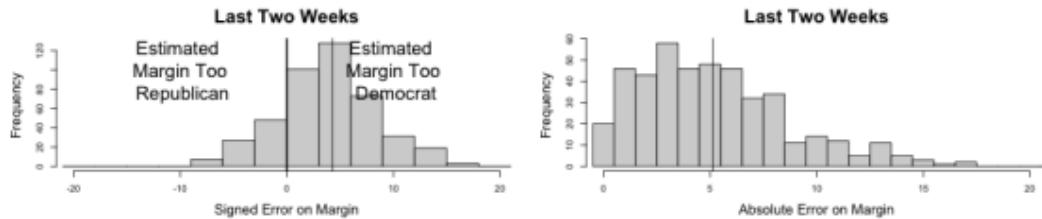
```

[1] 6.052873

On average, the poll-based prediction was more than 4 points larger for Biden's margin than the actual result.

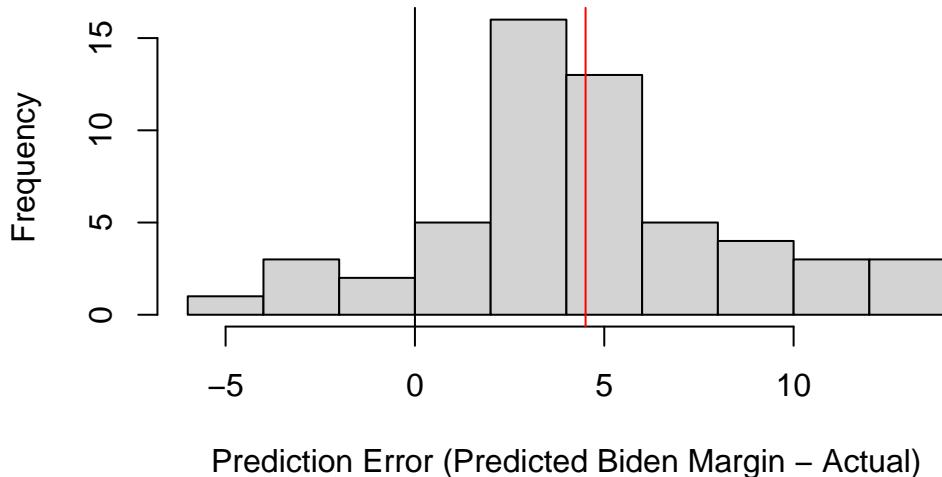
We can create a plot similar to the left plot from the AAPOR report.

To illustrate the stability of the polling error in state-level presidential polls across the election campaign, Figure 5 graphs the distribution of signed error and absolute error for all state-level presidential polls conducted in the last two weeks, the last week, and the last three days of the 2020 election. Regardless of the timeframe, the distributions and the averages (indicated by a vertical line in each histogram) of the polling error are nearly identical.<sup>20</sup>



```
Histogram of Prediction Errors to Show Bias
hist(predictionerror,
 xlab = "Prediction Error (Predicted Biden Margin - Actual)",
 main = "Histogram of Prediction Error in Latest Polls")
abline(v=mean(predictionerror), col="red")
abline(v=0)
```

## Histogram of Prediction Error in Latest Polls



Bonus: Another way to visualize the prediction error

- Let's create our own version of this AAPOR Plot

## 7. Performance of 2020 Polls by State

*Polls in most states overstated the Biden-Trump margin. Even within states, polls overstated the Democratic-Republican margin in senatorial and gubernatorial contests more than they overstated the Biden-Trump margin. The polling error was larger the more support Trump received in 2016 even after accounting for other between-state differences plausibly related to polling error.*

The correlates of polling error by state can be examined by exploring whether or not the average polling error in a state correlates with a state's political environment.

Figure 11 begins this exploration by plotting the average signed error for each state. It is immediately obvious that polls overstated the Biden-Trump margin in nearly every state. Polls overstated the relative support for Trump in only a handful of states.

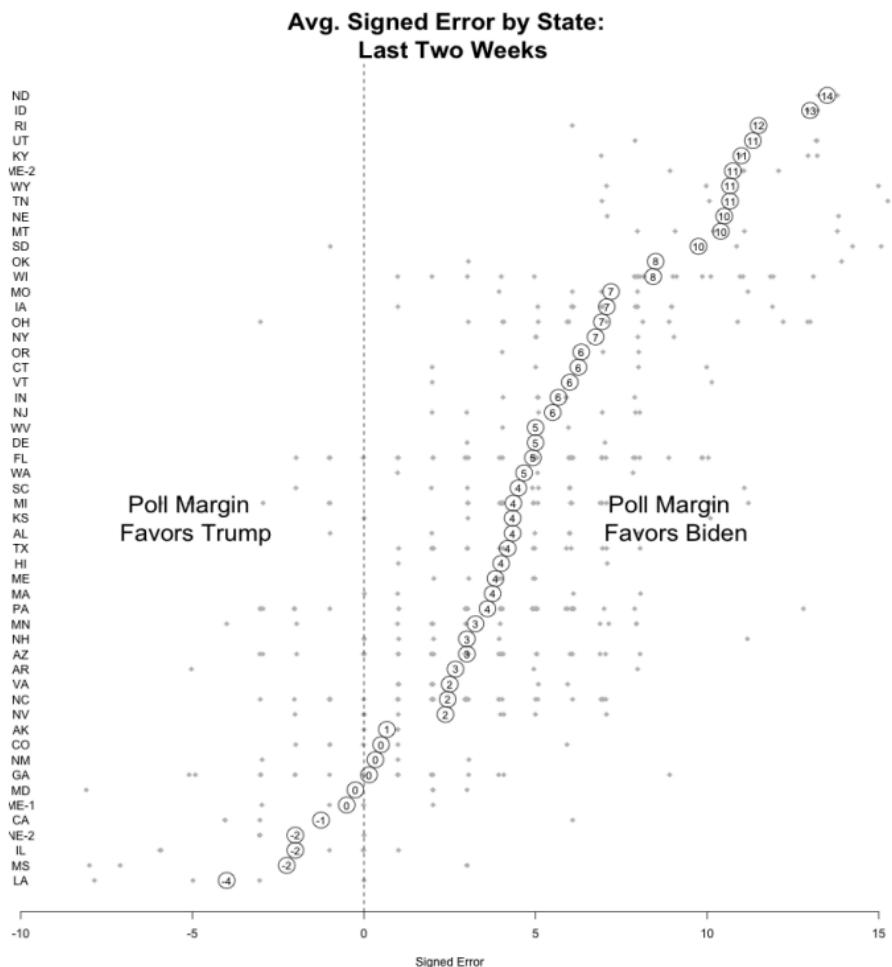
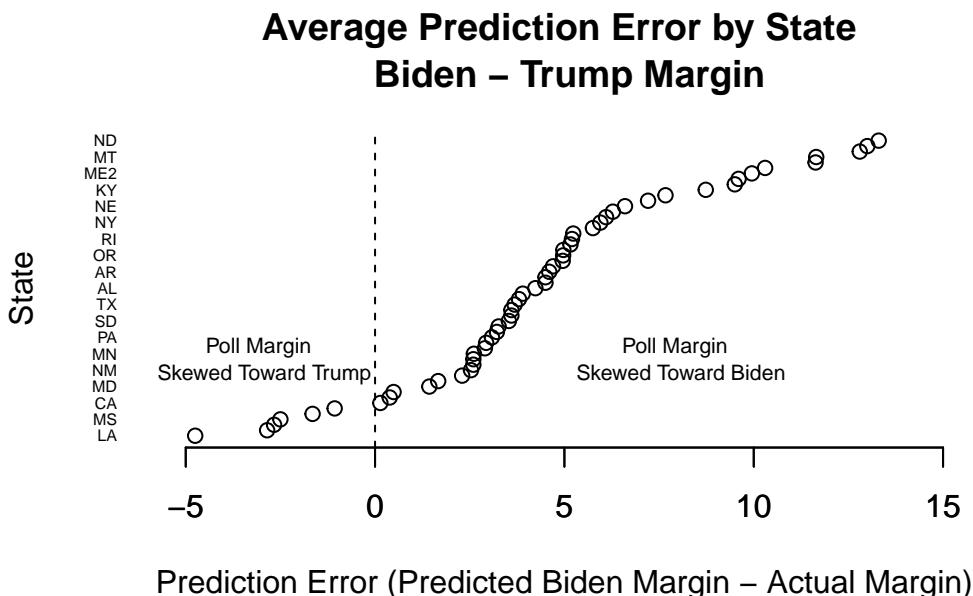


Figure 11. State-Level Average Signed Error by State. Circles denote the average signed error for each state. The signed error of individual polls in each state are plotted in grey points.

We will plot the prediction error on the x-axis, and list the corresponding states on the y-axis.

- We will sort the prediction error to make it easier to see the pattern of results.

```
plot(x=sort(predictionerror), y=1:length(predictionerror),
 main="Average Prediction Error by State \n Biden - Trump Margin",
 ylab="State",
 xlab="Prediction Error (Predicted Biden Margin - Actual Margin)",
 yaxt="n",
 bty="n",
 xlim = c(-5, 15))
abline(v=0, lty=2)
axis(2, 1:length(predictionerror), labels=names(sort(predictionerror)), las=1,
 cex.axis=.5,tick=F)
axis(1, seq(-5, 15, 5), seq(-5, 15, 5))
text(-3, 15, "Poll Margin \n Skewed Toward Trump", cex=.7)
text(8, 15, "Poll Margin \n Skewed Toward Biden", cex=.7)
```



#### 7.4.3.2 Classification

Instead of quantifying how far we were off, let's see where we were right vs. where we were wrong.

## Classification

- true positive: correctly predicting Biden to be the winner
- false positive: incorrectly predicting Biden to be the winner
- true negative: correctly predicting Biden to be the loser
- false negative: incorrectly predicting Biden to be the loser

## Confusion Matrix

Let's classify our predictions.

```
actualwins <- ifelse(result.margin > 0, "Biden Won", "Trump Won")
predictedwins <- ifelse(polls.margin > 0, "Biden Won", "Trump Won")
```

```
table(predictedwins, actualwins)
```

		actualwins	
		Biden Won	Trump Won
predictedwins	Biden Won	28	3
	Trump Won	0	24

Where did the polls get it wrong?

```
actualwins[actualwins != predictedwins]
```

```
FL ME2 NC
"Trump Won" "Trump Won" "Trump Won"
```

What's your conclusion?

- Are the polls alright?
- How could you improve the prediction?
- Wait a second... why even poll?

# 8 Prediction with Regression

We are continuing our topic of prediction, this time adding a new tool: linear regression.

Recall that we predict (estimate/guess) some unknown using information we have – and do so as accurately and precisely as possible.

1. Choose an approach
  - Using an observed (known) measure as a direct proxy to predict an outcome
  - *Using one or more observed (known) measures in a regression model to predict an outcome*
  - (Beyond the course) Using a statistical model to select the measures to use for predicting an outcome
2. Assess accuracy and precision
3. Iterate to improve the prediction/classification
  - Often, we **repeat steps 1-3** until we are confident in your method for predicting.
4. Danger Zone: Eventually, after you have tested the approach and are satisfied with the accuracy, you may start applying it to new data for which you do not know the right answer.

## 8.1 Regression in the wild.

Regression is used across many domains for prediction and classification, from fantasy football to making World Cup predictions, or even predicting how far a contestant will go on *The Bachelor* or *The Bachelorette*.

The screenshot shows a Reddit post on the r/fantasyfootball subreddit. The post is titled "Using Linear Regression to Make Fantasy Football Picks" and was posted by u;brnko 3 years ago. It has 56 upvotes and a link to medium.com/ml-eve... . The post has 12 comments, an award icon, share, save, hide, and report options, and is 84% upvoted. The post is displayed on a light gray background with a blue header.

Using data to predict reality TV outcomes.

The screenshot shows a news article from NBC titled "Using Data to Predict This Season's Winner of 'The Bachelor'". The article discusses how a simple Excel spreadsheet revealed more secrets about "The Bachelor" to the author than Chris Harrison ever could. The author is Lindsay Schupps. The article includes a photo of a beach and a screenshot of a mobile app interface for a fantasy league.

In politics, we might use regression to build campaign models— predicting which voters are persuadable, which supporters will volunteer at campaign events, which supporters will turn out to vote, etc.

## **Political Campaigns and Big Data**

### **Faculty Research Working Paper Series**

---

**David W. Nickerson**

University of Notre Dame

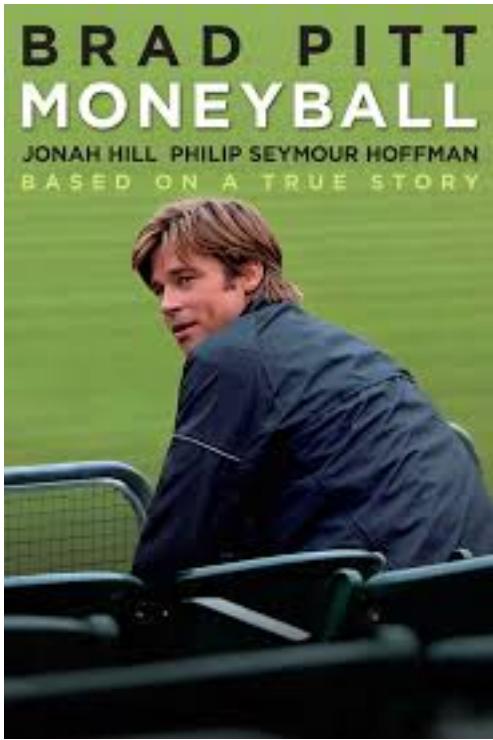
**Todd Rogers**

Harvard Kennedy School

### **8.2 Application: Baseball Predictions**

For our first example, we will stay outside of politics and use regression to predict the success of a baseball team.

[Moneyball](#) is a \$100 million Hollywood movie that is all about linear regression... and some baseball... and Brad Pitt, but really... it's MOSTLY about linear regression



The movie describes the Oakland A's shift to start using data to build their team. They make two observations 1) To win baseball games, you need runs. 2) To score runs, you need to get on base. We can estimate what on base percentage we would need as a team to score enough runs to make the playoffs in a typical season.

We will use regression to make these predictions.

*For a video explainer of the code for this application, see below. (Via youtube, you can speed up the playback to 1.5 or 2x speed.)*

<https://www.youtube.com/watch?v=wtn-W8Uv32E>

We use `baseball.csv` data

- RS: runs scored;
- RA: runs allowed;
- W: wins;
- Playoffs: whether team made playoffs;
- OBP: on base percentage;
- BA: batting average;
- SLG: Slugging Percentage

```
baseball <- read.csv("baseball.csv")
```

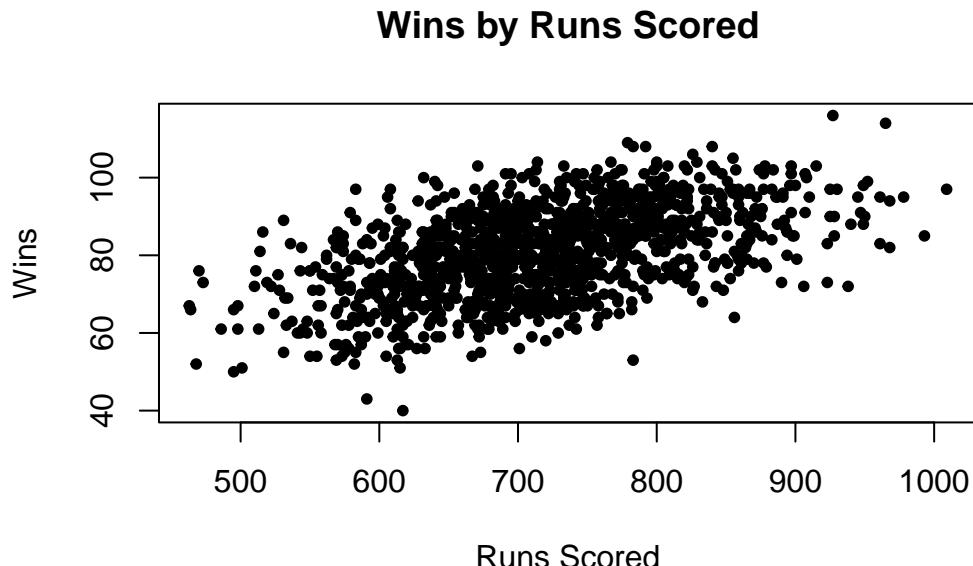
```
head(baseball)
```

	Team	League	Year	RS	RA	W	OBP	SLG	BA	Playoffs	RankSeason
1	ARI	NL	2012	734	688	81	0.328	0.418	0.259	0	NA
2	ATL	NL	2012	700	600	94	0.320	0.389	0.247	1	4
3	BAL	AL	2012	712	705	93	0.311	0.417	0.247	1	5
4	BOS	AL	2012	734	806	69	0.315	0.415	0.260	0	NA
5	CHC	NL	2012	613	759	61	0.302	0.378	0.240	0	NA
6	CHW	AL	2012	748	676	85	0.318	0.422	0.255	0	NA

	RankPlayoffs	G	OOBP	OSLG
1	NA	162	0.317	0.415
2	5	162	0.306	0.378
3	4	162	0.315	0.403
4	NA	162	0.331	0.428
5	NA	162	0.335	0.424
6	NA	162	0.319	0.405

Below we can see the first observation made: Runs scored are highly correlated with team wins



What the A's noticed is that a team's On Base Percentage is also highly correlated with runs scored. This aligns with conventional wisdom. Players get a lot of hype when they achieve a high OBP.

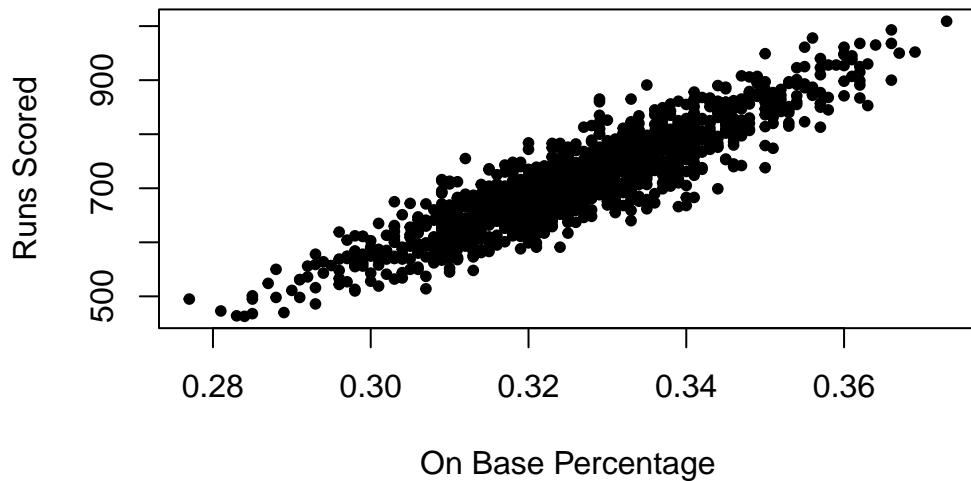


Bryce Harper had one of the Phillies' three homers in Game 5, bringing their playoff total to 23, the second most all time through 11 games of a postseason. Mark J. Rebilas-USA TODAY Sports

Harper is one of only 11 players with at least 100 plate appearances and a postseason on base percentage plus slugging percentage greater than 1.000. - USA Today

This correlation shows up in our data, too.

## Runs Scored by On Base Percentage

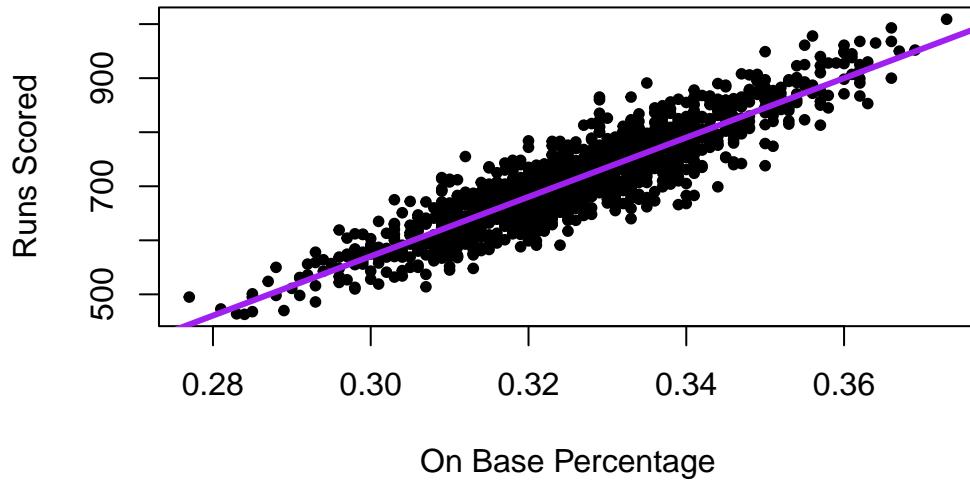


### 8.3 Step 1: Approach- Regression in R

A regression draws a “best fit line” between the points. This allows us – for any given OBP – to estimate the number of runs scored.

- Our best prediction of the number of runs scored would be the spot on the purple line directly above a given OBP.

## Runs Scored by On Base Percentage



The regression model is  $Y = \alpha + \beta X + \epsilon$ . Let's demystify this.

- A regression model describes the relationship between one or more independent variables  $X$  (explanatory variables) and an outcome variable  $Y$  (dependent variable)
  - For example, the relationship between our independent variable, On Base Percentage, and our dependent variable, Runs Scored
- We want to know what happens with our dependent variable  $Y$  if our independent variable  $X$  increases.
  - As we increase our On Base Percentage, a regression model will help us estimate how much we should expect our Runs Scored to increase (or decrease)
- $\alpha$  and  $\beta$  are considered “parameters” – things we don’t know but want to estimate. These two numbers will define exactly how we think  $X$  and  $Y$  are related.
- No two variables are perfectly related, so we also have the  $\epsilon$  term, which describes the error in the model

When we have data, we estimate  $Y$ ,  $\alpha$ , and  $\beta$ :  $\hat{Y} = \hat{\alpha} + \hat{\beta}X$ .

- The  $\hat{}$  over the letters means those are our estimated values.

In R, the regression syntax is similar to what we use in making a boxplot: `fit <- lm(y ~ x, data = mydata)`

- `fit` is just whatever you want to call the output of the model,

- `y` is the name of the dependent variable,
- `x` is the name of the independent variable, and
- `mydata` is whatever you have called your dataframe. E.g.:

```
fit <- lm(RS ~ OBP, data = baseball)
```

When we have data, we estimate  $Y$ ,  $\alpha$ , and  $\beta$ :  $\hat{Y} = \hat{\alpha} + \hat{\beta}X$ .

- Our model gives us the “coefficient” estimates for  $\hat{\alpha}$  and  $\hat{\beta}$ .

```
coef(fit)
```

(Intercept)	OBP
-1076.602	5490.386

The first coefficient is  $\hat{\alpha}$ , this represents the intercept – the estimated value our dependent variable will take if our independent variable is 0.

- The value the estimated runs scored would be if a team had a 0.000 on base percentage. In our case, this value is estimated to be negative, which is impossible (but it would also be unusual for a team to have a 0.000 on base percentage). Therefore, the intercept isn’t inherently substantively interesting to us.

The second coefficient is  $\hat{\beta}$  is the slope This represents the expected change in our dependent variable for a 1-unit increase in our independent variable.

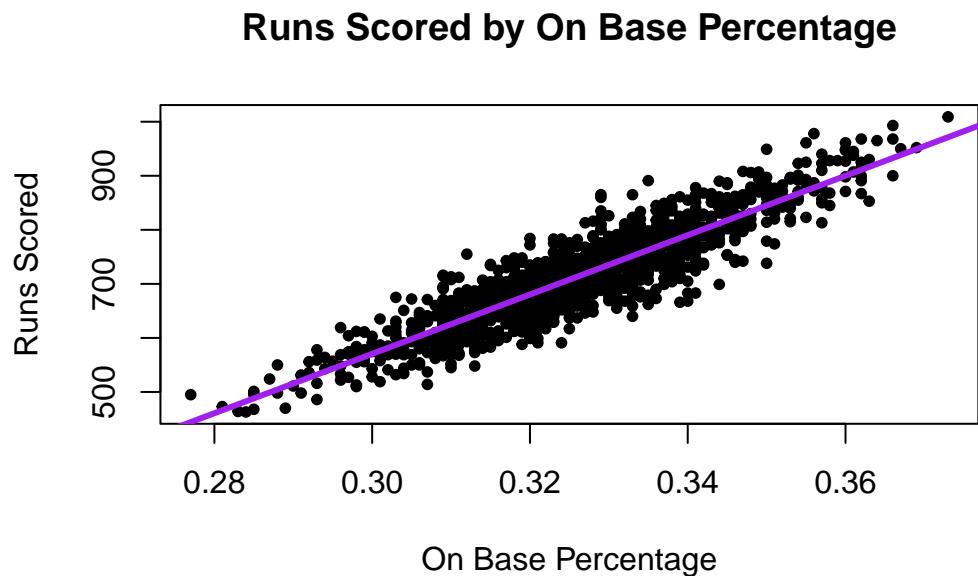
- For example, if we go from a 0.000 on base percentage to a 1.000 on base percentage, we would expect a 5490.4 increase in runs scored.
- Note: slope can be positive or negative similar to correlation.
- Note: slope is in the units of the dependent variable (e.g., runs). It is not constrained to be between -1 and 1.
- It is telling us that the greater the OBP, the better!

### 8.3.1 Visualizing a regression

We can plot the regression using a scatterplot and `abline()`.

```
plot(x=baseball$OBP, y=baseball$RS,
 ylab = "Runs Scored",
 xlab = "On Base Percentage",
 main="Runs Scored by On Base Percentage",
 pch=20)
```

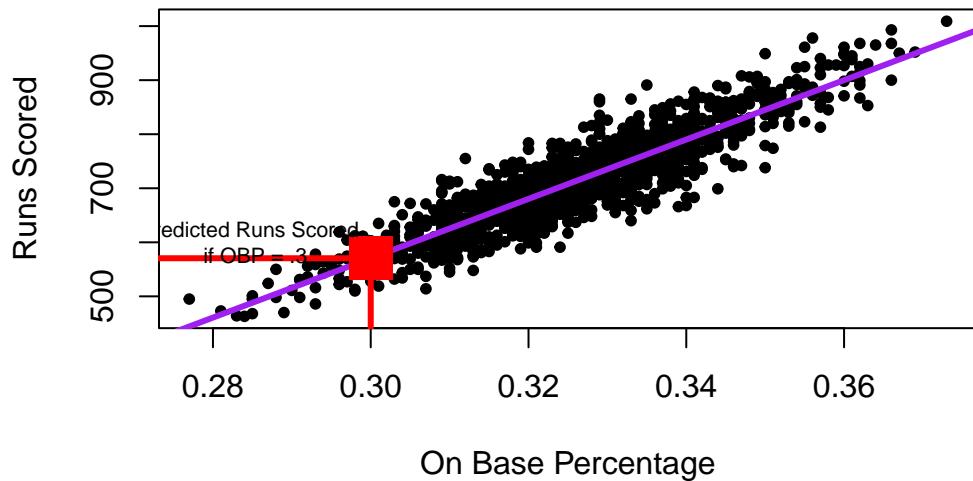
```
Add regression line
abline(fit, lwd=3, col = "purple") # add line
```



#### 8.3.2 Making predictions with regression

A regression model allows us to estimate or “predict” values of our dependent variable for a given value of our independent variable.

## Runs Scored by On Base Percentage



The red dot represents our estimate (best prediction) of the number of runs scored if a team has an on base percentage of .300. In R, we can calculate this value using `predict()`.

- The syntax is `predict(fit, data.frame(x = value))` where `fit` is the name of the model, `x` is the name of the independent variable, and `value` represents the value for the independent variable for which you want to predict your outcome (e.g., .300).

```
predict(fit, data.frame(OBP=.300))
```

```
1
570.5137
```

Under the hood, this is just using the regression formula described above. For example, to estimate the number of runs scored for a .300 on base percentage, we take  $\hat{\alpha} + \hat{\beta} * .300$

- Note that below we compare the output of the `predict` function to our output if we manually calculated the estimated value.

```
predict(fit, data.frame(OBP=.300))
```

```
1
570.5137
```

```
a + b*.300
coef(fit)[1] + coef(fit)[2]*.300
```

```
(Intercept)
570.5137
```

Let's say a team thought they needed about 900 runs scored to get to the playoffs, and they were pretty sure they could get a team on base percentage of .500. How many runs would they be expected to score with that OBP? Do you think they will make the playoffs?

Try on your own, then expand for the solution.

```
predict(fit, data.frame(OBP=.500))
```

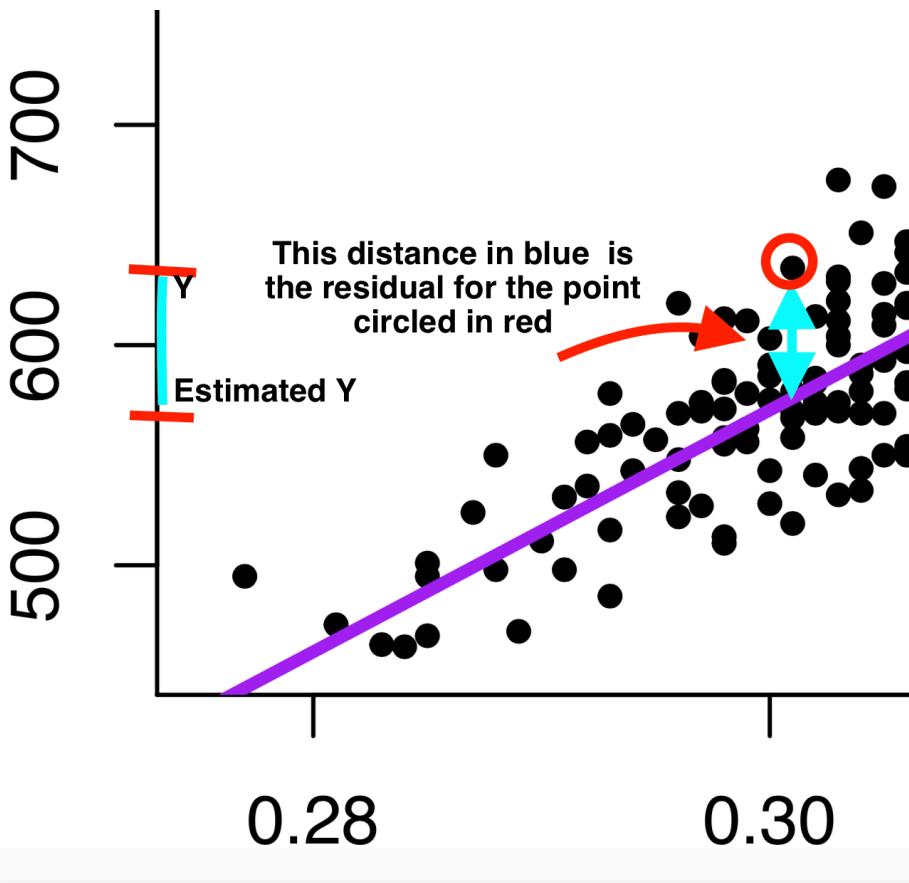
```
1
1668.591
```

It's greater than 900, so we should feel good about our chances.

## 8.4 Step 2: Checking accuracy of model

Understanding prediction error: Where do  $\hat{\alpha}$  and  $\hat{\beta}$  come from? Recall that a regression tries to draw a “best fit line” between the points of data.

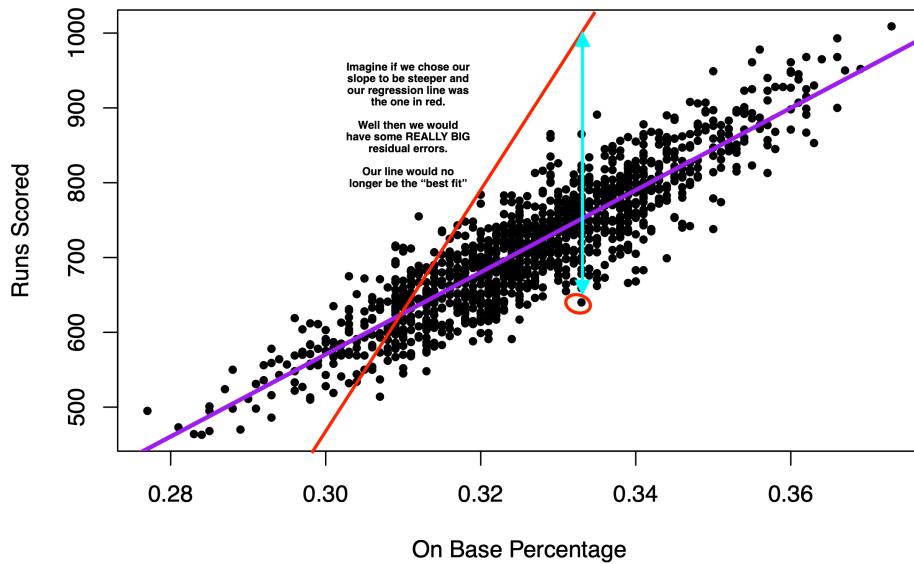
Under the hood of the regression function, we are searching for the values of  $\hat{\alpha}$  and  $\hat{\beta}$  that try to minimize the distance between the individual points and the regression line.



This distance is called the residual:  $\hat{\epsilon}_i = Y_i - \hat{Y}_i$ .

- This is our ***prediction error***: How far off our estimate of  $Y$  is ( $\hat{Y}_i$ ) from the true value of  $Y$  ( $Y_i$ )
- Linear regressions choose  $\hat{\alpha}$  and  $\hat{\beta}$  to minimize the “squared distance” of this error (think of this as the magnitude of the distance). This is why we tend to call this type of linear regression ordinary least squares (OLS regression).

If instead we chose the red line in the image below to be the regression line, you can see that the typical prediction error would be much larger. That's why we end up with the purple line.



#### 8.4.1 Root Mean Squared Error

Just like we had root mean squared error in our poll predictions, we can calculate this for our regression.

- Just like with the polls, this is the square root of the mean of our squared prediction errors, or “residuals” in the case of regression.
  - R will give us this output using this formula where `fit` is the hypothetical name we have provided for the regression model: `sqrt(mean(fit$residuals^2))`

```
sqrt(mean(fit$residuals^2))
```

```
[1] 39.78956
```

- In our case, using on based percentage to predict runs scored, our estimates are off typically, by about 40 runs scored.
- On the graph, this means that the typical distance between a black point and the purple line is about 40.

## 8.5 Step 3: Iterate and Compare Models

When building predictive models, often researchers want to minimize this Root-Mean Squared Error – minimizing the magnitude of the typical prediction error (the distance between the actual value of our outcome, and the true value)

Example: Let's compare the RMSE from two different models:

```
Predicting Runs Scored with OBP
fit <- lm(RS ~ OBP, data = baseball)

Predicting Runs Scored with Batting Average
fit2 <- lm(RS ~ BA, data = baseball)
```

The Oakland A's noticed that OBP was a more precise predictor than BA, and RMSE gives us one way to assess this.

### 8.5.1 Regression with Multiple Predictors

You can also add more than 1 predictor to a regression using the + sign.

```
Predicting Runs Scored with OBP and Slugging Percentage
fit3 <- lm(RS ~ OBP + SLG, data = baseball)
sqrt(mean(fit3$residuals^2))
```

```
[1] 25.09135
```

Look how the RMSE dropped again, improving our prediction.

## 8.6 Application: Predicting Campaign Donations

Can we predict campaign donations?



Data from Barber, Michael J., Brandice Canes-Wrone, and Sharece Thrower. "Ideologically sophisticated donors: Which candidates do individual contributors finance?" American Journal of Political Science 61.2 (2017): 271-288

```
load("donationdata.RData")
```

#### Variables

- **donation:** 1=made donation to senator, 0=no donation made
- **total\_donation:** Dollar amount of donation made by donor to Senator
- **sameparty:** 1=self-identifies as being in the candidate's party; 0 otherwise
- **NetWorth:** Donor's net worth. 1=less than 250k, 2=250-500k; 3=500k-1m; 4=1-2.5m; 5=2.5-5m; 6=5-10m; 7=more than 10m
- **IncomeLastYear:** Donor's household annual income in 2013. 1=less than 50k; 2=50-100k; 3=100-125k; 4=125-150k; 5=150-250k; 6=250-300k; 7=300-350k; 8=350-400k; 9=400-500k; 10=more than 500k
- **peragsen:** percent issue agreement between donor and senator
- **per2agchal:** percent issue agreement between donor and the senator's challenger
- **cook:** Cook competitiveness score for the senator's race. 1 = Solid Dem or Solid Rep; 2 = Likely
- **matchcommf:** 1=Senator committee matches donor's profession as reported in FEC file; 0=otherwise
- **Edsum:** Donor's self-described educational attainment. 1=less than high school; 2=high school; 3=some college; 4=2-year college degree; 5=4-year college degree; 6=graduate degree

Data represent information on past donors to campaigns across different states. The key dependent variable that we want to predict is **total\_donation**: the total dollar amount a particular person in the data gave to their senator in the 2012 election campaign.

Can we predict how much someone donates to a U.S. Senate campaign?

1. Choose approach: regression of donations on donor characteristics
2. Check accuracy: calculate root-mean-squared error
3. Iterate: try different regression model specifications

Let's try a prediction based on a person's income.

```
fit <- lm(total_donation ~ IncomeLastYear, data = donationdata)
```

From this, we can

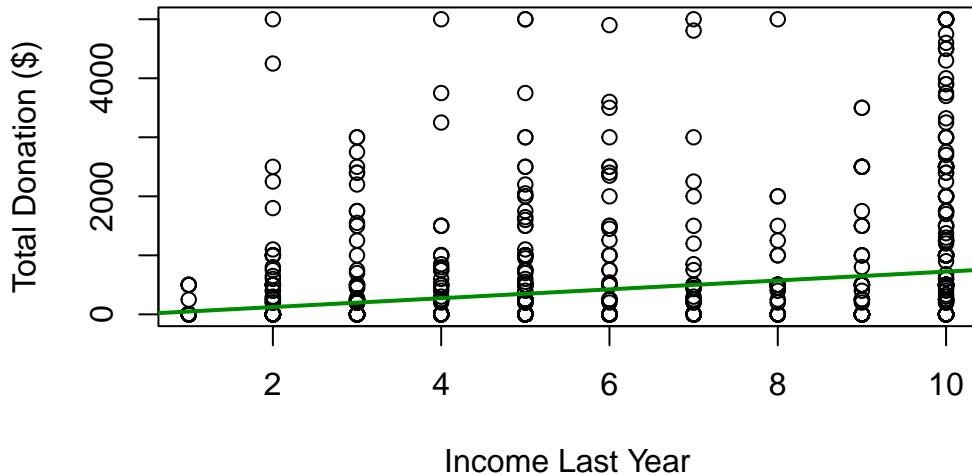
- Plot the relationship
- Make specific predictions at different levels of income
- Check accuracy by calculating the prediction errors and RMSE

### 8.6.1 Visualizing the results

Note that the correlation is a bit weaker here.

```
plot(x=donationdata$IncomeLastYear,
 y=donationdata$total_donation,
 ylab= "Total Donation ($)",
 xlab = "Income Last Year",
 main = "Predicting Total Donations Using Income")
abline(fit, col="green4", lwd=2)
```

## Predicting Total Donations Using Income



### 8.6.2 Step 1: Calculate Predictions

We can calculate predictions based on a level of income. Example: Level 5 of income represents an income of \$150k-250k. What level of donation would we expect?

```
predict(fit, data.frame(IncomeLastYear = 5))
```

```
1
348.8581
```

```
alternative using coef()
coef(fit)[1] + coef(fit)["IncomeLastYear"]*5
```

```
(Intercept)
348.8581
```

### 8.6.3 Step 2: Check Accuracy

We can calculate the Root Mean Squared Error

```
sqrt(mean(fit$residuals^2))
```

```
[1] 914.5273
```

#### 8.6.4 Step 3: Iterate

YOUR TURN: Change the model and see if it improves the prediction using RMSE using `sqrt(mean(fit$residuals^2))`.

#### 8.6.5 Adding Model Predictors

New Model Example

```
fitnew <- lm(total_donation ~ IncomeLastYear + NetWorth + sameparty,
 data=donationdata)
```

New Predictions: note how we add more variables

```
predict(fitnew, data.frame(IncomeLastYear = 5, NetWorth = 4, sameparty = 1))
```

```
1
406.9705
```

```
alternative using coef()
coef(fitnew)[1] + coef(fitnew)["IncomeLastYear"]*5 +
 coef(fitnew)["NetWorth"]*4 + coef(fitnew)["sameparty"]*1
```

```
(Intercept)
406.9705
```

Root Mean Squared Error

```
sqrt(mean(fitnew$residuals^2))
```

```
[1] 908.9817
```

When we have multiple predictors, this changes our interpretation of the coefficients slightly.

- We now interpret the slope as the change in the outcome expected with a 1-unit change in the independent variable— holding all other variables constant (or “controlling” for all other variables)
- For example, for a 1-unit change in Income, we would expect about a \$68 increase in estimated donations, holding constant Net Worth and whether the person shared partisanship with the senator.

```
coef(fitnew)
```

(Intercept)	IncomeLastYear	NetWorth	sameparty
-242.02780	67.96825	29.55847	190.92323

Think of this like a set of light switches. How does adjusting one light switch affect the light in the room— holding constant all other switches.



When we make predictions with multiple variables, we have to tell R where we want to set each variable’s value.

```
predict(fitnew, data.frame(IncomeLastYear = 5, NetWorth = 4, sameparty = 1))
```

```
1
406.9705
```

See how the prediction changes if you shift `IncomeLastYear` but keep Net Worth and partisanship where they are. That’s the idea of “controlling” for the other variables!

How could we keep improving the predictions?

Eventually, we would want to apply this prediction model in a real-world setting.

- How could campaigns use these types of prediction models?

## 8.7 Uncertainty with Prediction

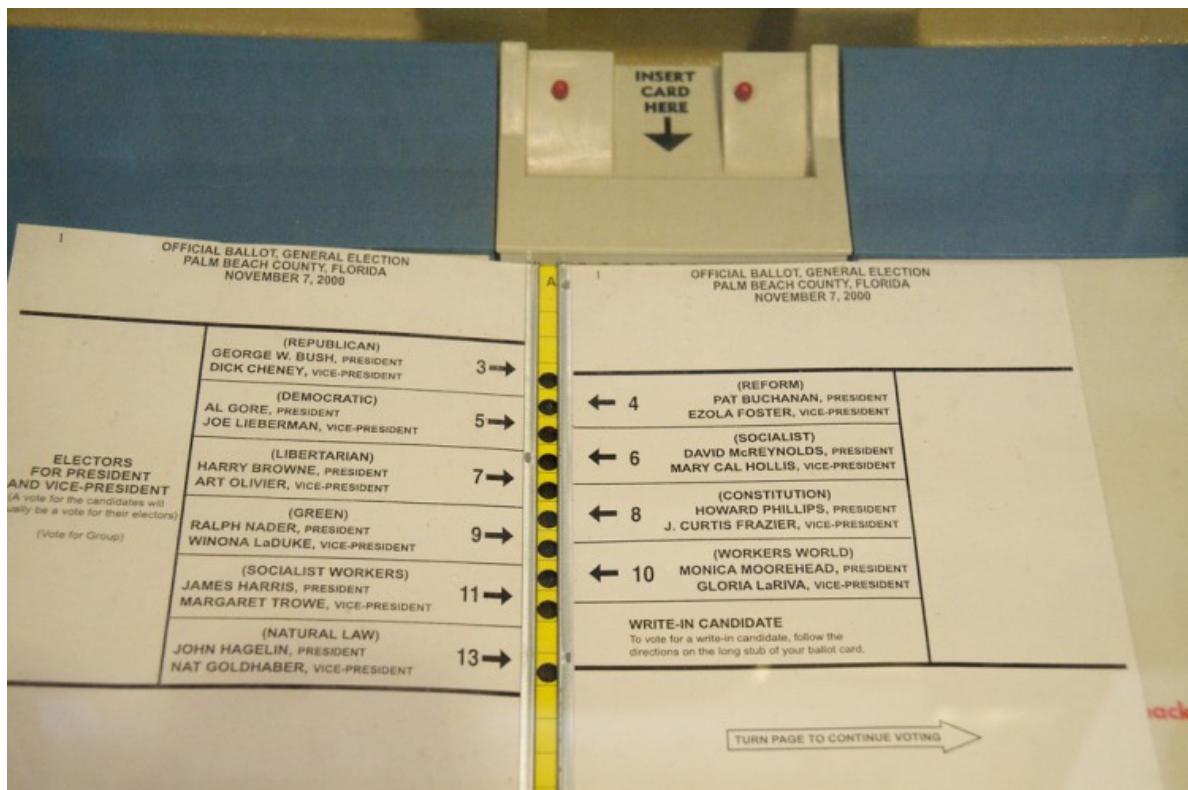
Regression (and other prediction algorithms) give us our best guess

- But any guess has some uncertainty, prediction error, and potential outliers
- Sometimes these errors can be systematic
- Even when we use more advanced statistical models
- A “best guess” is often better than a random guess— but shouldn’t necessarily be treated as “ground truth.”

Prediction helps us guess unknowns with observed data, but MUST PROCEED WITH CAUTION

### 8.7.1 Example: Butterfly Ballot in Florida

In the U.S. 2000 presidential election, the race came down to Florida, which was extremely close. As part of the contest, different counties in Florida came under a microscope. One result that seemed unusual was the amount of votes Buchanan received in certain areas, which seemed to be a result of an odd ballot design choice. In this exercise, we examine voting patterns in Florida during the 2000 election.



For more on the 2000 race, you can watch this [video](#).

Load the data and explore the variables

- 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

```
florida <- read.csv("florida.csv")
```

Chapter 4 in QSS also discusses this example.

Using what you learned from the last section, try to complete the following steps:

- Regress Buchanan 2000 votes (your Y) on Perot 1996 (your X) votes
- Create a scatterplot of the two variables and add the regression line
- Find and interpret the slope coefficient for the relationship between Perot and Buchanan votes
- Calculate the root-mean-squared error for the regression and interpret this

Try on your own, then expand for the solution.

For every 1 additional vote Perot received in 1996, we expect Buchanan to receive .036 additional votes in 2000.

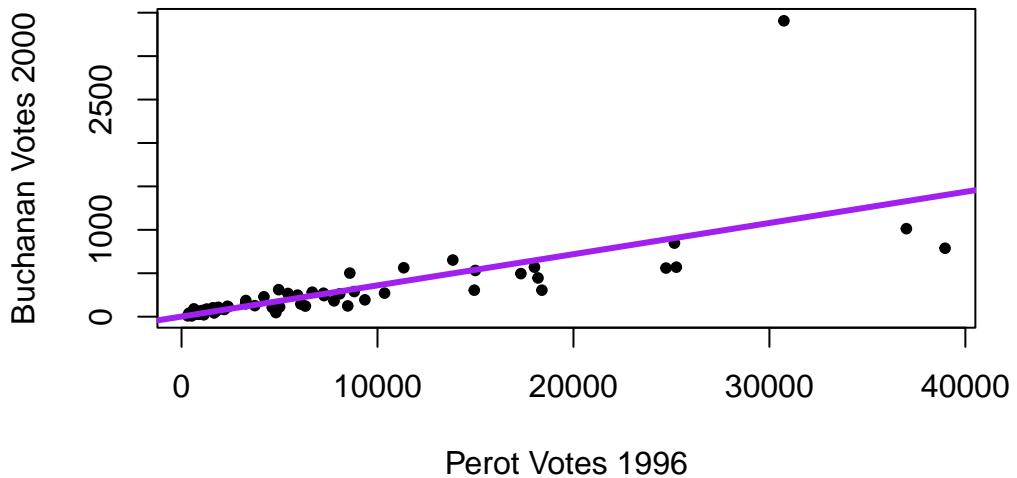
```
fit <- lm(Buchanan00 ~ Perot96, data = florida)
coef(fit)

(Intercept) Perot96
1.34575212 0.03591504
```

In 1996, Perot received 8 million votes as a third-party candidate. Buchanan received less than 1/2 a million. Overall Perot received more votes, but where Perot received votes in 1996 was positively correlated with where Buchanan received votes in 2000.

```
plot(x=florida$Perot96,
 y=florida$Buchanan00,
 ylab="Buchanan Votes 2000",
 xlab="Perot Votes 1996",
 pch=20)
```

```
abline(fit, lwd=3, col="purple")
```



```
sqrt(mean(fit$residuals^2))
```

```
[1] 311.6187
```

A typical prediction error is about 316.4 votes above or below the Buchanan total.

### 8.7.2 Multiple Predictors

Can we reduce the error by adding more variables?

```
fitnew <- lm(Buchanan00 ~ Perot96 + Dole96 + Clinton96, data = florida)
coef(fitnew)
```

	(Intercept)	Perot96	Dole96	Clinton96
20.572650070	0.030663207	-0.001559196	0.001865809	

Again, when we have multiple predictors, this changes our interpretation of the coefficients slightly.

- We now interpret the slope as the change in the outcome expected with a 1-unit change in the independent variable— holding all other variables constant (or “controlling” for all other variables)
- For example, a 1-unit increase (a 1-vote increase) in the number of Perot voters in 1996 is associated with a 0.03 vote increase in the number of Buchanan votes in 2000, holding constant the number of Clinton and Dole votes a county received.

When we make predictions with multiple variables, we have to tell R where we want to set each variable’s value.

```
predict(fitnew, data.frame(Perot96=20000, Clinton96=300000, Dole96=300000))
```

```
1
725.8208
```

See how the prediction changes if you shift Perot96 but keep the other variables where they are. That’s the idea of “controlling” for the other variables!

The addition of the new variables, in this case, made very little difference in the RMSE.

```
sqrt(mean(fit$residuals^2))
```

```
[1] 311.6187
```

```
sqrt(mean(fitnew$residuals^2))
```

```
[1] 308.7296
```

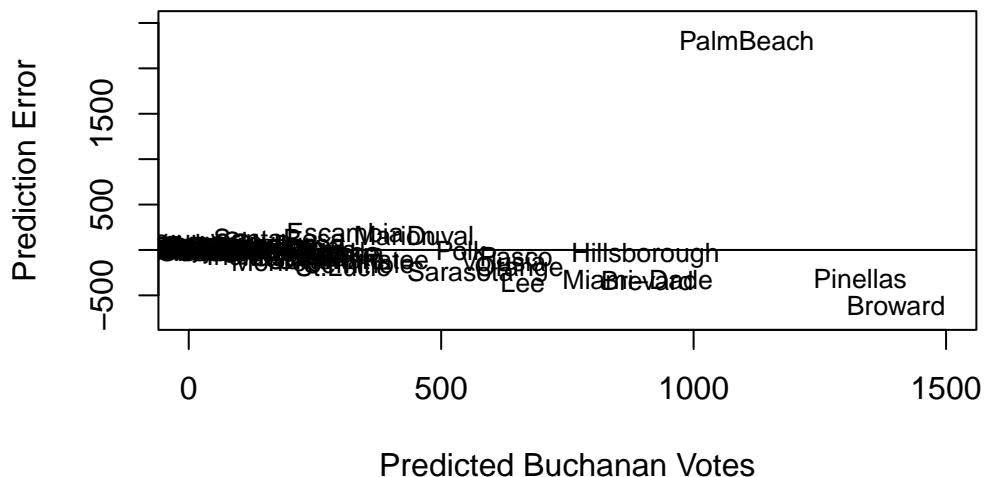
With little change from the addition of predictors, let’s stick with the more simple model and explore the prediction errors.

```
plot(x=fitted(fit), # predicted outcome
 y=resid(fit), # prediction error
 type="n", # makes the plot blank
 xlim = c(0, 1500),
 ylim = c(-750, 2500),
 xlab = "Predicted Buchanan Votes",
```

```

ylab = "Prediction Error")
abline(h = 0) # adds horizontal line
text(x=fitted(fit), y=resid(fit), labels = florida$county, cex=.8)

```



How does the prediction error change if we remove Palm Beach County?

```

florida.pb <- subset(florida, subset = (county != "PalmBeach"))
fit2 <- lm(Buchanan00 ~ Perot96, data = florida.pb)
sqrt(mean(fit2$residuals^2))

```

[1] 86.41017

My, oh my, our RMSE also goes way down if we remove Palm Beach. Something unique seems to be happening in that county. See this [academic paper](#) for an elaboration of the evidence that “The Butterfly [ballot] Did it.”

### 8.7.3 Confidence Intervals

Social scientists like to characterize the uncertainty in their predictions using what is called a “confidence interval.”

- Confidence intervals show a range of values that are likely to contain the true value

```
predict(fit, data.frame(Perot96 = 13600), interval = "confidence")
```

```
fit lwr upr
1 489.7903 394.8363 584.7443
```

By default, R supplies the 95% confidence interval.

- For example, our estimate is for a county with 13,600 votes for Perot in 1996, the expected Buchanan vote is 489.79 votes.
  - The confidence interval is 394.84 to 584.74 votes, which means we believe there is a 95% chance that this interval contains the true value of the Buchanan 2000 vote share.
- Instead of thinking about our prediction as just 489.79, we should think about the entire interval as having a good chance of including the true value.

Similarly, our coefficients also have uncertainty.

```
coef(fit)
```

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

```
confint(fit)
```

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

For every 1 vote increase in the Perot 1996 vote, we expect a  $\hat{\beta} = .036$  increase in Buchanan votes. However, the confidence interval is 0.027 to 0.045.

- We think there is a 95% chance that this interval 0.027 to 0.045 includes the true  $\beta$ , describing the rate of change in Buchanan votes for a given change in Perot 1996 votes

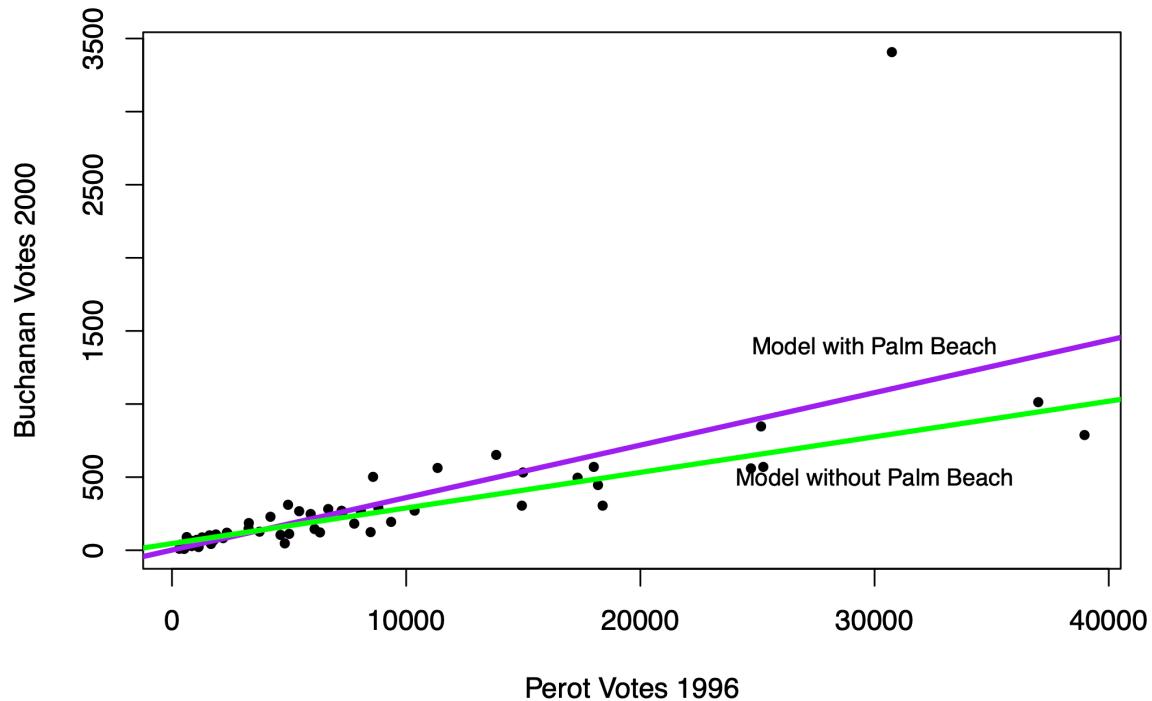
## 8.8 Cross-Validation

So far, we've been "cheating." We've been analyzing a prediction in cases where we know the right answer. Now we will focus more squarely on how to develop an "out-of-sample" prediction.

Problem: Models that fit our existing ("in-sample") data might not be the best for predicting out-of-sample data.

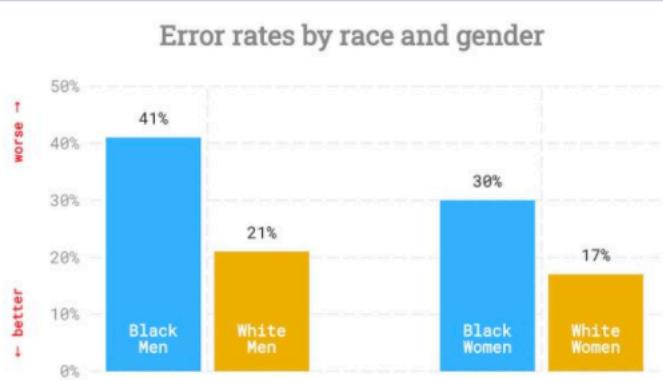
Example: Compare regression line with vs. without Palm Beach included in the sample.

- Outliers "in-sample" can lead to overfitting to weird, idiosyncratic data points



Example: Error Rates in Speech Recognition. See study [here](#)

- Predictions/Classification that works well for one group might not work well for all groups



### The Race Gap in Speech Recognition

The leading speech recognition tools misunderstand black speakers twice as often as whites.

[fairspeech.stanford.edu](http://fairspeech.stanford.edu)

Problem: Models that fit our existing (“in-sample”) data might not be the best for predicting out-of-sample data. Approaches to diagnose the problem or help address it:

- Detect potential outliers within existing data by exploring the prediction errors
- Make sure the training (in-sample) data is as representative as possible
- Incorporate out-of-sample testing in prediction process

### 8.8.1 Cross-Validation Process

Cross-validation incorporates the idea of out-of-sample testing into the process of how we evaluate the accuracy of prediction and classification approaches.

Cross-validation (train vs. test data)

1. Subset your data into two portions: Training and Test data.
2. Run a model based on the training data.
3. Make a prediction and test the accuracy on the test data.
4. Repeat process training and testing on different portions of the data.
5. Summarize the results and choose a preferred model
  - Eventually: Apply this model to entirely new data

Goal: Test accuracy in a way that can help detect overfitting. See how well our model will generalize to new data (data the model hasn't seen).

## 8.8.2 Application: Forecasting Election Results

Macro political and economic fundamentals are sometimes used for early forecasting of an election. We will build a version of this model and test its accuracy using a process of “Leave-one-out” cross-validation.

Below is a video explainer of this application, which uses cross-validation.

<https://www.youtube.com/watch?v=Wqj9Jm8heAA>

The data and model are based on the [FAIR model](#) of forecasting.

```
fair <- read.csv("fair.csv")
```

Key Variables:

- VP: Democratic share of the two-party presidential vote
- t: Year (presidential years only)
- G: growth rate of real per capita GDP in the first 3 quarters
- P: growth rate of the GDP deflator in the first 15 quarters of the recent administration
- Z: number of quarters in the first 15 quarters of recent administration in which the growth rate of real per capita GDP is greater than 3.2 percent at an annual rate
- I: 1 if Democrats in WH and -1 if Republicans in WH
- WAR: 1 if 1920, 1944, 1948 (denoting the “WAR” elections, which are believed to be particular)
- DUR: indicating how many consecutive terms Democrats/Republicans have been office (e.g., in 2020 it will be 0 because Republicans will have been in office for only 1 term.)

Let's propose a model

```
fit <- lm(VP ~ DUR, data = fair)
```

Let's propose an alternative model and see which one we think is better.

```
fit2 <- lm(VP ~ G*I + DUR, data = fair)
```

Note: The asterisk represents an “interaction.” See QSS Chapter 4. We use this when we think the effect of one variable (growth) may depend on the values of another variable (the party of who is in office).

### 8.8.2.1 Steps 1 and 2

We are going to run a model where each time we ‘leave out’ one row of data (in our case, one election). Let's try this once:

```

years <- fair$t

Step 1: Subset data into two portions
traindata <- subset(fair, t != years[1])
testdata <- subset(fair, t == years[1])

Step 2: Run model on training data
fit <- lm(VP ~ DUR, data = traindata)
fit2 <- lm(VP ~ G*I + DUR, data = traindata)

```

### 8.8.2.2 Step 3: Predict and assess accuracy with test data

Out-of-Sample prediction

```

Step 3: Make a Prediction using test data and
yhat.fit <- predict(fit, testdata)
yhat.fit2 <- predict(fit2, testdata)

```

Prediction error (Truth - Prediction)

```

Step 3: Test accuracy of prediction
error.fit <- testdata$VP - yhat.fit
error.fit2 <- testdata$VP - yhat.fit2

```

### 8.8.2.3 Step 4: Repeat process across all data

Step 4: Let's do this for each row, storing the prediction errors.

```

Iteration vector
years <- fair$t
Empty container vectors
errors.fit <- rep(NA, length(years))
errors.fit2 <- rep(NA, length(years))

Loop (copy paste meat from above)
for(i in 1:length(years)){
 traindata <- subset(fair, t != years[i])
 testdata <- subset(fair, t == years[i])
 fit <- lm(VP ~ DUR, data = traindata)

```

```

fit2 <- lm(VP ~ G*I + DUR , data = traindata)
yhat.fit <- predict(fit, testdata)
yhat.fit2 <- predict(fit2, testdata)
errors.fit[i] <-testdata$VP - yhat.fit
errors.fit2[i] <-testdata$VP - yhat.fit2
}

```

#### 8.8.2.4 Step 5: Summarize performance

Step 5: Summarize the model performance

```

RMSE
sqrt(mean((errors.fit)^2))

```

[1] 7.170149

```

sqrt(mean((errors.fit2)^2))

```

[1] 3.793135

```

Mean Absolute Error
mean(abs(errors.fit))

```

[1] 5.937542

```

mean(abs(errors.fit2))

```

[1] 3.363163

Which model tends to have less error?

#### 8.8.2.5 Applying Model to New Data

Eventually, you might further test the model on data that has been “held out”— data that neither your train/test has seen. How good was our model? We can do this for the 2016 election, which was not in the data.

Truth: 2016 VP was 51.1 Democratic “two-party” vote share.

```
Let's use the winner of our two models
fit2 <- lm(VP ~ G*I + DUR, data = fair)
51.1-predict(fit2, data.frame(G=0.97, I=1, DUR=1))
```

```
1
2.472147
```

2016 values based on the FAIR [site](#)

### 8.8.2.6 Challenge

Can you build a better model? What would your prediction for 2020 have been?

- -5.07: growth rate of real per capita GDP in the first 3 quarters of 2020 (annual rate) (G)
- 1.80: growth rate of the GDP deflator in the first 15 quarters of the Trump administration, (annual rate) (P)
- 3: number of quarters in the first 15 quarters of the Trump administration in which the growth rate of real per capita GDP is greater than 3.2 percent at an annual rate (Z)
- DUR=0
- I = -1

Values based on the FAIR [site](#)