Gov 50: 1. Introduction

Matthew Blackwell

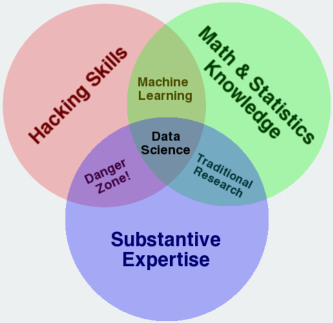
Harvard University

Roadmap

1. Welcome and Motivation
2. Course Details

1/ Welcome and Motivation

What is data science?



- **Data science**: wrangling, visualizing, and analyzing data to understand the world
- Who does data science? Tech companies, non-tech companies, nonprofits, governments.

Credit: Drew Conway

Glassdoor's No. 3 best job in the U.S. has seen job growth surge 480%

BY **MEGHAN MALAS**

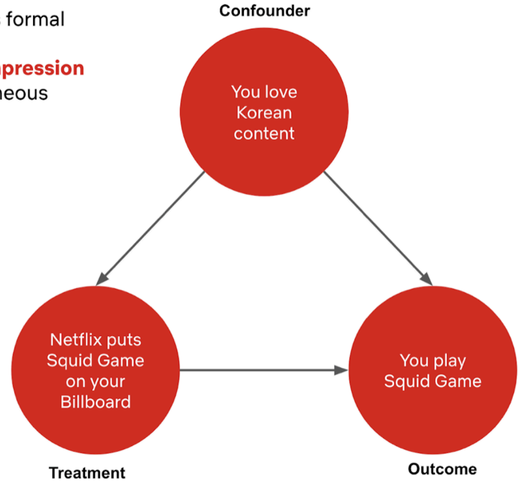
March 08, 2022, 1:12 PM

A COMMUTER BOARDS A BAY AREA RAPID TRANSIT (BART) TRAIN IN THE NEW MONTGOMERY STATION IN SAN FRANCISCO, CALIFORNIA, AS SEEN IN MARCH 2022. (PHOTOGRAPHER: DAVID PAUL MORRIS—BLOOMBERG/GETTY IMAGES)

What problems are data scientists working on?

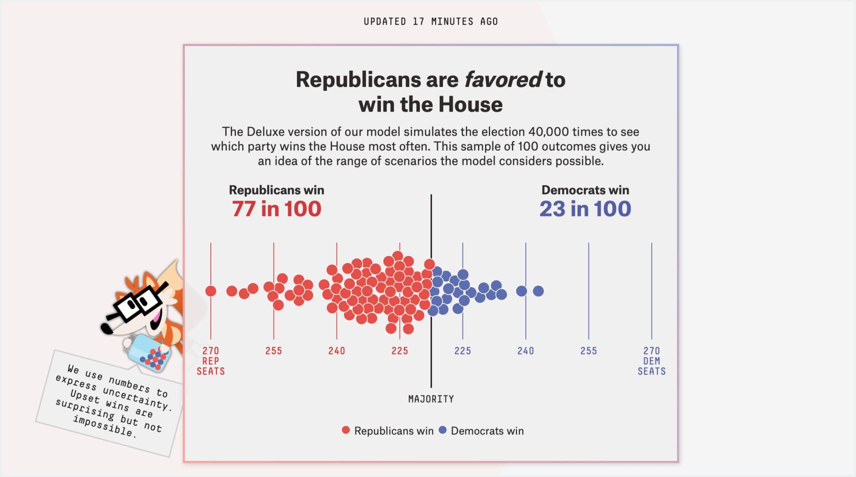
Causality

Causal Inference provides formal tools to tease out the true **incremental** value of an **impression** for each profile: Heterogeneous Treatment Effect (**HTE**)



Compared to machine learning, causal inference allows us to build a robust framework that controls for confounders in order to estimate the true incremental impact to members

Prediction



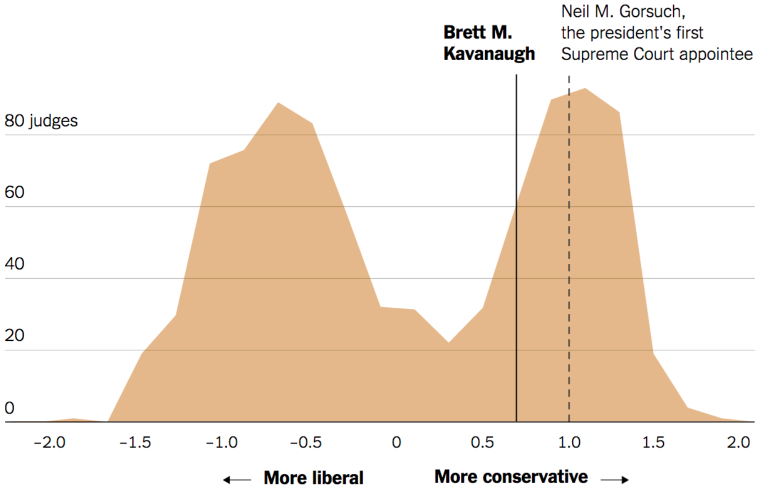
UPDATED 17 MINUTES AGO

Republicans are *favored* to win the House

The Deluxe version of our model simulates the election 40,000 times to see which party wins the House most often. This sample of 100 outcomes gives you an idea of the range of scenarios the model considers possible.

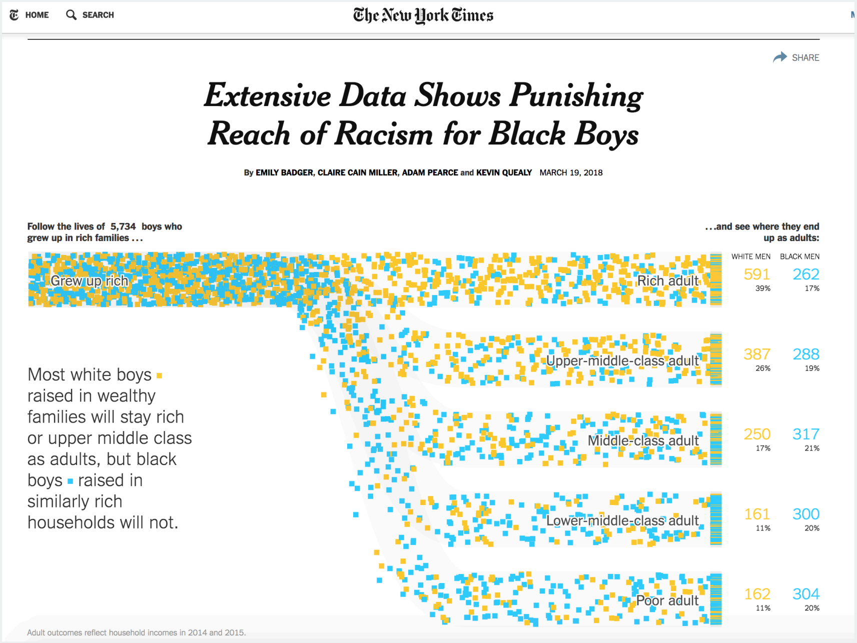
Measurement

How Kavanaugh's Ideology Compares With Other Federal Judges



Based on the campaign finance scores of all current and former federal district and court of appeals judges nominated since 1980. | Source: Database on Ideology, Money in Politics, and Elections; Adam Bonica, Stanford University Department of Political Science; Maya Sen, Harvard University, Kennedy School of Government; Adam Chilton and Kyle Rozema, University of Chicago Law School.

Understanding the socioeconomic world



HOME SEARCH The New York Times SHARE

Extensive Data Shows Punishing Reach of Racism for Black Boys

By **EMILY BADGER, CLAIRE CAIN MILLER, ADAM PEARCE** and **KEVIN QUEALY** MARCH 19, 2018

Follow the lives of 5,734 boys who grew up in rich families . . .

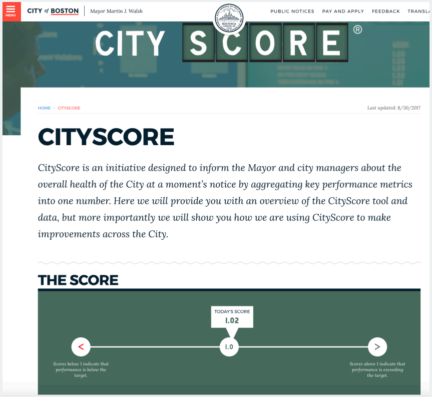
. . .and see where they end up as adults:

	WHITE MEN		BLACK MEN	
Rich adult	591	39%	262	17%
Upper-middle-class adult	387	26%	288	19%
Middle-class adult	250	17%	317	21%
Lower-middle-class adult	161	11%	300	20%
Poor adult	162	11%	304	20%

Most white boys ▪ raised in wealthy families will stay rich or upper middle class as adults, but black boys ▪ raised in similarly rich households will not.

Adult outcomes reflect household incomes in 2014 and 2015.

Making government work better



CITY of BOSTON | Mayor Martin J. Walsh

PUBLIC NOTICES PAY AND APPLY FEEDBACK TRANSL

CITY SCORE®

HOME › CITYSCORE

Last updated: 8/30/2017

CITYSCORE

CityScore is an initiative designed to inform the Mayor and city managers about the overall health of the City at a moment's notice by aggregating key performance metrics into one number. Here we will provide you with an overview of the CityScore tool and data, but more importantly we will show you how we are using CityScore to make improvements across the City.

THE SCORE

TODAY'S SCORE 1.02

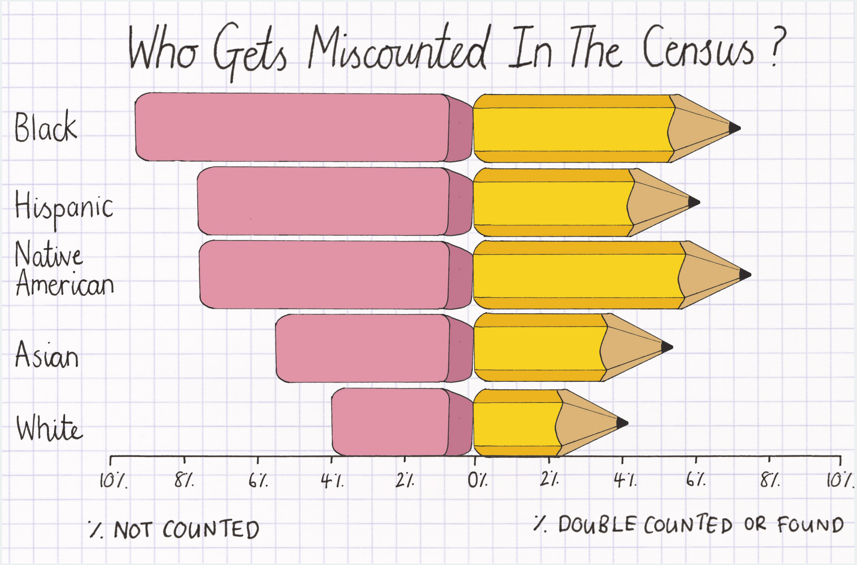
1.0

Scores below 1 indicate that performance is below the target.

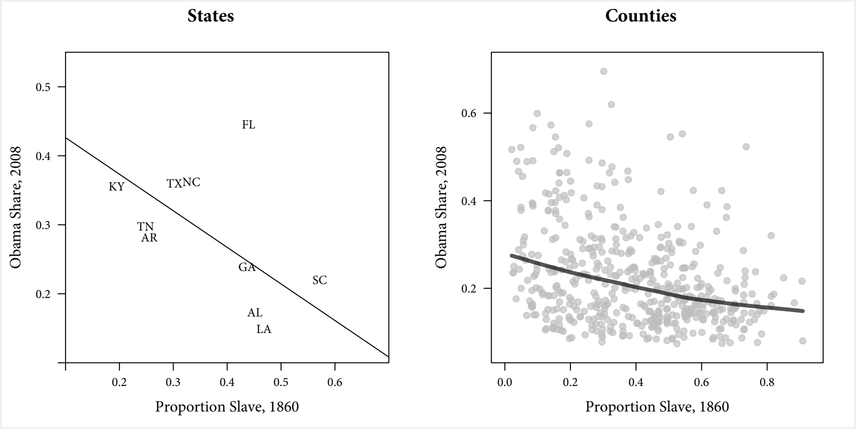
Scores above 1 indicate that performance is exceeding the target.

Topic	Day	Week	Month	QTR
311 CALL CENTER PERFORMANCE			0.94	0.93
CODE ENFORCEMENT ON-TIME %	1.23	1.24	1.24	1.24
CODE ENFORCEMENT TRASH COLLECTION	1.25	1.18	1.15	1.10
GRAFFITI ON-TIME %	0.33	0.13	0.20	0.27
MISSED TRASH ON-TIME %	1.21	1.20	1.20	1.20
PARKS MAINTENANCE ON-TIME %	0.82	0.83	0.90	0.88
POTHOLE ON-TIME %	1.25	0.88	0.69	0.67
SIGN INSTALLATION ON-TIME %	1.00	0.23	0.24	0.49
SIGNAL REPAIR ON-TIME %	1.25	1.25	1.10	1.09
STREETLIGHT ON-TIME %	0.55	0.60	0.56	0.54
TREE MAINTENANCE ON-TIME %	1.19	1.18	1.17	1.13
ON-TIME PERMIT REVIEWS	0.88	0.88	0.85	0.81
LIBRARY USERS	1.51	1.42	1.43	1.42
BPS ATTENDANCE				0.90
BFD RESPONSE TIME	0.91	0.94	0.94	0.94
BFD INCIDENTS	1.03	0.92	0.93	0.94
EMS RESPONSE TIME	0.90	0.84	0.84	0.84
PART 1 CRIMES	2.26	1.48	1.41	1.40

Combining art and data to inform



Understanding how the past matters



2/ Course Details

About me

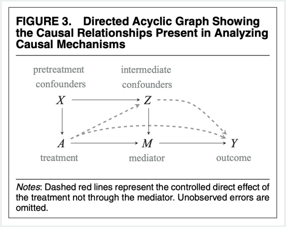
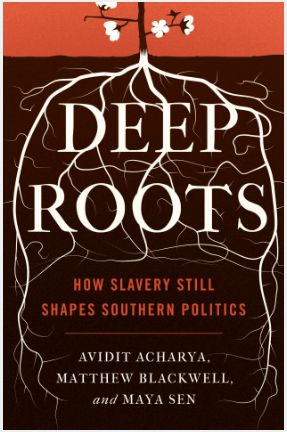


FIGURE 3. Directed Acyclic Graph Showing the Causal Relationships Present in Analyzing Causal Mechanisms

$$\begin{tikzcd}
\text{pretreatment confounders} & \text{intermediate confounders} & \\
X \arrow[r] \arrow[d] & Z \arrow[d] \arrow[dr, dashed] & \\
A \arrow[r] \arrow[ur, dashed] \arrow[rr, dashed, bend left] & M \arrow[r] & Y \\
\text{treatment} & \text{mediator} & \text{outcome}
\end{tikzcd}$$

Notes: Dashed red lines represent the controlled direct effect of the treatment not through the mediator. Unobserved errors are omitted.


DEEP ROOTS
HOW SLAVERY STILL SHAPES SOUTHERN POLITICS
AVIDIT ACHARYA, MATTHEW BLACKWELL, and MAYA SEN

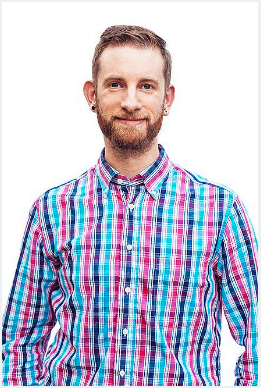
What will you learn in this class?

- Summarize and visualize data
- Wrangle messy data into tidy forms
- Evaluate claims about causality
- Be able to use linear regression to analyze data
- Understand uncertainty in data analysis and how to quantify it
- Use professional tools like R, RStudio, git, and GitHub

Teaching philosophy

- Deliberate pacing and tons of support.
- Emphasize intuition and computational approaches over mathematical equations.
- Practice, practice, practice.

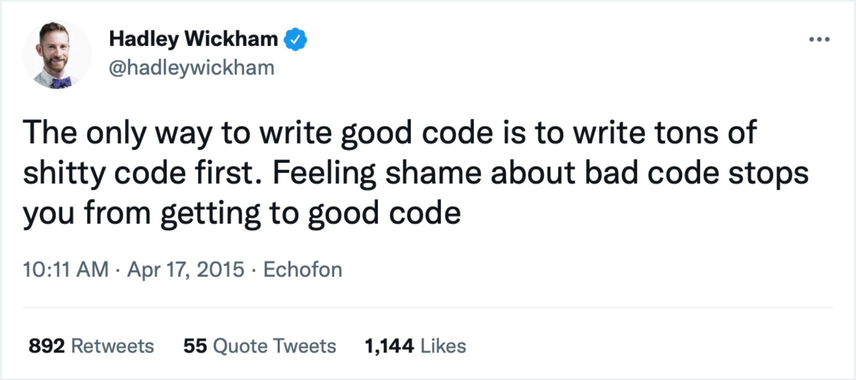
Pep talk, part I



Hadley Wickham (chief data scientist at RStudio)

It's easy when you start out programming to get really frustrated and think, "Oh it's me, I'm really stupid," or, "I'm not made out to program." But, that is absolutely not the case. Everyone gets frustrated. I still get frustrated occasionally when writing R code. It's just a natural part of programming. So, it happens to everyone and gets less and less over time. Don't blame yourself. Just take a break, do something fun, and then come back and try again later.

Pep talk, part II



Hadley Wickham
@hadleywickham

The only way to write good code is to write tons of shitty code first. Feeling shame about bad code stops you from getting to good code

10:11 AM · Apr 17, 2015 · Echofon

892 Retweets 55 Quote Tweets 1,144 Likes

Should I take this course?

- Prerequisites: **NONE** (no prior coding, statistics, data science)
- Gov 50 fulfills Gov methods requirement, data science track, and QRD
- Material useful to students interested in political science, sociology, economics, public policy, health policy, and many other fields in the social sciences.

Class meetings

- Lectures:
    - Broad coverage of the course material.
    - Coding demonstrations (follow along with your laptop!)
    - Slides/videos will be posted to Canvas shortly before lecture.
- Section:
    - Guided practice through problems and concepts led by our amazing TFs.
    - Material in section will closely mirror assignments.
- Optional speaker series with industry data scientists, TBA!

Teaching fellows

Angelo Dagonel

Dorothy Manevich

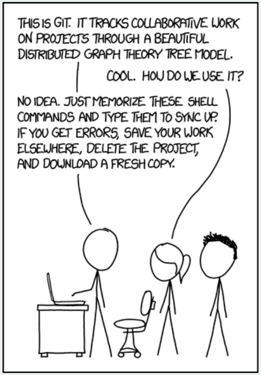
Sooahn Shin

Dominic Valentino

Computing

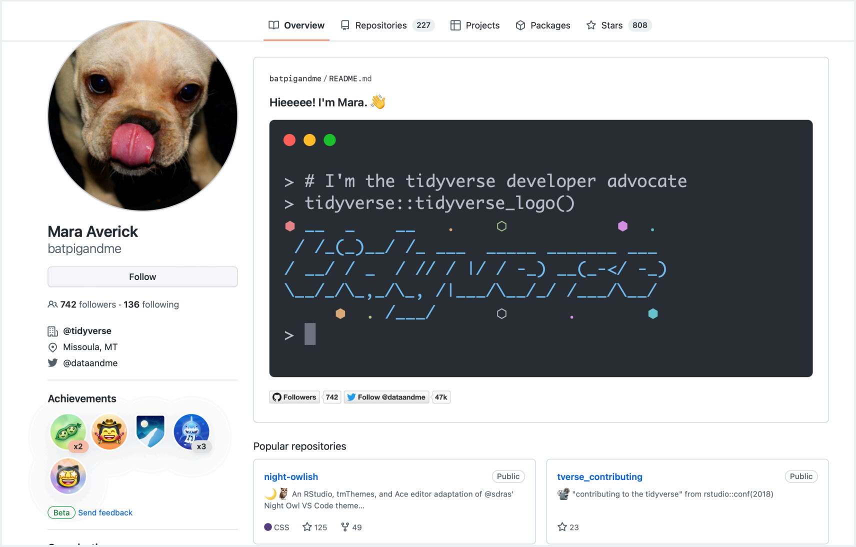
- We'll use the R statistical environment to analyze data
    - It's free
    - Extremely popular for data analysis
    - Academics, 538, NYT, Facebook, Google, Twitter, nonprofits, governments all use R.
    - Huge benefit to your resume to have R skills.
- Interface with R via a program called RStudio
- Problem Set 0 on the website helps get everything installed.
- Lots of help in section, study halls, office hours.

git and GitHub


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

- Other core tools: git and GitHub
    - **Version control system**: an archive of project versions.
    - Allows you to revert back to old versions easily
    - Makes collaboration much more mangeable.
- Will feel very odd at first, but you git used to it
- Why learn this now?
    - Knowing git/GitHub is a huge plus for data jobs.
    - Your GitHub profile can showcase your amazing new skills with data!

Sample GitHub profile



Overview Repositories 227 Projects Packages Stars 808

Mara Averick
batpigandme

Follow

742 followers · 136 following

@tidyverse
Missoula, MT
@dataandme

Achievements

x2 x3

Beta Send feedback

batpigandme / README.md

Hieeeee! I'm Mara. 👋

```
> # I'm the tidyverse developer advocate
> tidyverse::tidyverse_logo()
⬢ __  _    __   .    ⬡           ⬢  .
 / /_(_)__/ /_ ___  ____ _______ ___
/ __/ / _  / // / |/ / -_) __(_-</ -_)
\__/_/\_,_/\_, /|___/\__/_/ /___/\__/
     ⬢  . /___/      ⬡      .       ⬢
> 
```

Followers 742 Follow @dataandme 47k

Popular repositories

night-owlish Public
🌙🦉 An RStudio, tmThemes, and Ace editor adaptation of @sdras' Night Owl VS Code theme...
● CSS ☆ 125 49

tverse_contributing Public
"contributing to the tidyverse" from rstudio::conf(2018)
☆ 23

GovCodes workshops

- Gov department providing supplemental GovCodes workshops to provide additional computing practice.
- First meeting: tomorrow! Be on the lookout for a sign-up email.
    - Topic: getting everything installed and working on your computer!
    - Good to attend if Problem Set 0 is giving you trouble.

Textbook

- 3 primary textbooks (links on syllabus):
  - Modern Dive (free online)
  - “Quantitative Social Science: An Introduction in tidyverse” by Kosuke Imai (not free)
  - Introduction to Modern Statistics (free online)
- We’ll move back and forth.
- Sometimes same material in two/three different books. Choose which helps most!

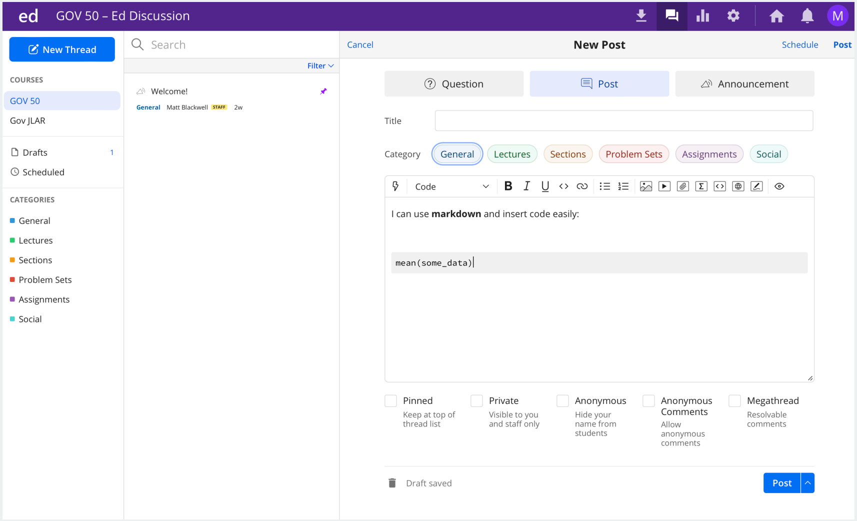
Assignments

- Roughly weekly homeworks throughout semester
    - Posted on Thursday morning, due following Wednesday.
    - Dates on syllabus
    - Lowest score dropped.
- Two take-home “exams” which are just HWs done by yourself.
- Final project: a data essay
    - Find data, pose a research question, answer it using data.
    - Submitted as a public GitHub repository and website
    - First item in your public data portfolio

Tutorials

- Getting practice with R can be overwhelming, so we'll introduce new skills through online tutorials.
- Guided practice on R, helping to introduce new concepts.
    - Low stakes/stress: graded simply on completion.
    - Due on Monday nights
- Lecture/HW won't be the first time you're trying some code!

Ed discussion board



I can use **markdown** and insert code easily:

```
mean(some_data)
```

Grades

- Grade breakdown as follows:
  - R tutorials (10% of final grade)
  - Homeworks (40% of final grade)
  - Exams (30% of final grade)
  - Final project (20% of final grade)
- Final grade is curved
- **Bump-up**: we bump up grades of students close to the cutoff who make valuable contributions to the course.

Study Halls

- Study Halls: a place to work on Gov 50 and get help.
    - Will happen weekly, exact number of hours will depend on enrollment.
    - Peer tutors with experience in statistics and R will be on hand to help you if you get stuck or have question.
    - Best to come in groups and work together, grab a tutor when stuck.
- Bottom line: **we want you to succeed in this class!**

What should you do today?

- Try to get everything set up on your computer (Problem Set 0)
- Start Tutorial 1 on basics of R and data visualization
    - Can be done on the web before installing R on your computer.
- Respond to sign-up requests for GovCodes and section times.
- **Tell your friends**: data science is more fun with friends along for the journey.

Gov 50: 2. R, RStudio, and Rmarkdown

Matthew Blackwell

Harvard University

Roadmap

1. Working in Plain Text
2. Let’s take a touR
3. Using Rmarkdown
4. Getting R bearings
5. Our first visualizations

1/ Working in Plain Text

The two computer revolutions

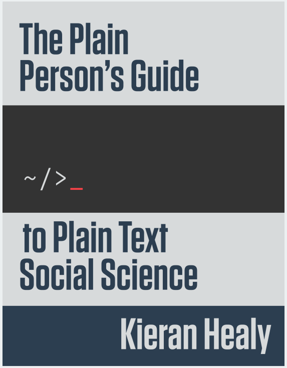
The frontier of computing

- Touch-based interfaces
- Single app at a time
- Little multitasking between apps
- Hides the file system

Where statistical computing lives

- Windows and pointers
- Multi-tasking, multiple windows
- Works heavily with the file system
- Underneath it's UNIX and the command line

Plain-text tools for data analysis



- Often free, open-sourced, and powerful.
- Large, friendly communities around them.
- Tons of resources
- But… far from the touch-based paradigm of modern computing
- So why use them?

The process of data science is instrinsically messy

Office vs engineering model of computing

What's real in the project? How are changes managed?

In the Office model

- Formatted documents are real.
- Intermediate ouptuts copy/pasted into documents.
- Changes are tracked inside files.
- Final output is the file you are working on (e.g., Word file or maybe converted to a PDF).

In the Engineering model

- Plain-text files are real.
- Intermediate outputs are produced via code, often inside documents.
- Changes are tracked outside files.
- Final outputs are assembled programatically and converted to desired output format.

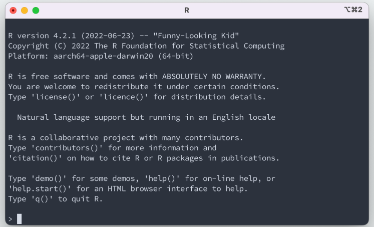
Pros and cons to each approach

- Office model:
  - Everyone knows Word, Excel, Google Docs.
  - “Track changes” is powerful and easy.
  - Wait, how did I make this figure?
  - Which version of my code made this table?
  - `Blackwell_report_final_submitted_edits_FINAL_v2.docx`
- Engineering model:
  - Plain text is universally portable.
  - Push button, recreate analysis.
  - Why won’t R just do what I want!
  - Version control is a pain.
  - `Object of type 'closure' is not subsettable`

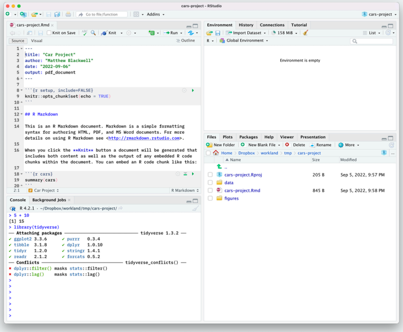
We’ll tend toward the Engineering model because it’s better suited to keep the mess in check.

2/ Let's take a touR

R versus RStudio



```
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: aarch64-apple-darwin20 (64-bit)

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

  Natural language support but running in an English locale

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

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

>
```



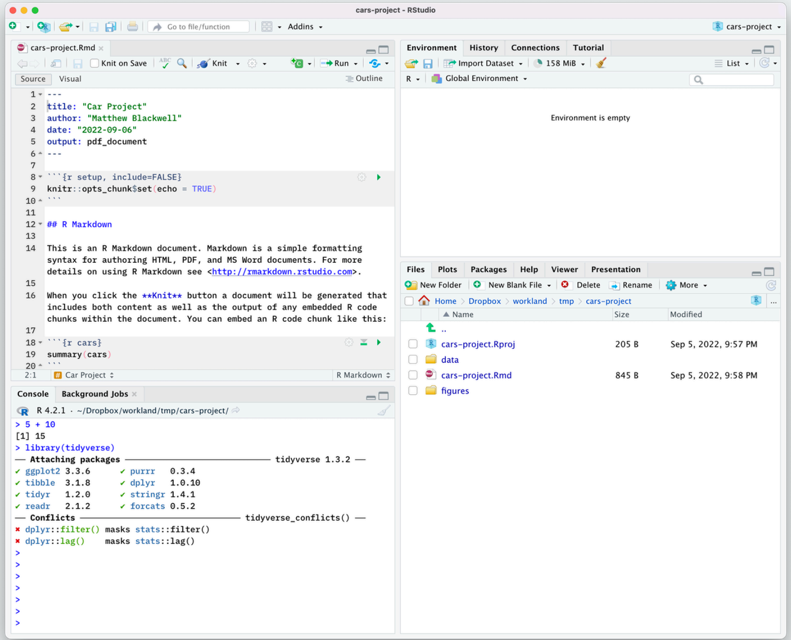
~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

R Markdown

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

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

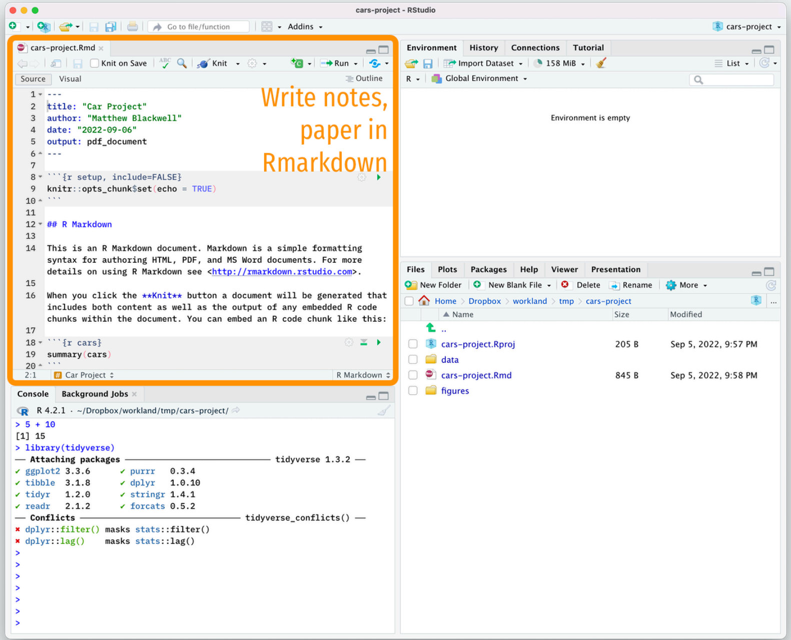
```{r cars}
summary(cars)
```
```

```
R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────── tidyverse 1.3.2 ──
✓ ggplot2 3.3.6      ✓ purrr   0.3.4
✓ tibble  3.1.8      ✓ dplyr   1.0.10
✓ tidyr   1.2.0      ✓ stringr 1.4.1
✓ readr   2.1.2      ✓ forcats 0.5.2
── Conflicts ────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Home > Dropbox > workland > tmp > cars-project

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

```



~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

R Markdown

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

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

```{r cars}
summary(cars)
```
~~~

```
R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6      ✔ purrr   0.3.4
✔ tibble  3.1.8      ✔ dplyr   1.0.10
✔ tidyr   1.2.0      ✔ stringr 1.4.1
✔ readr   2.1.2      ✔ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		



Write notes, paper in Rmarkdown

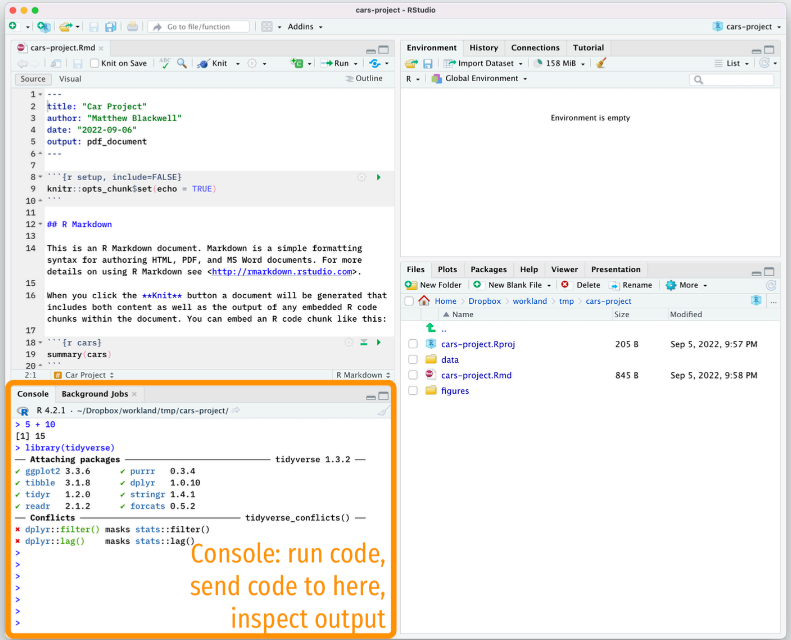
~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

R Markdown

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

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

```{r cars}
summary(cars)
```
```

| Name | Size | Modified |
|---|---|---|
| .. | | |
| cars-project.Rproj | 205 B | Sep 5, 2022, 9:57 PM |
| data | | |
| cars-project.Rmd | 845 B | Sep 5, 2022, 9:58 PM |
| figures | | |

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6   ✔ purrr   0.3.4
✔ tibble 3.1.8   ✔ dplyr   1.0.10
✔ tidyr   1.2.0   ✔ stringr 1.4.1
✔ readr   2.1.2   ✔ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()   masks stats::lag()
>
>
>
>
>
>
>
```
```



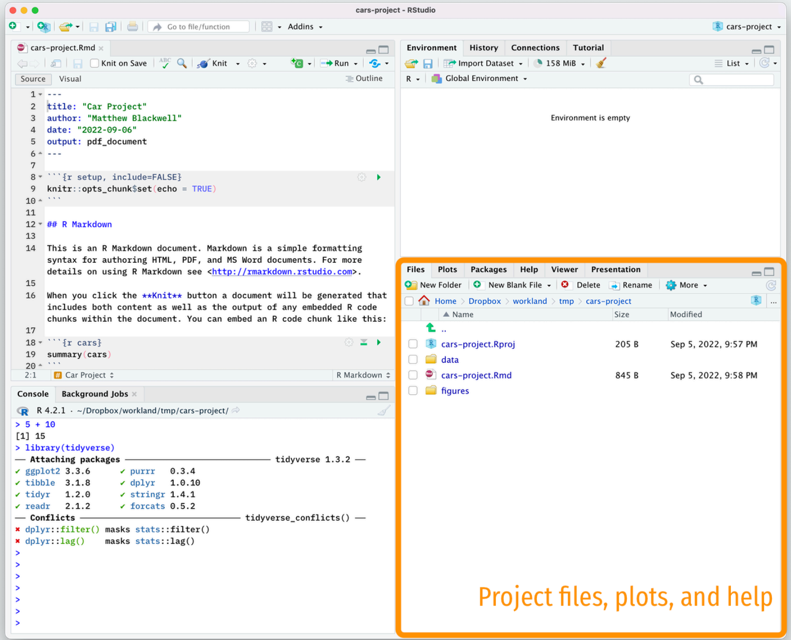
~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

R Markdown

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

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

```{r cars}
summary(cars)
```
```

Environment is empty

Home > Dropbox > workland > tmp > cars-project

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6     ✔ purrr   0.3.4
✔ tibble  3.1.8     ✔ dplyr   1.0.10
✔ tidyr   1.2.0     ✔ stringr 1.4.1
✔ readr   2.1.2     ✔ forcats 0.5.2
── Conflicts ────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Console: run code, send code to here, inspect output
```



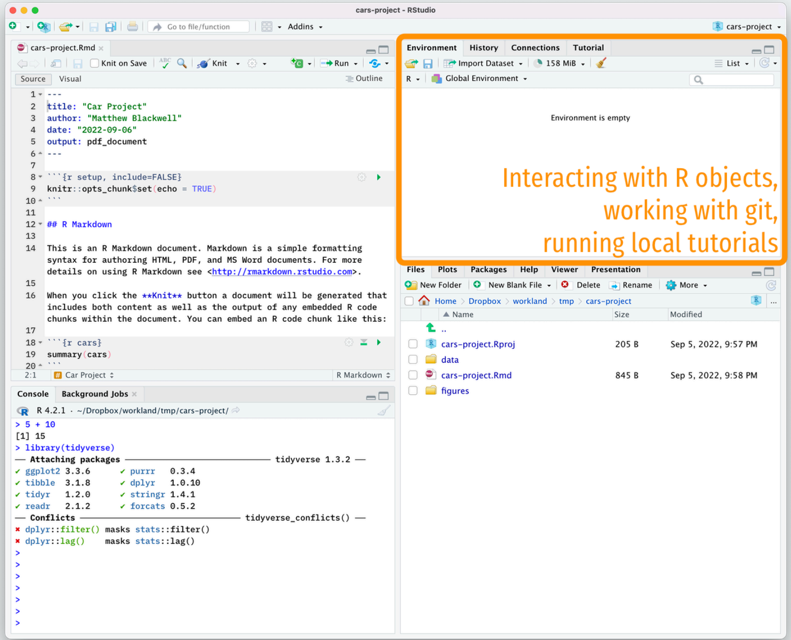
~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

## R Markdown

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

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

```{r cars}
summary(cars)
```
```

```
R 4.2.1 · ~/Dropbox/workland/tmp/cars-project/
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6     ✔ purrr   0.3.4
✔ tibble  3.1.8     ✔ dplyr   1.0.10
✔ tidyr   1.2.0     ✔ stringr 1.4.1
✔ readr   2.1.2     ✔ forcats 0.5.2
── Conflicts ────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

Project files, plots, and help
```



~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

R Markdown

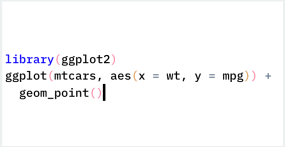
This is an R Markdown document. Markdown is a simple formatting
syntax for authoring HTML, PDF, and MS Word documents. For more
details on using R Markdown see <http://rmarkdown.rstudio.com>.

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

```{r cars}
summary(cars)
```
```

```
> 5 + 10
[1] 15
> library(tidyverse)
── Attaching packages ─────────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6      ✔ purrr   0.3.4
✔ tibble  3.1.8      ✔ dplyr   1.0.10
✔ tidyr   1.2.0      ✔ stringr 1.4.1
✔ readr   2.1.2      ✔ forcats 0.5.2
── Conflicts ────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
>
>
>
>
>
>
>
```

Name	Size	Modified
..		
cars-project.Rproj	205 B	Sep 5, 2022, 9:57 PM
data		
cars-project.Rmd	845 B	Sep 5, 2022, 9:58 PM
figures		

Interacting with R objects,
working with git,
running local tutorials
```

# 3/ Using Rmarkdown

# The acts of coding



```r
library(ggplot2)
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

Figure: 1. Writing code

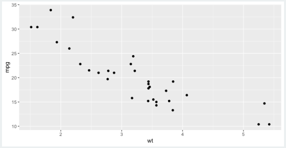
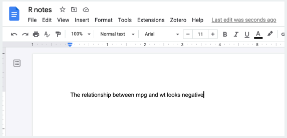


Figure: 2. Looking at output



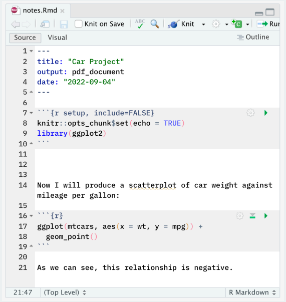
R notes
File Edit View Insert Format Tools Extensions Zotero Help
Last edit was seconds ago
100%
Normal text
Arial
11

The relationship between mpg and wt looks negative

Figure: 3. Taking notes

How to do all of these efficiently?

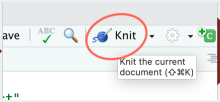
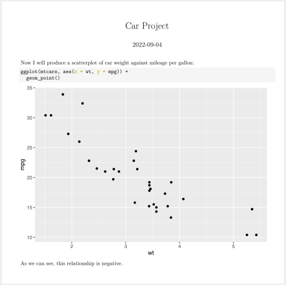
Rmarkdown files to the rescue



~~~
---
title: "Car Project"
output: pdf_document
date: "2022-09-04"
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2)
~~~

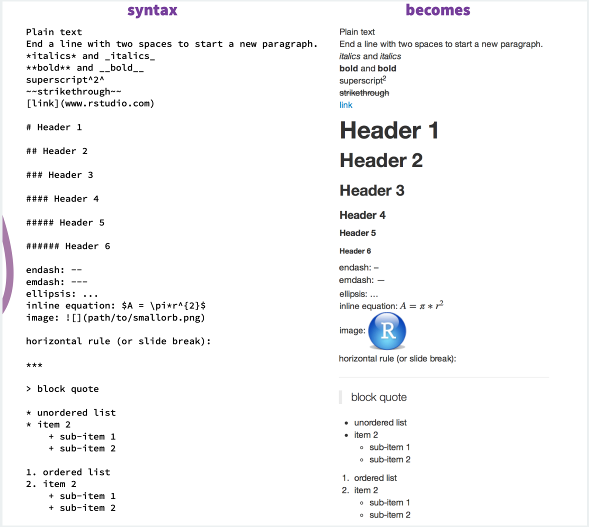
Now I will produce a scatterplot of car weight against
mileage per gallon:

```{r}
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

As we can see, this relationship is negative.
```

Figure: Rmarkdown file

Keep code and notes together in plain text


ave
Knit

Knit the current document (⇧⌘K)

Figure: Knit in R



Car Project

2022-09-04

Now I will produce a scatterplot of car weight against mileage per gallon:

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```

As we can see, this relationship is negative.

Figure: PDF output

Produce nice-looking outputs in different formats
```

Markdown: formatting in plain text

Non-code text in Rmd files is plain text with formatting instructions



syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

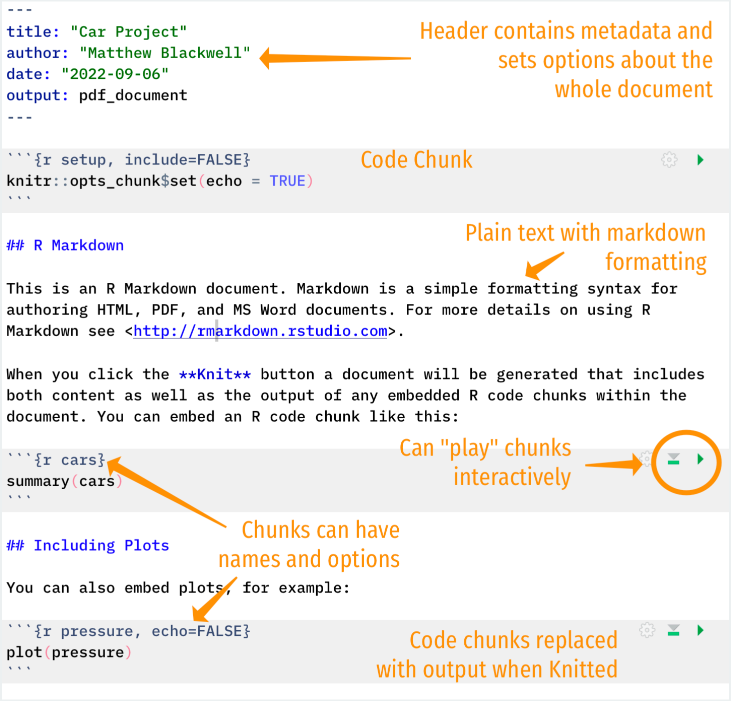
endash: –
emdash: —
ellipsis: ...
inline equation: $A = \pi * r^2$
image: R

horizontal rule (or slide break):

> block quote

- unordered list
- item 2
    - sub-item 1
    - sub-item 2

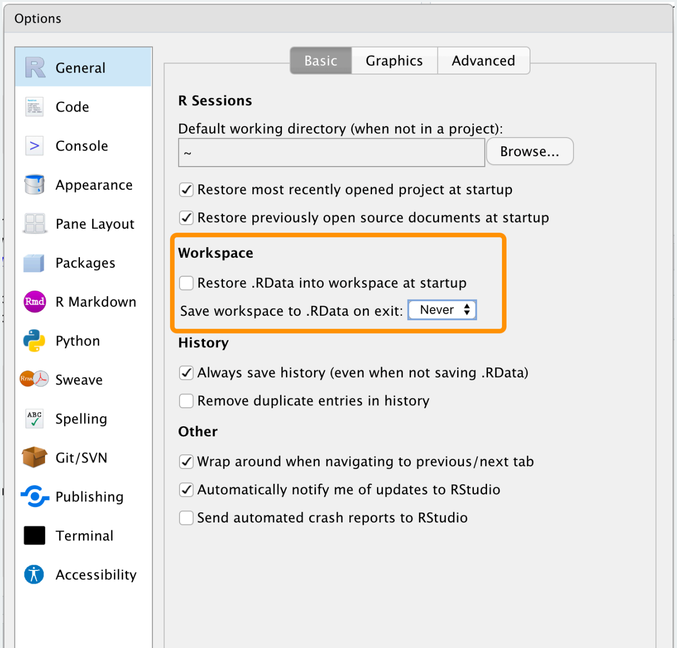
1. ordered list
2. item 2
    - sub-item 1
    - sub-item 2



~~~
---
title: "Car Project"
author: "Matthew Blackwell"
date: "2022-09-06"
output: pdf_document
---

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

## R Markdown

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

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

```{r cars}
summary(cars)
```

## Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```
~~~

Header contains metadata and sets options about the whole document
Code Chunk
Plain text with markdown formatting
Can "play" chunks interactively
Chunks can have names and options
Code chunks replaced with output when Knitted

Remember what's real



Options

R Sessions

Default working directory (when not in a project): ~

[✓] Restore most recently opened project at startup

[✓] Restore previously open source documents at startup

Workspace

[ ] Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

History

[✓] Always save history (even when not saving .RData)

[ ] Remove duplicate entries in history

Other

[✓] Wrap around when navigating to previous/next tab

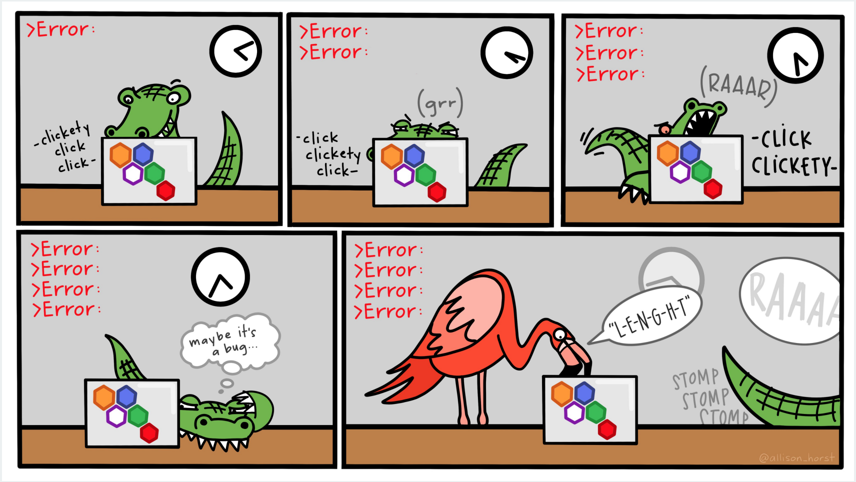
[✓] Automatically notify me of updates to RStudio

[ ] Send automated crash reports to RStudio

4/ Getting R bearings

Try to type your code by hand

Typing speeds up the try-fail cycle



Physically typing the code is best way to familiarize yourself with R and the try-fail-try-fail-try-succeed cycle

Credit: Allison Horst

What R looks like

Code that you can type and run:

```r
## Any R code that begins with the # character is a comment
## Comments are ignored by R

my_numbers <- c(4, 8, 15, 16, 23, 42) # Anything after # is also a comment
```

Output from code prefixed by ## by convention:

```r
my_numbers
```

```
## [1]  4  8 15 16 23 42
```

Output also has a counter in brackets when over one line:

```r
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
## [13] "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
## [25] "y" "z"
```

Everything in R has a name

```
my_numbers # just created this
```

```
## [1]  4  8 15 16 23 42
```

```
letters # this is built into R
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
## [13] "m" "n" "o" "p" "q" "r" "s" "t" "u" "v" "w" "x"
## [25] "y" "z"
```

```
pi # also built in
```

```
## [1] 3.14
```

Some names are forbidden (NA, TRUE, FALSE, etc) or strongly not recommended (c, mean, `table`)

We do things in R with functions

Functions take in objects, perform actions, and return outputs:

```r
mean(x = my_numbers)
```

```
## [1] 18
```

- `x` is the argument name,
- `my_numbers` is what we're passing to the that argument

If you omit the argument name, R will assume the default order:

```r
mean(my_numbers)
```

```
## [1] 18
```

Getting help with R

How do we know the default argument order? Look to help files:

```r
help(mean)
?mean # shorter
```

- Sometimes inscrutable, so look elsewhere:
  - Google, StackOverflow, Twitter, RStudio Community.
  - Ask on Ed or on class Slack.
  - Come to section, office hours, study hall.
- Get help **early** before becoming too frustrated!
  - Easy to overlook small issues like missing commas, etc.

Functions live in packages

Packages are bundles of functions written by other users that we can use.

Install packages using `install.packages()` to have them on your machine:

```
install.packages("ggplot2")
```

Load them into your R session with `library()`:

```
library(ggplot2)
```

Now we can use any function provided by `ggplot2`.

Functions live in packages

We can also use the `mypackage::` prefix to access package functions without loading:

```r
knitr::kable(head(mtcars))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.5	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.0	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.6	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.4	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.0	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.46	20.2	1	0	3	1

5/ Our first visualizations

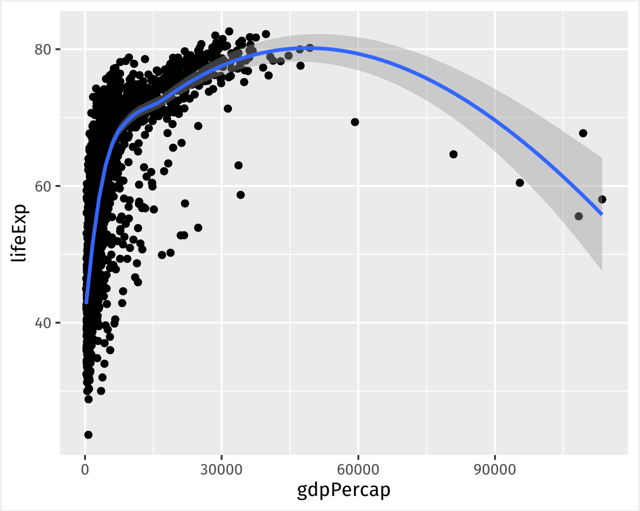
Gapminder data

```r
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPe~1
##    <fct>       <fct>     <int>   <dbl>    <int>   <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333    779.
##  2 Afghanistan Asia       1957    30.3  9240934    821.
##  3 Afghanistan Asia       1962    32.0 10267083    853.
##  4 Afghanistan Asia       1967    34.0 11537966    836.
##  5 Afghanistan Asia       1972    36.1 13079460    740.
##  6 Afghanistan Asia       1977    38.4 14880372    786.
##  7 Afghanistan Asia       1982    39.9 12881816    978.
##  8 Afghanistan Asia       1987    40.8 13867957    852.
##  9 Afghanistan Asia       1992    41.7 16317921    649.
## 10 Afghanistan Asia       1997    41.8 22227415    635.
## # ... with 1,694 more rows, and abbreviated variable
## #   name 1: gdpPercap
```

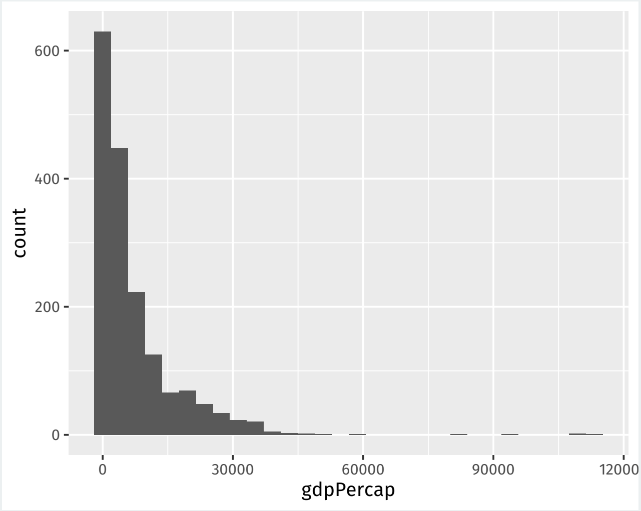
Plotting life expectancy over time

```
ggplot(gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point() + geom_smooth(method = "loess")
```



A histogram of GDP per capita

```r
ggplot(gapminder, mapping = aes(x = gdpPercap)) +
  geom_histogram()
```



Gov 50: 3. Data Visualization

Matthew Blackwell

Harvard University

Roadmap

1. Building plots by layers
2. Histograms and boxplots
3. Grouped data

1/ Building plots by layers

Midwest data

```
midwest
```

```
## # A tibble: 437 x 28
##      PID county    state  area popto~1 popde~2 popwh~3 popbl~4 popam~5
##    <int> <chr>     <chr> <dbl>   <int>   <dbl>   <int>   <int>   <int>
##  1   561 ADAMS     IL    0.052   66090   1271.   63917    1702      98
##  2   562 ALEXANDER IL    0.014   10626    759     7054    3496      19
##  3   563 BOND      IL    0.022   14991    681.   14477     429      35
##  4   564 BOONE     IL    0.017   30806   1812.   29344     127      46
##  5   565 BROWN     IL    0.018    5836    324.    5264     547      14
##  6   566 BUREAU    IL    0.05    35688    714.   35157      50      65
##  7   567 CALHOUN   IL    0.017    5322    313.    5298       1       8
##  8   568 CARROLL   IL    0.027   16805    622.   16519     111      30
##  9   569 CASS      IL    0.024   13437    560.   13384      16       8
## 10   570 CHAMPAIGN IL    0.058  173025   2983.  146506   16559     331
## # ... with 427 more rows, 19 more variables: popasian <int>,
## #   popother <int>, percwhite <dbl>, percblack <dbl>,
## #   percamerindan <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>,
## #   percbelowpoverty <dbl>, percchildbelowpovert <dbl>,
## #   percadultpoverty <dbl>, percelderlypoverty <dbl>, ...
```

Building up a graph in pieces

Create ggplot object and direct it to the correct data:

```r
p <- ggplot(data = midwest)
```

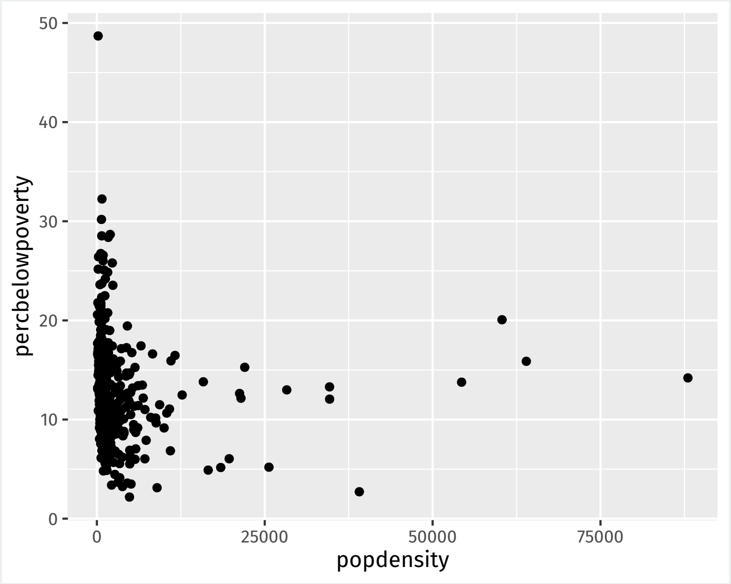
Mapping: tell ggplot what visual aesthetics correspond to which variables

```r
p <- ggplot(data = midwest,
            mapping = aes(x = popdensity,
                          y = percbelowpoverty))
```

Other aesthetic mappings: `color`, `shape`, `size`, etc.

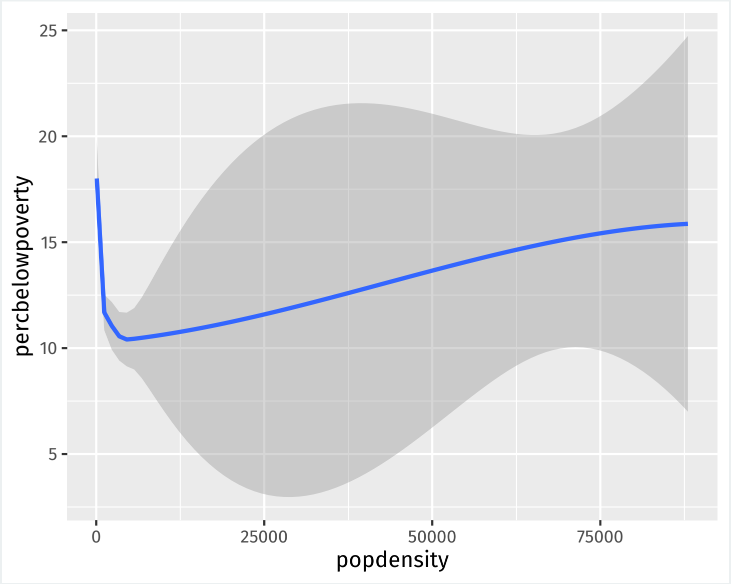
Adding a geom layer

```r
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point()
```


percbelowpoverty
popdensity
0
10
20
30
40
50
0
25000
50000
75000

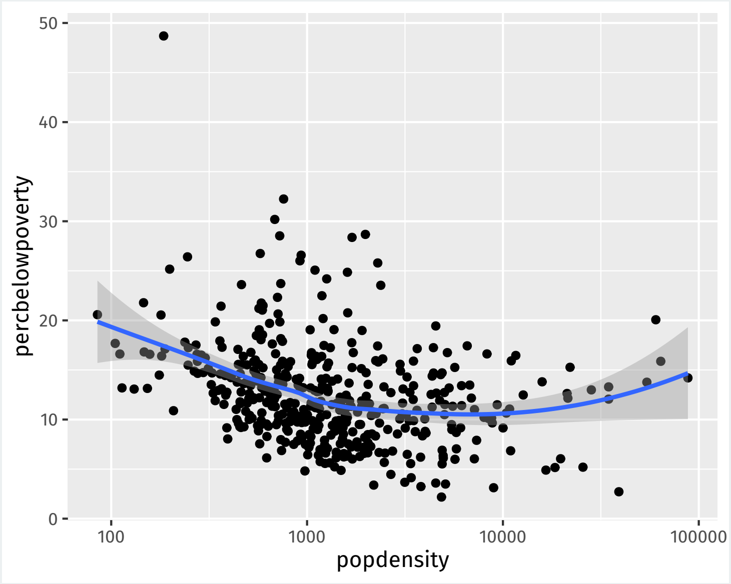
Trying a new geom

```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_smooth()
```


25
20
15
10
5
percbelowpoverty
0
25000
50000
75000
popdensity

Layering geoms is additive

```r
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10()
```



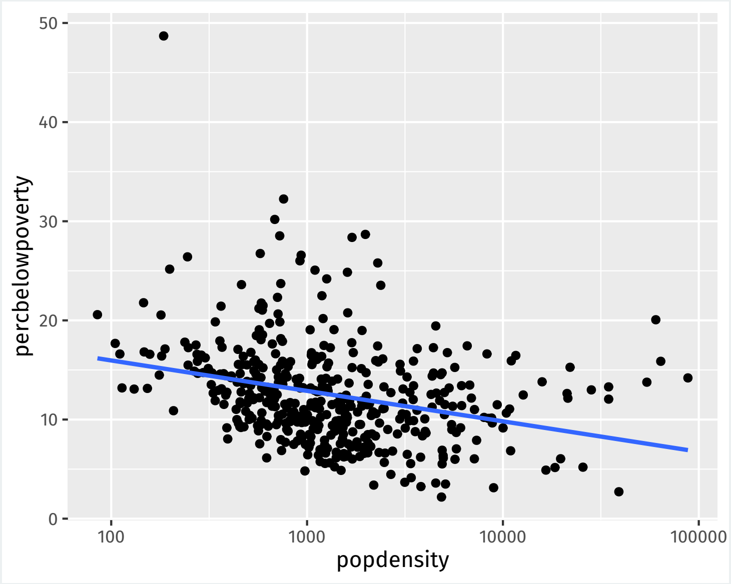
50
40
30
20
10
0
percbelowpoverty
100
1000
10000
100000
popdensity

Geoms are functions

Geoms can take arguments:

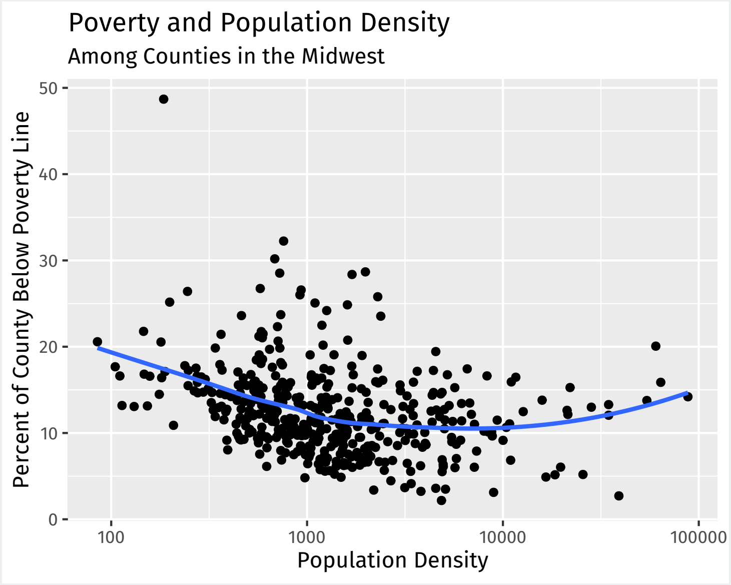
```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point() +
  geom_smooth(method = ”lm”, se = FALSE) +
  scale_x_log10()
```

Tells `geom_smooth` to do a linear fit with no error region


50
40
30
20
10
0
percbelowpoverty
100
1000
10000
100000
popdensity

Adding informative labels

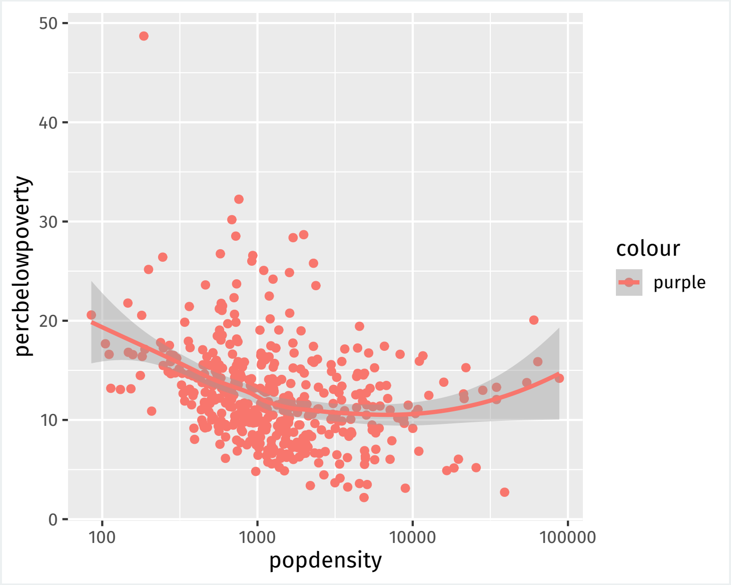
```r
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point() +
  geom_smooth(method = ”loess”, se = FALSE) +
  scale_x_log10() +
  labs(x = ”Population Density”,
       y = ”Percent of County Below Poverty Line”,
       title = ”Poverty and Population Density”,
       subtitle = ”Among Counties in the Midwest”,
       source = ”US Census, 2000”)
```


Poverty and Population Density
Among Counties in the Midwest
Percent of County Below Poverty Line
0
10
20
30
40
50
100
1000
10000
100000
Population Density

Mapping vs setting aesthetics

```r
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty,
                     color = "purple")) +
  geom_point() +
  geom_smooth() +
  scale_x_log10()
```

Wait what?



50
40
30
20
10
0
percbelowpoverty
100
1000
10000
100000
popdensity
colour
purple

Mapping always refers to variables

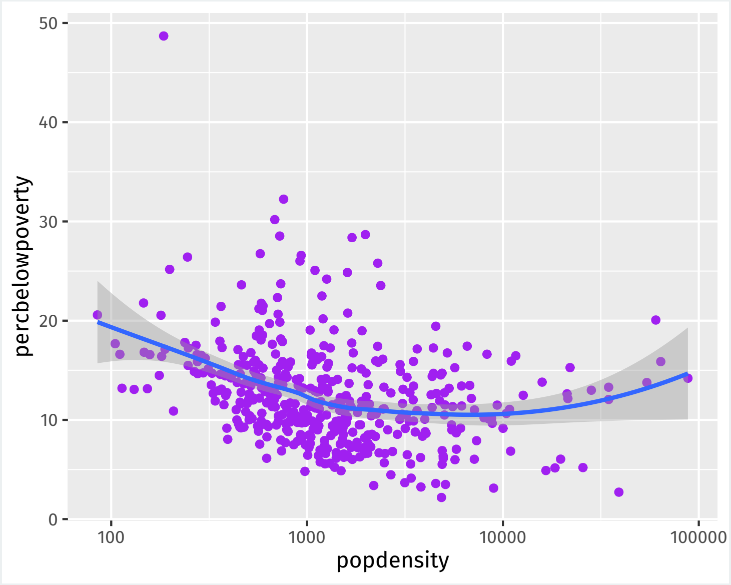
If passed a value other than a variable name, ggplot will implicitly create a variable with that value (in this case ”purple” that is constant)

```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty,
                     color = ”purple”)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10()
```

Setting aesthetics

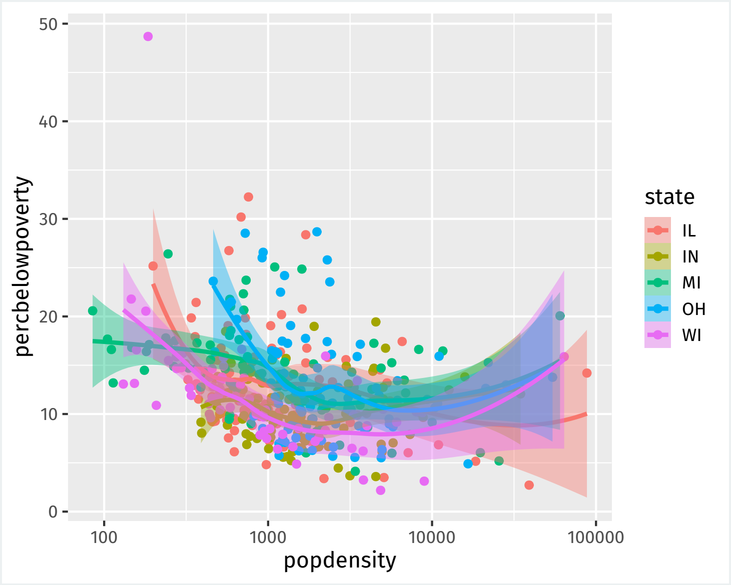
Set the color outside the `mapping = aes()` format.

```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point(color = ”purple”) +
  geom_smooth() +
  scale_x_log10()
```


percbelowpoverty
50
40
30
20
10
0
100
1000
10000
100000
popdensity

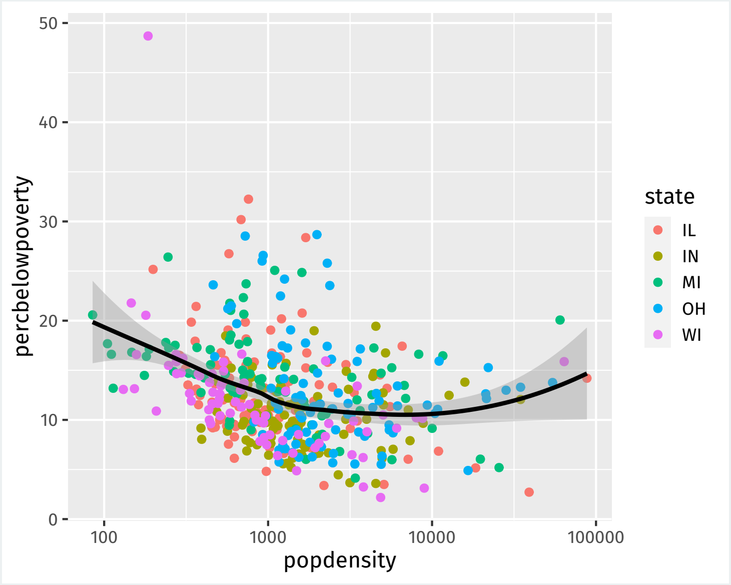
Mapping more aesthetics

```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty,
                     color = state,
                     fill = state)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10()
```


50
40
30
20
10
0
percbelowpoverty
100
1000
10000
100000
popdensity
state
IL
IN
MI
OH
WI

Mappings can be done on a per geom basis

```
ggplot(data = midwest,
       mapping = aes(x = popdensity,
                     y = percbelowpoverty)) +
  geom_point(mapping = aes(color = state)) +
  geom_smooth(color = ”black”) +
  scale_x_log10()
```



50
40
30
20
10
0
percbelowpoverty
100
1000
10000
100000
popdensity
state
IL
IN
MI
OH
WI

2/ Histograms and boxplots

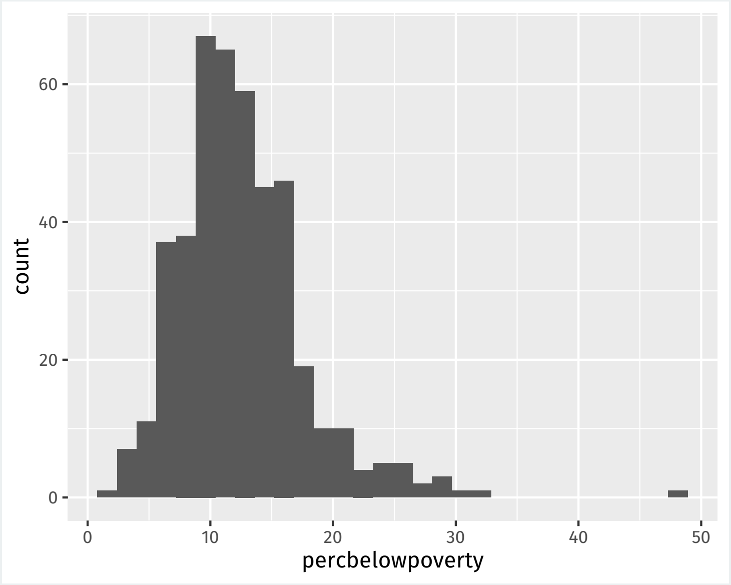
Histograms

Histograms show where there are more or fewer observations of a numeric variable.

```
ggplot(data = midwest,
       mapping = aes(x = percbelowpoverty)) +
  geom_histogram()
```

Split up range of variable into bins, count how many are in each bin.

y aesthetic calculated automatically.

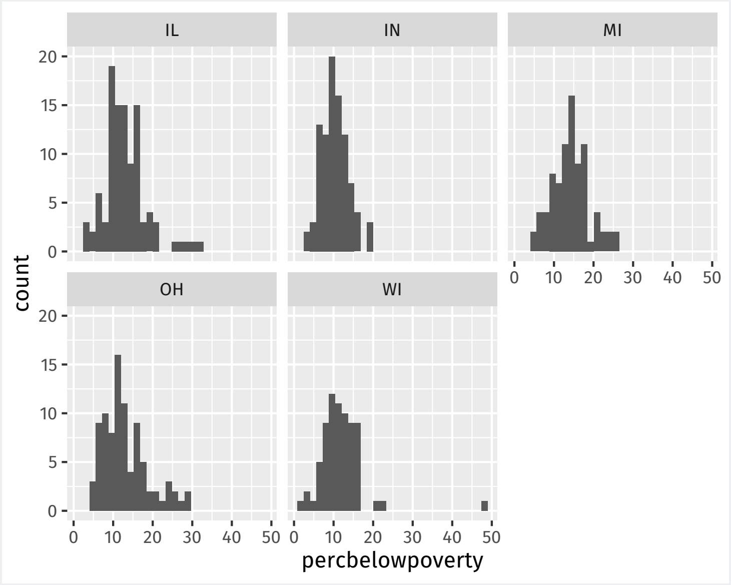

count
60
40
20
0
0
10
20
30
40
50
percbelowpoverty

Creating small multiples with facets

Small multiples: a series of similar graphs with the same scale/axes to help with comparing different partitions of a dataset.

```
ggplot(data = midwest,
       mapping = aes(x = percbelowpoverty)) +
  geom_histogram() +
  facet_wrap(~ state)
```

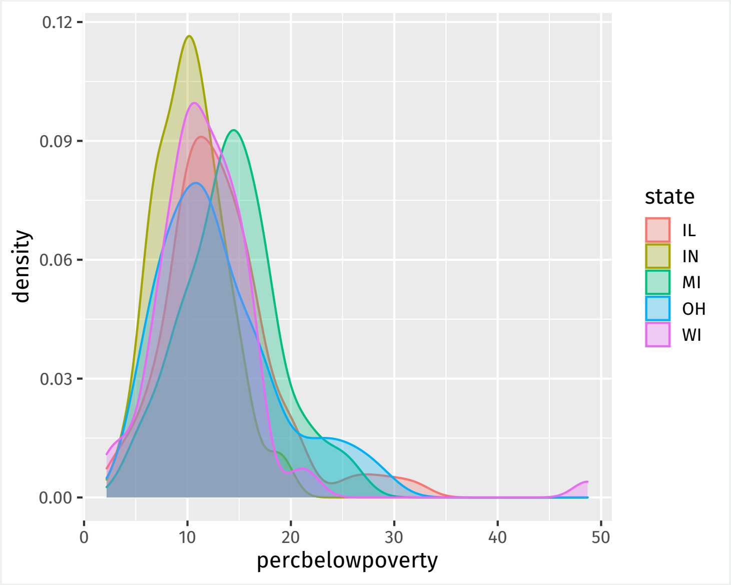
We'll see more of the `~ variable` syntax (called a formula).


IL
IN
MI
OH
WI
count
percbelowpoverty
0
5
10
15
20
0
10
20
30
40
50

Density as alternative to histograms

A **kernel density** plot is a smoothed version of a histogram and slightly easier to overlay.

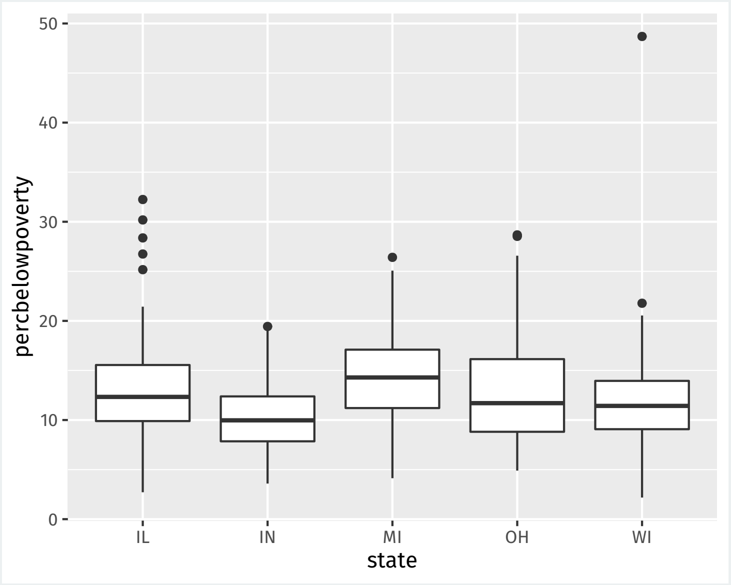
```
ggplot(data = midwest,
       mapping = aes(x = percbelowpoverty,
                     fill = state, color = state)) +
  geom_density(alpha = 0.3)
```


0.12
0.09
0.06
0.03
0.00
density
0
10
20
30
40
50
percbelowpoverty
state
IL
IN
MI
OH
WI

Boxplots

Boxplots are another way to compare distributions across discrete groups.

```r
ggplot(data = midwest,
       mapping = aes(x = state,
                     y = percbelowpoverty)) +
  geom_boxplot()
```


percbelowpoverty
50
40
30
20
10
0
IL
IN
MI
OH
WI
state

Boxplots in R

- “Box” represents middle 50% of the data.
    - 25% of the data above the box, 25% below
    - Width of the box is called the inter quartile range (IQR)
- Horizontal line in the box is the median
    - 50% of the data above the median, 50% below
- “Whiskers” represents either:
    - 1.5 $\times$ IQR or max/min of the data, whichever is smaller.
    - Points beyond whiskers are outliers.

3/ Grouped data

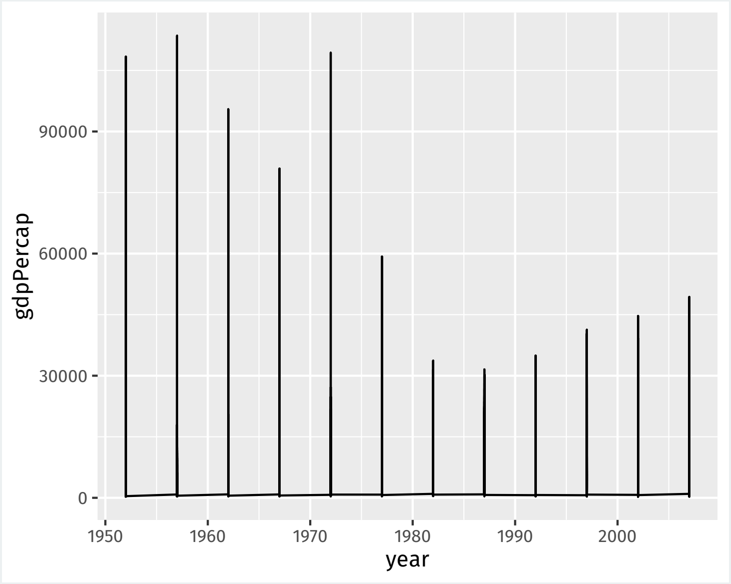
Back to the gapminder data

```r
glimpse(gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afgh~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, As~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 19~
## $ lifeExp   <dbl> 28.8, 30.3, 32.0, 34.0, 36.1, 38.4, 39.9, 40.8, 41~
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14~
## $ gdpPercap <dbl> 779, 821, 853, 836, 740, 786, 978, 852, 649, 635, ~
```

Let’s plot the trend in income

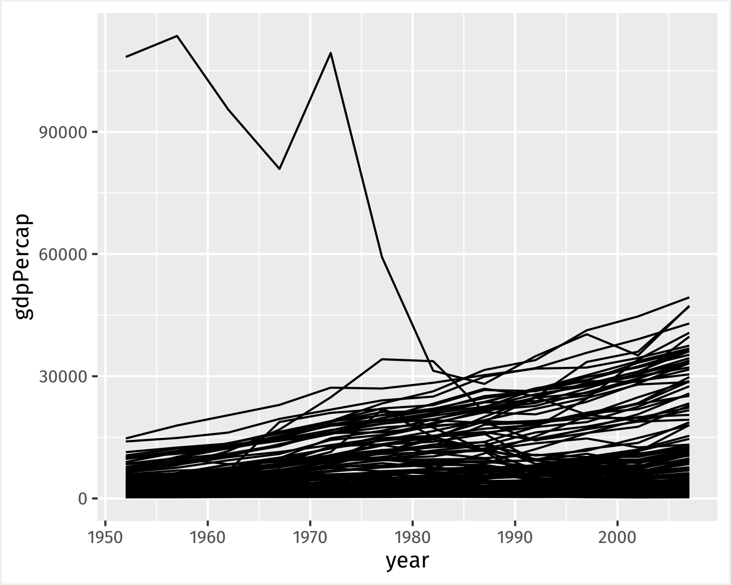
```
ggplot(data = gapminder,
       mapping = aes(x = year,
                     y = gdpPercap)) +
  geom_line()
```



`geom_line` connects points from different countries in the same year.

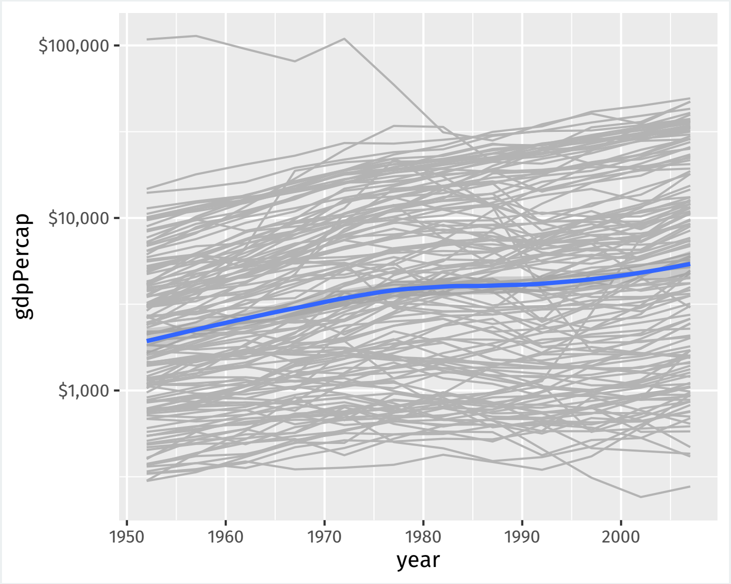
Tell `geom_line` how to group the lines

```r
ggplot(data = gapminder,
       mapping = aes(x = year,
                     y = gdpPercap)) +
  geom_line(mapping = aes(group = country))
```


gdpPercap
90000
60000
30000
0
1950
1960
1970
1980
1990
2000
year

Scales

```
ggplot(data = gapminder,
       mapping = aes(x = year,
                     y = gdpPercap)) +
  geom_line(mapping = aes(group = country), color = "grey70") +
  geom_smooth(method = "loess") +
  scale_y_log10(labels = scales::dollar)
```


$100,000
$10,000
$1,000
gdpPercap
1950
1960
1970
1980
1990
2000
year

Gov 50: 4. Data Wrangling

Matthew Blackwell

Harvard University

Roadmap

1. Data Wrangling
2. Operating on rows
3. Operating on columns
4. Operating on groups

1/ Data Wrangling

Why?

data.frames vs tibbles

- The standard R object for datasets is the `data.frame`
    - Each column is a vector of the same length.
    - Columns can be different types
- Access columns with $: `mydata$myvariable`

```
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8
## [12] 16.4 17.3 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5
## [23] 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4
```

Problems with data frames

```
mtcars
```

```
##                      mpg cyl  disp  hp drat   wt qsec vs am
## Mazda RX4           21.0   6 160.0 110 3.90 2.62 16.5  0  1
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.88 17.0  0  1
## Datsun 710          22.8   4 108.0  93 3.85 2.32 18.6  1  1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.21 19.4  1  0
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.44 17.0  0  0
## Valiant             18.1   6 225.0 105 2.76 3.46 20.2  1  0
## Duster 360          14.3   8 360.0 245 3.21 3.57 15.8  0  0
## Merc 240D           24.4   4 146.7  62 3.69 3.19 20.0  1  0
## Merc 230            22.8   4 140.8  95 3.92 3.15 22.9  1  0
## Merc 280            19.2   6 167.6 123 3.92 3.44 18.3  1  0
## Merc 280C           17.8   6 167.6 123 3.92 3.44 18.9  1  0
## Merc 450SE          16.4   8 275.8 180 3.07 4.07 17.4  0  0
## Merc 450SL          17.3   8 275.8 180 3.07 3.73 17.6  0  0
## Merc 450SLC         15.2   8 275.8 180 3.07 3.78 18.0  0  0
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.25 18.0  0  0
## Lincoln Continental 10.4   8 460.0 215 3.00 5.42 17.8  0  0
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.34 17.4  0  0
## Fiat 128            32.4   4  78.7  66 4.08 2.20 19.5  1  1
## Honda Civic         30.4   4  75.7  52 4.93 1.61 18.5  1  1
## Toyota Corolla      33.9   4  71.1  65 4.22 1.83 19.9  1  1
```

tibbles: a tidyverse alternative

```
midwest
```

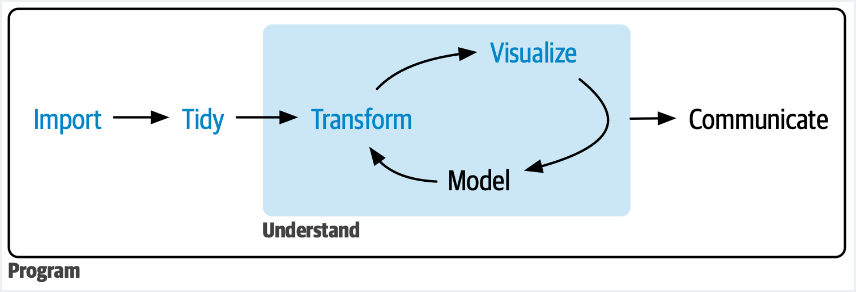
```
## # A tibble: 437 x 28
##       PID county    state  area poptotal popdensity
##     <int> <chr>     <chr> <dbl>    <int>      <dbl>
##  1    561 ADAMS     IL    0.052    66090      1271.
##  2    562 ALEXANDER IL    0.014    10626       759
##  3    563 BOND      IL    0.022    14991       681.
##  4    564 BOONE     IL    0.017    30806      1812.
##  5    565 BROWN     IL    0.018     5836       324.
##  6    566 BUREAU    IL    0.05     35688       714.
##  7    567 CALHOUN   IL    0.017     5322       313.
##  8    568 CARROLL   IL    0.027    16805       622.
##  9    569 CASS      IL    0.024    13437       560.
## 10    570 CHAMPAIGN IL    0.058   173025      2983.
## # ... with 427 more rows, and 22 more variables:
## #   popwhite <int>, popblack <int>,
## #   popamerindian <int>, popasian <int>,
## #   popother <int>, percwhite <dbl>, percblack <dbl>,
## #   percamerindan <dbl>, percasian <dbl>,
```

rows x columns

column types

abridged output

Transform-Visualize-Model cycle



Credit: Hadley Wickham

dplyr: a package for data transformation

- All `dplyr` functions:
    - Take a dataset as their first argument
    - Manipulate the dataset in some way
    - Returns the manipulated dataset

pipe

Nested calls can be hard to read (have to read inside out):

```
f(g(h(r(x))))
```

The pipe |> allows us to move output between functions (|> = “and then”):

```
x |>
  r() |>
  h() |>
  g() |>
  h()
```

The piped output goes to the first argument by default.

Local news data

- How does station ownership affect local news coverage?
- Martin and McCrain (2019) use data on local news at TV stations before and after a large acquisition by a conglomorate.

Variable	Description
`callsign`	Callsign of the station
`affiliation`	Network affiliation of the station
`date`	Airdate of news
`weekday`	Day of the week of airdate
`ideology`	Measure of news slant (bigger is more conservative)
`national_politics`	Avg proportion of segments on national politics
`local_politics`	Avg proportion of segments on national politics
`sinclair2017`	Station acquired by Sinclair group in Sept 2017
`post`	Date is before/after acquisition (0/1)

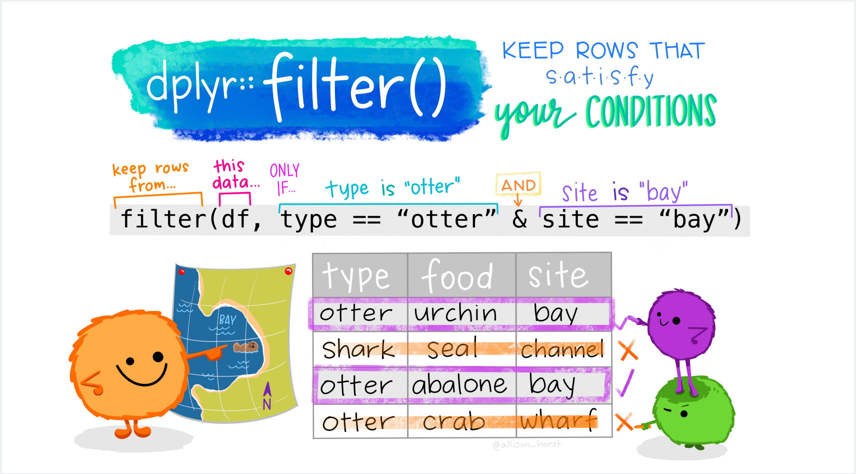
```r
library(gov50data)
data(news)
news
```

```
## # A tibble: 3,137 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KRBC     NBC         2017-06-05 Mon      NA       0.0286
##  2 KTAB     CBS         2017-06-05 Mon      NA       0.0286
##  3 KXVA     FOX         2017-06-05 Mon      NA       0.0393
##  4 KPAX     CBS         2017-06-06 Tue      NA       0.00357
##  5 KTAB     CBS         2017-06-06 Tue      NA       0.0945
##  6 KECI     NBC         2017-06-07 Wed       0.0655  0.225
##  7 KPAX     CBS         2017-06-07 Wed       0.0853  0.283
##  8 KRBC     NBC         2017-06-07 Wed       0.0183  0.130
##  9 KTAB     CBS         2017-06-07 Wed       0.0850  0.0901
## 10 KTMF     ABC         2017-06-07 Wed       0.0842  0.152
## # ... with 3,127 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

2/ Operating on rows

filter()

`filter()` selects rows that satisfy the argument you pass it:



type	food	site
otter	urchin	bay
shark	seal	channel
otter	abalone	bay
otter	crab	wharf

Credit: Allison Horst

```r
news |>
  filter(weekday == "Tue")
```

```
## # A tibble: 626 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KPAX     CBS         2017-06-06 Tue     NA        0.00357
##  2 KTAB     CBS         2017-06-06 Tue     NA        0.0945
##  3 KAEF     ABC         2017-06-13 Tue      0.0242   0.180
##  4 KBVU     FOX         2017-06-13 Tue      0.00894  0.186
##  5 KBZK     CBS         2017-06-13 Tue      0.129    0.306
##  6 KCVU     FOX         2017-06-13 Tue      0.114    0.124
##  7 KECI     NBC         2017-06-13 Tue      0.115    0.283
##  8 KHSL     CBS         2017-06-13 Tue      0.0821   0.274
##  9 KNVN     NBC         2017-06-13 Tue      0.120    0.261
## 10 KPAX     CBS         2017-06-13 Tue      0.0984   0.208
## # ... with 616 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

Multiple conditions means “and”

```r
news |>
  filter(weekday == "Tue",
         affiliation == "FOX")
```

```
## # A tibble: 130 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KBVU     FOX         2017-06-13 Tue      0.00894   0.186
##  2 KCVU     FOX         2017-06-13 Tue      0.114     0.124
##  3 WEMT     FOX         2017-06-13 Tue      0.235     0.149
##  4 WYDO     FOX         2017-06-13 Tue      0.0949    0.182
##  5 KBVU     FOX         2017-06-20 Tue     NA         0.0229
##  6 KCVU     FOX         2017-06-20 Tue     NA         0.0170
##  7 KXVA     FOX         2017-06-20 Tue     NA         0.0203
##  8 WEMT     FOX         2017-06-20 Tue      0.268     0.134
##  9 WYDO     FOX         2017-06-20 Tue      0.0590    0.155
## 10 KBVU     FOX         2017-06-27 Tue     NA         0.0601
## # ... with 120 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

logicals

- Comparing two values/vectors:
    - `>`/`>=`: greater than/greater than or equal to
    - `<`/`<=`: less than/less than or equal to
    - `==`/`!=`: equal to/not equal to
- Combining multiple logical statements:
    - `&`: and
    - `|`: or

Common gotcha!

```r
news |>
  filter(weekday = "Tue")
```

```
## Error in `filter()`:
## ! We detected a named input.
## i This usually means that you've used `=` instead of `==`.
## i Did you mean `weekday == "Tue"`?
```

```r
news |>
  filter(affiliation == "FOX" | affiliation == "ABC")
```

```
## # A tibble: 1,525 x 10
##    callsign affiliation date       weekday  ideology natio~1
##    <chr>    <chr>       <date>     <ord>       <dbl>   <dbl>
##  1 KXVA     FOX         2017-06-05 Mon     NA        0.0393
##  2 KTMF     ABC         2017-06-07 Wed      0.0842   0.152
##  3 KTXS     ABC         2017-06-07 Wed     -0.000488 0.0925
##  4 KXVA     FOX         2017-06-07 Wed     NA        0.00718
##  5 KAEF     ABC         2017-06-08 Thu      0.0426   0.213
##  6 KBVU     FOX         2017-06-08 Thu     -0.0860   0.169
##  7 KTMF     ABC         2017-06-08 Thu      0.0433   0.179
##  8 KTXS     ABC         2017-06-08 Thu      0.0627   0.158
##  9 KXVA     FOX         2017-06-08 Thu     NA        0.0124
## 10 WCTI     ABC         2017-06-08 Thu      0.139    0.225
## # ... with 1,515 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

```r
news |>
  filter(ideology < 0 & weekday == "Tue")
```

```
## # A tibble: 66 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KAEF     ABC         2017-06-27 Tue     -0.0117     0.162
##  2 KECI     NBC         2017-06-27 Tue     -0.00362    0.177
##  3 KHSL     CBS         2017-06-27 Tue     -0.0735     0.170
##  4 KNVN     NBC         2017-06-27 Tue     -0.0175     0.180
##  5 KPAX     CBS         2017-06-27 Tue     -0.134      0.219
##  6 KTXS     ABC         2017-06-27 Tue     -0.0307     0.129
##  7 WCTI     ABC         2017-06-27 Tue     -0.0308     0.187
##  8 WITN     NBC         2017-06-27 Tue     -0.0233     0.155
##  9 WJHL     CBS         2017-06-27 Tue     -0.00388    0.166
## 10 WNCT     CBS         2017-06-27 Tue     -0.130      0.181
## # ... with 56 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

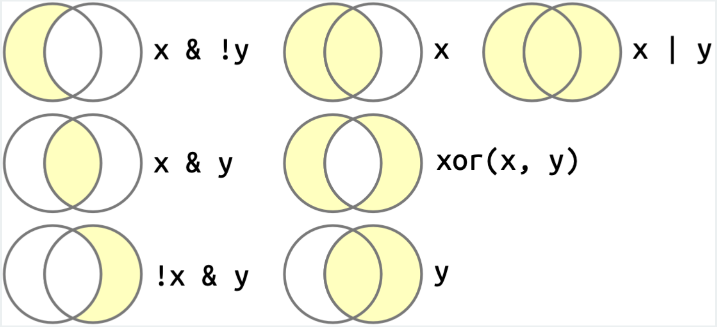
Combining %in%

When combining | and ==, useful to use %in%:

```
news |>
  filter(weekday %in% c("Mon", "Fri"))
```

```
## # A tibble: 1,253 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KRBC     NBC         2017-06-05 Mon      NA        0.0286
##  2 KTAB     CBS         2017-06-05 Mon      NA        0.0286
##  3 KXVA     FOX         2017-06-05 Mon      NA        0.0393
##  4 KAEF     ABC         2017-06-09 Fri       0.0870   0.153
##  5 KBVU     FOX         2017-06-09 Fri      NA        0.0553
##  6 KECI     NBC         2017-06-09 Fri       0.115    0.216
##  7 KPAX     CBS         2017-06-09 Fri       0.0882   0.315
##  8 KRBC     NBC         2017-06-09 Fri       0.0929   0.152
##  9 KTAB     CBS         2017-06-09 Fri       0.0588   0.0711
## 10 KTMF     ABC         2017-06-09 Fri      NA        0.0495
## # ... with 1,243 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

Complicated logicals



`arrange()`

`arrange()` will reorder the rows based on the values of the columns.

With multiple arguments, sort by first argument, then second, then third...

Arrange by callsign then date

```r
news |>
  arrange(callsign, date)
```

```
## # A tibble: 3,137 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KAEF     ABC         2017-06-08 Thu       0.0426   0.213
##  2 KAEF     ABC         2017-06-09 Fri       0.0870   0.153
##  3 KAEF     ABC         2017-06-12 Mon       0.0135   0.149
##  4 KAEF     ABC         2017-06-13 Tue       0.0242   0.180
##  5 KAEF     ABC         2017-06-14 Wed       0.123    0.182
##  6 KAEF     ABC         2017-06-15 Thu       0.0778   0.114
##  7 KAEF     ABC         2017-06-16 Fri      NA        0.109
##  8 KAEF     ABC         2017-06-19 Mon       0.778    0.0823
##  9 KAEF     ABC         2017-06-20 Tue       0.115    0.131
## 10 KAEF     ABC         2017-06-21 Wed      -0.315    0.130
## # ... with 3,127 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

Which station-dates were the most liberal?

```r
news |>
  arrange(ideology)
```

```
## # A tibble: 3,137 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KRBC     NBC         2017-10-19 Thu       -0.674   0.0731
##  2 WJHL     CBS         2017-12-08 Fri       -0.673   0.0364
##  3 KRBC     NBC         2017-10-18 Wed       -0.586   0.0470
##  4 KCVU     FOX         2017-06-22 Thu       -0.414   0.158
##  5 KRBC     NBC         2017-12-11 Mon       -0.365   0.0674
##  6 KAEF     ABC         2017-06-21 Wed       -0.315   0.130
##  7 KTMF     ABC         2017-12-01 Fri       -0.303   0.179
##  8 KWYB     ABC         2017-12-01 Fri       -0.303   0.160
##  9 KTVM     NBC         2017-09-01 Fri       -0.302   0.0507
## 10 KNVN     NBC         2017-12-08 Fri       -0.299   0.121
## # ... with 3,127 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

Which station-dates were the most conservative?

Use desc() to reverse the order:

```
news |>
  arrange(desc(ideology))
```

```
## # A tibble: 3,137 x 10
##    callsign affiliation date       weekday ideology nation~1
##    <chr>    <chr>       <date>     <ord>      <dbl>    <dbl>
##  1 KAEF     ABC         2017-06-19 Mon        0.778   0.0823
##  2 WYDO     FOX         2017-07-19 Wed        0.580   0.126
##  3 KRCR     ABC         2017-10-03 Tue        0.566   0.123
##  4 KAEF     ABC         2017-10-18 Wed        0.496   0.0892
##  5 KBVU     FOX         2017-11-16 Thu        0.491   0.159
##  6 KTMF     ABC         2017-11-06 Mon        0.455   0.138
##  7 KAEF     ABC         2017-06-29 Thu        0.447   0.126
##  8 KPAX     CBS         2017-11-23 Thu        0.437   0.125
##  9 KTAB     CBS         2017-11-16 Thu        0.427   0.0631
## 10 KCVU     FOX         2017-07-06 Thu        0.406   0.154
## # ... with 3,127 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

3/ Operating on columns

`select()`:

`select()` selects columns via their names.

Selecting based on names

```r
news |>
  select(callsign, date, ideology)
```

```
## # A tibble: 3,137 x 3
##    callsign date       ideology
##    <chr>    <date>        <dbl>
##  1 KRBC     2017-06-05  NA
##  2 KTAB     2017-06-05  NA
##  3 KXVA     2017-06-05  NA
##  4 KPAX     2017-06-06  NA
##  5 KTAB     2017-06-06  NA
##  6 KECI     2017-06-07   0.0655
##  7 KPAX     2017-06-07   0.0853
##  8 KRBC     2017-06-07   0.0183
##  9 KTAB     2017-06-07   0.0850
## 10 KTMF     2017-06-07   0.0842
## # ... with 3,127 more rows
```

Selecting based on a range of variables

```r
news |>
  select(callsign:ideology)
```

```
## # A tibble: 3,137 x 5
##    callsign affiliation date       weekday ideology
##    <chr>    <chr>       <date>     <ord>      <dbl>
##  1 KRBC     NBC         2017-06-05 Mon       NA
##  2 KTAB     CBS         2017-06-05 Mon       NA
##  3 KXVA     FOX         2017-06-05 Mon       NA
##  4 KPAX     CBS         2017-06-06 Tue       NA
##  5 KTAB     CBS         2017-06-06 Tue       NA
##  6 KECI     NBC         2017-06-07 Wed        0.0655
##  7 KPAX     CBS         2017-06-07 Wed        0.0853
##  8 KRBC     NBC         2017-06-07 Wed        0.0183
##  9 KTAB     CBS         2017-06-07 Wed        0.0850
## 10 KTMF     ABC         2017-06-07 Wed        0.0842
## # ... with 3,127 more rows
```

Selecting all not in a range

```r
news |>
  select(!callsign:ideology)
```

```
## # A tibble: 3,137 x 5
##    national_politics local_politics sinclair2017  post month
##                <dbl>          <dbl>        <dbl> <dbl> <ord>
##  1           0.0286          0.0190            0     0 Jun
##  2           0.0286          0.0190            0     0 Jun
##  3           0.0393          0.0262            0     0 Jun
##  4           0.00357         0.194             0     0 Jun
##  5           0.0945          0.109             0     0 Jun
##  6           0.225           0.148             1     0 Jun
##  7           0.283           0.123             0     0 Jun
##  8           0.130           0.189             0     0 Jun
##  9           0.0901          0.138             0     0 Jun
## 10           0.152           0.129             0     0 Jun
## # ... with 3,127 more rows
```

Selecting all numeric columns

```r
news |>
  select(where(is.numeric))
```

```
## # A tibble: 3,137 x 5
##    ideology national_politics local_politics sinclai~1  post
##       <dbl>             <dbl>          <dbl>     <dbl> <dbl>
##  1  NA                0.0286         0.0190          0     0
##  2  NA                0.0286         0.0190          0     0
##  3  NA                0.0393         0.0262          0     0
##  4  NA                0.00357        0.194           0     0
##  5  NA                0.0945         0.109           0     0
##  6   0.0655           0.225          0.148           1     0
##  7   0.0853           0.283          0.123           0     0
##  8   0.0183           0.130          0.189           0     0
##  9   0.0850           0.0901         0.138           0     0
## 10   0.0842           0.152          0.129           0     0
## # ... with 3,127 more rows, and abbreviated variable name
## #   1: sinclair2017
```

Combining multiple selections

```r
news |>
  select(callsign:weekday, ends_with("politics"))
```

```
## # A tibble: 3,137 x 6
##    callsign affiliation date       weekday nationa~1 local~2
##    <chr>    <chr>       <date>     <ord>       <dbl>   <dbl>
##  1 KRBC     NBC         2017-06-05 Mon       0.0286   0.0190
##  2 KTAB     CBS         2017-06-05 Mon       0.0286   0.0190
##  3 KXVA     FOX         2017-06-05 Mon       0.0393   0.0262
##  4 KPAX     CBS         2017-06-06 Tue       0.00357  0.194
##  5 KTAB     CBS         2017-06-06 Tue       0.0945   0.109
##  6 KECI     NBC         2017-06-07 Wed       0.225    0.148
##  7 KPAX     CBS         2017-06-07 Wed       0.283    0.123
##  8 KRBC     NBC         2017-06-07 Wed       0.130    0.189
##  9 KTAB     CBS         2017-06-07 Wed       0.0901   0.138
## 10 KTMF     ABC         2017-06-07 Wed       0.152    0.129
## # ... with 3,127 more rows, and abbreviated variable names
## #   1: national_politics, 2: local_politics
```

`rename()`

`rename(new_name = old_name)` renames the `old_name` variable to `new_name`

```r
news |>
  rename(call_sign = callsign)
```

```
## # A tibble: 3,137 x 10
##    call_sign affiliation date       weekday ideology natio~1
##    <chr>     <chr>       <date>     <ord>      <dbl>   <dbl>
##  1 KRBC      NBC         2017-06-05 Mon       NA     0.0286
##  2 KTAB      CBS         2017-06-05 Mon       NA     0.0286
##  3 KXVA      FOX         2017-06-05 Mon       NA     0.0393
##  4 KPAX      CBS         2017-06-06 Tue       NA     0.00357
##  5 KTAB      CBS         2017-06-06 Tue       NA     0.0945
##  6 KECI      NBC         2017-06-07 Wed        0.0655 0.225
##  7 KPAX      CBS         2017-06-07 Wed        0.0853 0.283
##  8 KRBC      NBC         2017-06-07 Wed        0.0183 0.130
##  9 KTAB      CBS         2017-06-07 Wed        0.0850 0.0901
## 10 KTMF      ABC         2017-06-07 Wed        0.0842 0.152
## # ... with 3,127 more rows, 4 more variables:
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>, and abbreviated variable name
## #   1: national_politics
```

`mutate()`

`mutate(new_var = fun(old_vars))` adds new columns that are functions of existing columns.

```r
news |>
  mutate(
    national_local_diff = national_politics - local_politics,
    national_politics_perc = national_politics * 100
  ) |>
  select(callsign, date, national_politics, local_politics,
         national_local_diff, national_politics_perc)
```

```
## # A tibble: 3,137 x 6
##    callsign date       national_politics local_politics national_local_diff national_politics_perc
##    <chr>    <date>                 <dbl>          <dbl>               <dbl>                  <dbl>
##  1 KRBC     2017-06-05           0.0286          0.0190             0.00952                  2.86
##  2 KTAB     2017-06-05           0.0286          0.0190             0.00952                  2.86
##  3 KXVA     2017-06-05           0.0393          0.0262             0.0131                   3.93
##  4 KPAX     2017-06-06           0.00357         0.194             -0.191                    0.357
##  5 KTAB     2017-06-06           0.0945          0.109             -0.0145                   9.45
##  6 KECI     2017-06-07           0.225           0.148              0.0761                  22.5
##  7 KPAX     2017-06-07           0.283           0.123              0.160                   28.3
##  8 KRBC     2017-06-07           0.130           0.189             -0.0589                  13.0
##  9 KTAB     2017-06-07           0.0901          0.138             -0.0476                   9.01
## 10 KTMF     2017-06-07           0.152           0.129              0.0229                  15.2
## # ... with 3,127 more rows
```

`if_else()`

`if_else(test_condition, yes, no)` allows us to create a vector that depends on a logical

New vector gets `yes` expression when `test_condition` is TRUE, no otherwise

```r
news |>
  mutate(Ownership = if_else(sinclair2017 == 1,
                             "Acquired by Sinclair",
                             "Not Acquired")) |>
  select(callsign, affiliation, date, Ownership)
```

```
## # A tibble: 3,137 x 4
##    callsign affiliation date       Ownership
##    <chr>    <chr>       <date>     <chr>
##  1 KRBC     NBC         2017-06-05 Not Acquired
##  2 KTAB     CBS         2017-06-05 Not Acquired
##  3 KXVA     FOX         2017-06-05 Not Acquired
##  4 KPAX     CBS         2017-06-06 Not Acquired
##  5 KTAB     CBS         2017-06-06 Not Acquired
##  6 KECI     NBC         2017-06-07 Acquired by Sinclair
##  7 KPAX     CBS         2017-06-07 Not Acquired
##  8 KRBC     NBC         2017-06-07 Not Acquired
##  9 KTAB     CBS         2017-06-07 Not Acquired
## 10 KTMF     ABC         2017-06-07 Not Acquired
## # ... with 3,127 more rows
```

4/ Operating on groups

`group_by()`

`group_by(var)` divides the data into groups based on the `var` variable.

Doesn't change data yet, but subsequent operations will by `var`.

```r
news |>
  group_by(month)
```

```
## # A tibble: 3,137 x 10
## # Groups:   month [7]
##    callsign affil~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon     NA      0.0286  0.0190        0
##  2 KTAB     CBS     2017-06-05 Mon     NA      0.0286  0.0190        0
##  3 KXVA     FOX     2017-06-05 Mon     NA      0.0393  0.0262        0
##  4 KPAX     CBS     2017-06-06 Tue     NA      0.00357 0.194         0
##  5 KTAB     CBS     2017-06-06 Tue     NA      0.0945  0.109         0
##  6 KECI     NBC     2017-06-07 Wed      0.0655 0.225   0.148         1
##  7 KPAX     CBS     2017-06-07 Wed      0.0853 0.283   0.123         0
##  8 KRBC     NBC     2017-06-07 Wed      0.0183 0.130   0.189         0
##  9 KTAB     CBS     2017-06-07 Wed      0.0850 0.0901  0.138         0
## 10 KTMF     ABC     2017-06-07 Wed      0.0842 0.152   0.129         0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: affiliation,
## #   2: ideology, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

`summarize()`

`summarize(sum_var = fun(curr_var))` calculates summaries of variables by groups.

Ideological slant by weekday

```r
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 2
##   month slant_mean
##   <ord>      <dbl>
## 1 Jun       0.0786
## 2 Jul       0.103
## 3 Aug       0.105
## 4 Sep       0.0751
## 5 Oct       0.0862
## 6 Nov       0.0972
## 7 Dec       0.0774
```

Summaries by ownership and pre/post

```r
news |>
  group_by(sinclair2017, post) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE),
    national_mean = mean(national_politics, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 4
## # Groups:   sinclair2017 [2]
##   sinclair2017  post slant_mean national_mean
##          <dbl> <dbl>      <dbl>         <dbl>
## 1            0     0     0.100          0.118
## 2            0     1     0.0768         0.107
## 3            1     0     0.0936         0.124
## 4            1     1     0.0938         0.144
```

Summarize across types of variables

`across()` will apply a summary function across many variables

```
news |>
  group_by(sinclair2017, post) |>
  summarize(
    across(where(is.numeric), mean, na.rm = TRUE),
  )
```

```
## # A tibble: 4 x 5
## # Groups:   sinclair2017 [2]
##   sinclair2017  post ideology national_politics local_politics
##          <dbl> <dbl>    <dbl>             <dbl>          <dbl>
## 1            0     0   0.100              0.118          0.158
## 2            0     1   0.0768             0.107          0.150
## 3            1     0   0.0936             0.124          0.170
## 4            1     1   0.0938             0.144          0.147
```

Gov 50: 5. Data Wrangling and Barplots

Matthew Blackwell

Harvard University

Roadmap

1. Operating on rows
2. Operating on columns
3. Operating on groups
4. Creating barplots

Local news data

- How does station ownership affect local news coverage?
- Martin and McCrain (2019) use data on local news at TV stations before and after a large acquisition by a conglomorate.

Variable	Description
`callsign`	Callsign of the station
`affiliation`	Network affiliation of the station
`date`	Airdate of news
`weekday`	Day of the week of airdate
`ideology`	Measure of news slant (bigger is more conservative)
`national_politics`	Avg proportion of segments on national politics
`local_politics`	Avg proportion of segments on national politics
`sinclair2017`	Station acquired by Sinclair group in Sept 2017
`post`	Date is before/after acquisition (0/1)

```r
library(gov50data)
data(news)
news
```

```
## # A tibble: 3,137 x 10
##    callsign affil~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon     NA      0.0286  0.0190        0
##  2 KTAB     CBS     2017-06-05 Mon     NA      0.0286  0.0190        0
##  3 KXVA     FOX     2017-06-05 Mon     NA      0.0393  0.0262        0
##  4 KPAX     CBS     2017-06-06 Tue     NA      0.00357 0.194         0
##  5 KTAB     CBS     2017-06-06 Tue     NA      0.0945  0.109         0
##  6 KECI     NBC     2017-06-07 Wed      0.0655 0.225   0.148         1
##  7 KPAX     CBS     2017-06-07 Wed      0.0853 0.283   0.123         0
##  8 KRBC     NBC     2017-06-07 Wed      0.0183 0.130   0.189         0
##  9 KTAB     CBS     2017-06-07 Wed      0.0850 0.0901  0.138         0
## 10 KTMF     ABC     2017-06-07 Wed      0.0842 0.152   0.129         0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: affiliation,
## #   2: ideology, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

1/ Operating on rows

slice()

`slice()` can give you a specific set of rows:

```r
## first and third row
news |>
  slice(1, 3)
```

```
## # A tibble: 2 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-06-05 Mon          NA  0.0286  0.0190       0
## 2 KXVA     FOX      2017-06-05 Mon          NA  0.0393  0.0262       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

You can ask for a range of rows with `start:stop` syntax:

```r
## first three rows
news |>
  slice(1:3)
```

```
## # A tibble: 3 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-06-05 Mon          NA  0.0286  0.0190       0
## 2 KTAB     CBS      2017-06-05 Mon          NA  0.0286  0.0190       0
## 3 KXVA     FOX      2017-06-05 Mon          NA  0.0393  0.0262       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

slice_max()

`slice_max(var, n = 5)` will return the top 5 observations on column `var`

```
news |>
  slice_max(ideology, n = 5)
```

```
## # A tibble: 5 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KAEF     ABC      2017-06-19 Mon       0.778  0.0823   0.179       1
## 2 WYDO     FOX      2017-07-19 Wed       0.580  0.126    0.121       1
## 3 KRCR     ABC      2017-10-03 Tue       0.566  0.123    0.192       1
## 4 KAEF     ABC      2017-10-18 Wed       0.496  0.0892   0.217       1
## 5 KBVU     FOX      2017-11-16 Thu       0.491  0.159    0.184       1
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

slice_min()

`slice_min(var, n = 5)` will return the bottom 5 observations on column `var`

```
news |>
  slice_min(ideology, n = 5)
```

```
## # A tibble: 5 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-10-19 Thu      -0.674  0.0731   0.161       0
## 2 WJHL     CBS      2017-12-08 Fri      -0.673  0.0364   0.206       0
## 3 KRBC     NBC      2017-10-18 Wed      -0.586  0.0470   0.135       0
## 4 KCVU     FOX      2017-06-22 Thu      -0.414  0.158    0.172       1
## 5 KRBC     NBC      2017-12-11 Mon      -0.365  0.0674   0.163       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

2/ Operating on columns

`rename()`

`rename(new_name = old_name)` renames the `old_name` variable to `new_name`

```r
news |>
  rename(call_sign = callsign)
```

```
## # A tibble: 3,137 x 10
##    call_s~1 affil~2 date       weekday ideol~3 natio~4 local~5 sincl~6
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon     NA      0.0286   0.0190       0
##  2 KTAB     CBS     2017-06-05 Mon     NA      0.0286   0.0190       0
##  3 KXVA     FOX     2017-06-05 Mon     NA      0.0393   0.0262       0
##  4 KPAX     CBS     2017-06-06 Tue     NA      0.00357  0.194        0
##  5 KTAB     CBS     2017-06-06 Tue     NA      0.0945   0.109        0
##  6 KECI     NBC     2017-06-07 Wed      0.0655 0.225    0.148        1
##  7 KPAX     CBS     2017-06-07 Wed      0.0853 0.283    0.123        0
##  8 KRBC     NBC     2017-06-07 Wed      0.0183 0.130    0.189        0
##  9 KTAB     CBS     2017-06-07 Wed      0.0850 0.0901   0.138        0
## 10 KTMF     ABC     2017-06-07 Wed      0.0842 0.152    0.129        0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: call_sign,
## #   2: affiliation, 3: ideology, 4: national_politics,
## #   5: local_politics, 6: sinclair2017
```

`mutate()`

`mutate(new_var = fun(old_vars))` adds new columns that are functions of existing columns.

```r
news |>
  mutate(
    national_local_diff = national_politics - local_politics,
    national_politics_perc = national_politics * 100
  ) |>
  select(callsign, date, national_politics, local_politics,
         national_local_diff, national_politics_perc)
```

```
## # A tibble: 3,137 x 6
##    callsign date       national_politics local_politics national_local_diff national_politics_perc
##    <chr>    <date>                 <dbl>          <dbl>               <dbl>                  <dbl>
##  1 KRBC     2017-06-05           0.0286          0.0190             0.00952                  2.86 
##  2 KTAB     2017-06-05           0.0286          0.0190             0.00952                  2.86 
##  3 KXVA     2017-06-05           0.0393          0.0262             0.0131                   3.93 
##  4 KPAX     2017-06-06           0.00357         0.194             -0.191                    0.357
##  5 KTAB     2017-06-06           0.0945          0.109             -0.0145                   9.45 
##  6 KECI     2017-06-07           0.225           0.148              0.0761                  22.5  
##  7 KPAX     2017-06-07           0.283           0.123              0.160                   28.3  
##  8 KRBC     2017-06-07           0.130           0.189             -0.0589                  13.0  
##  9 KTAB     2017-06-07           0.0901          0.138             -0.0476                   9.01 
## 10 KTMF     2017-06-07           0.152           0.129              0.0229                  15.2  
## # ... with 3,127 more rows
```

`if_else()`

`if_else(test_condition, yes, no)` allows us to create a vector that depends on a logical

New vector gets `yes` expression when `test_condition` is TRUE, no otherwise

```r
news |>
  mutate(Ownership = if_else(sinclair2017 == 1,
                             "Acquired by Sinclair",
                             "Not Acquired")) |>
  select(callsign, affiliation, date, Ownership)
```

```
## # A tibble: 3,137 x 4
##    callsign affiliation date       Ownership
##    <chr>    <chr>       <date>     <chr>
##  1 KRBC     NBC         2017-06-05 Not Acquired
##  2 KTAB     CBS         2017-06-05 Not Acquired
##  3 KXVA     FOX         2017-06-05 Not Acquired
##  4 KPAX     CBS         2017-06-06 Not Acquired
##  5 KTAB     CBS         2017-06-06 Not Acquired
##  6 KECI     NBC         2017-06-07 Acquired by Sinclair
##  7 KPAX     CBS         2017-06-07 Not Acquired
##  8 KRBC     NBC         2017-06-07 Not Acquired
##  9 KTAB     CBS         2017-06-07 Not Acquired
## 10 KTMF     ABC         2017-06-07 Not Acquired
## # ... with 3,127 more rows
```

3/ Operating on groups

`group_by()`

`group_by(var)` divides the data into groups based on the `var` variable.

Doesn't change data yet, but subsequent operations will by `var`.

```r
news |>
  group_by(month)
```

```
## # A tibble: 3,137 x 10
## # Groups:   month [7]
##    callsign affil~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon     NA      0.0286  0.0190        0
##  2 KTAB     CBS     2017-06-05 Mon     NA      0.0286  0.0190        0
##  3 KXVA     FOX     2017-06-05 Mon     NA      0.0393  0.0262        0
##  4 KPAX     CBS     2017-06-06 Tue     NA      0.00357 0.194         0
##  5 KTAB     CBS     2017-06-06 Tue     NA      0.0945  0.109         0
##  6 KECI     NBC     2017-06-07 Wed      0.0655 0.225   0.148         1
##  7 KPAX     CBS     2017-06-07 Wed      0.0853 0.283   0.123         0
##  8 KRBC     NBC     2017-06-07 Wed      0.0183 0.130   0.189         0
##  9 KTAB     CBS     2017-06-07 Wed      0.0850 0.0901  0.138         0
## 10 KTMF     ABC     2017-06-07 Wed      0.0842 0.152   0.129         0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: affiliation,
## #   2: ideology, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

`summarize()`

`summarize(sum_var = fun(curr_var))` calculates summaries of variables by groups.

Ideological slant by weekday

```r
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 2
##   month slant_mean
##   <ord>      <dbl>
## 1 Jun       0.0786
## 2 Jul       0.103
## 3 Aug       0.105
## 4 Sep       0.0751
## 5 Oct       0.0862
## 6 Nov       0.0972
## 7 Dec       0.0774
```

Summaries by ownership and pre/post

```r
news |>
  group_by(sinclair2017, post) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE),
    national_mean = mean(national_politics, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 4
## # Groups:   sinclair2017 [2]
##   sinclair2017  post slant_mean national_mean
##          <dbl> <dbl>      <dbl>         <dbl>
## 1            0     0     0.100          0.118
## 2            0     1     0.0768         0.107
## 3            1     0     0.0936         0.124
## 4            1     1     0.0938         0.144
```

Summarize across types of variables

`across()` will apply a summary function across many variables

```r
news |>
  group_by(sinclair2017, post) |>
  summarize(
    across(where(is.numeric), mean, na.rm = TRUE),
  )
```

```
## # A tibble: 4 x 5
## # Groups:   sinclair2017 [2]
##   sinclair2017  post ideology national_politics local_politics
##          <dbl> <dbl>    <dbl>             <dbl>          <dbl>
## 1            0     0   0.100              0.118          0.158
## 2            0     1   0.0768             0.107          0.150
## 3            1     0   0.0936             0.124          0.170
## 4            1     1   0.0938             0.144          0.147
```

kable() to produce nice tables

```
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  ) |>
  knitr::kable()
```

month	slant_mean
Jun	0.079
Jul	0.103
Aug	0.105
Sep	0.075
Oct	0.086
Nov	0.097
Dec	0.077

Giving nicer column names

```r
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  ) |>
  knitr::kable(col.names = c("Month", "Avg. Slant"))
```

Month	Avg. Slant
Jun	0.079
Jul	0.103
Aug	0.105
Sep	0.075
Oct	0.086
Nov	0.097
Dec	0.077

Producing a table of counts of a categorical variable

```r
news |>
  group_by(affiliation) |>
  summarize(n = n())
```

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           863
## 2 CBS           807
## 3 FOX           662
## 4 NBC           805
```

Helper function `count()`

`count()` does the same thing:

```
news |>
  count(affiliation)
```

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           863
## 2 CBS           807
## 3 FOX           662
## 4 NBC           805
```

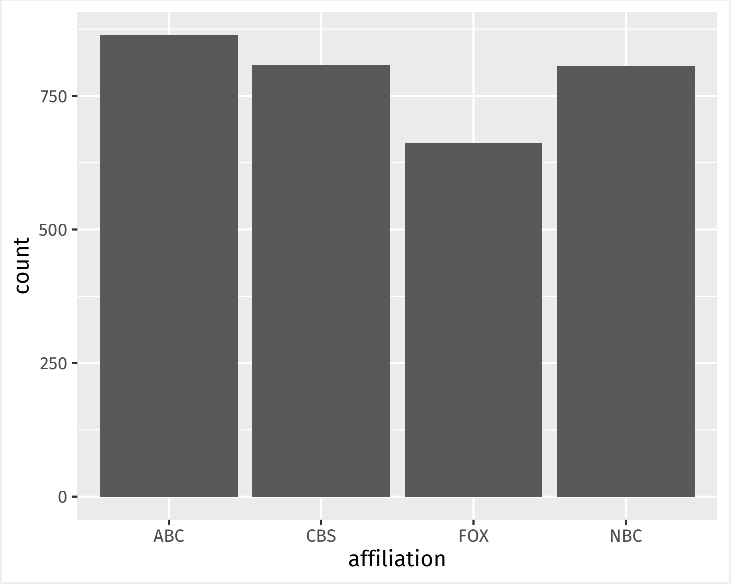
4/ Creating barplots

Combining our skills

Let’s combine our tools to produce a bar plot with `geom_bar()`

By default, bar plots take a single variable and show the number of observations in each category.

```r
ggplot(news, mapping = aes(x = affiliation)) +
  geom_bar()
```



Barplots of non-counts

Barplots can represent a lot beyond counts, including variables in our dataset or group summaries.

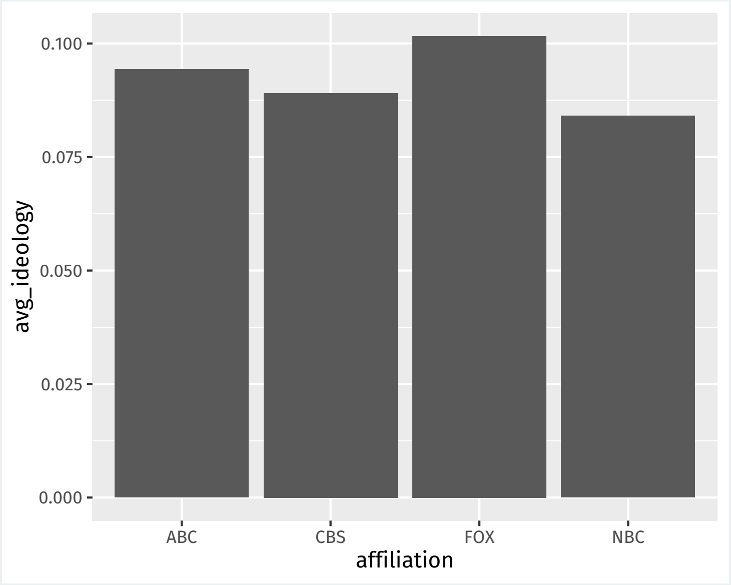
When the height of the bar is another variable in our data and not just a count, we set the `x` and `y` aesthetics and use `geom_col()` instead of `geom_bar()`.

Let’s create a group summary:

```r
aff_ideology_means <- news |>
  group_by(affiliation) |>
  summarize(avg_ideology = mean(ideology, na.rm = TRUE))
aff_ideology_means
```

```
## # A tibble: 4 x 2
##   affiliation avg_ideology
##   <chr>              <dbl>
## 1 ABC               0.0943
## 2 CBS               0.0891
## 3 FOX               0.102
## 4 NBC               0.0841
```

```r
ggplot(aff_ideology_means, aes(x = affiliation, y = avg_ideology)) +
  geom_col()
```



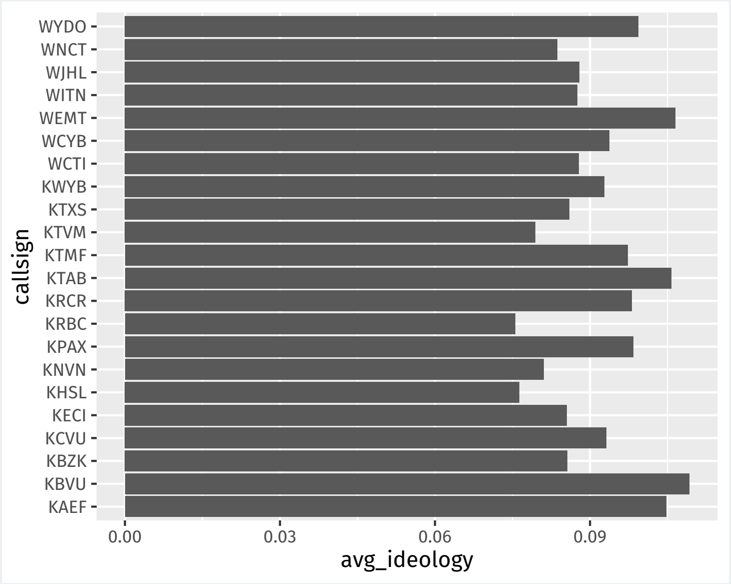
0.100
0.075
0.050
0.025
0.000
avg_ideology
ABC
CBS
FOX
NBC
affiliation

A more complicated example

Let’s create a barplot that shows the top 10 stations in terms of slant. First, let’s get the data:

```
station_ideology <- news |>
  group_by(callsign, affiliation) |>
  summarize(avg_ideology = mean(ideology, na.rm = TRUE)) |>
  slice_max(avg_ideology, n = 20)
```

```r
ggplot(station_ideology, aes(x = avg_ideology, y = callsign)) +
  geom_col()
```

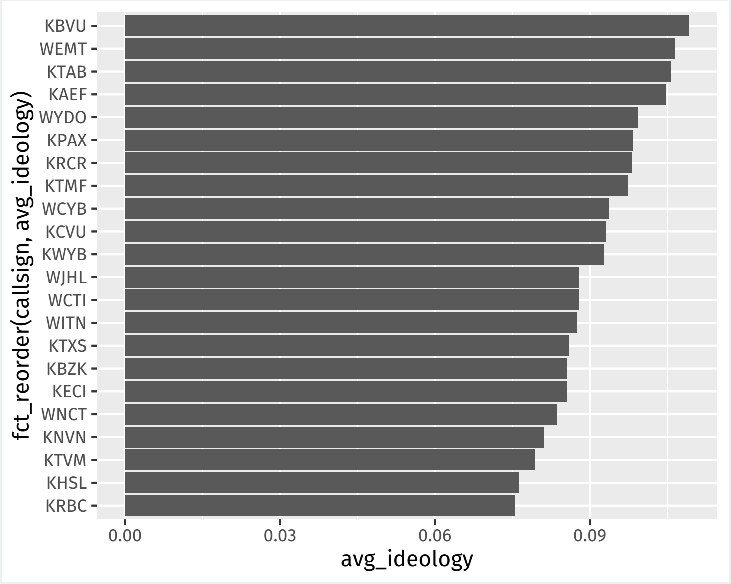


How do we reorder the stations?

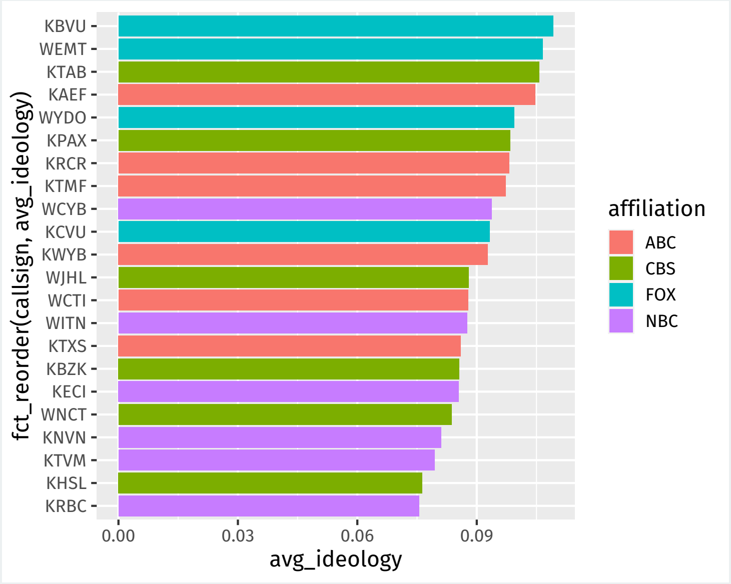
We would like to order the stations by ideology.

`fct_reorder(group, order_var)` function (loaded with tidyverse) will reorder the groups by the order bar (low to high). Easiest to put this in the mapping.

```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col()
```



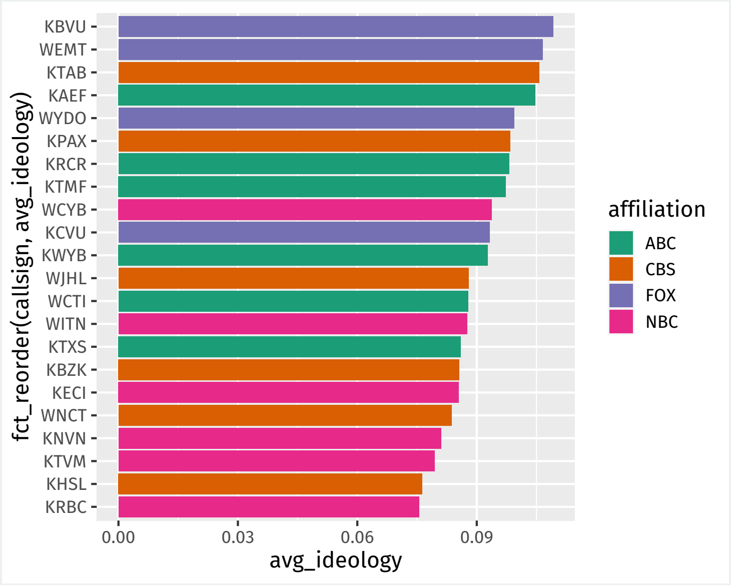
```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation))
```



Setting the color palette

We can use color palettes from a project called ColorBrewer

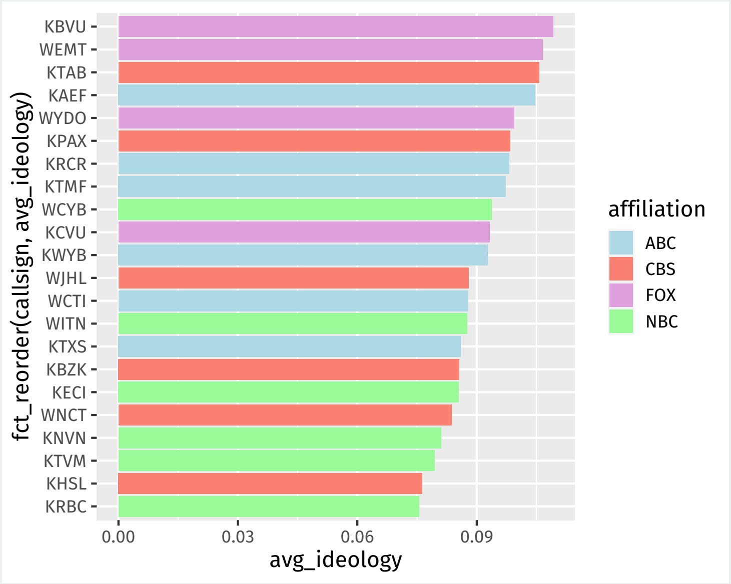
```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_brewer(palette = "Dark2")
```



KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
fct_reorder(callsign, avg_ideology)
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC

Manually setting the color palette

```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = c(ABC = "lightblue",
                               CBS = "salmon",
                               FOX = "plum",
                               NBC = "palegreen"))
```

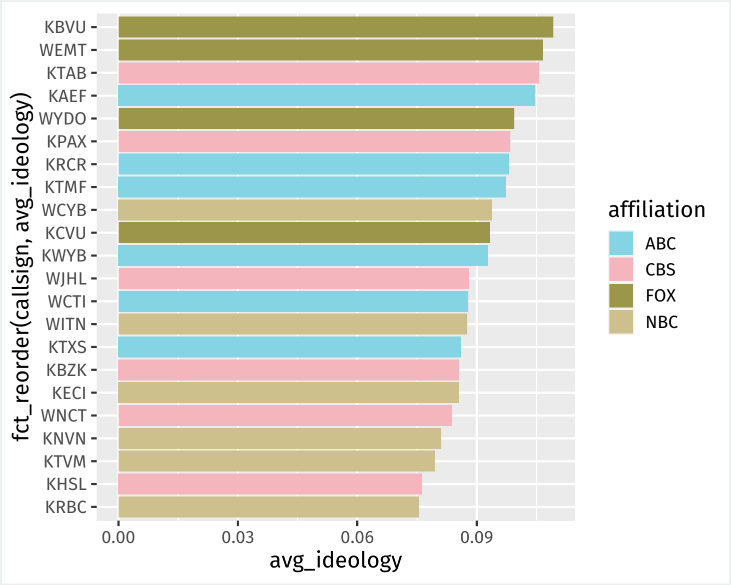


fct_reorder(callsign, avg_ideology)
KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC

Fun with colors

Other packages provide more palettes:

```
library(wesanderson)
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = wes_palette("Moonrise3"))
```


fct_reorder(callsign, avg_ideology)
KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC

Gov 50: 6. Causality

Matthew Blackwell

Harvard University

Roadmap

1. What is causality?
2. Randomized experiments
3. Calculating effects

1/ What is causality?

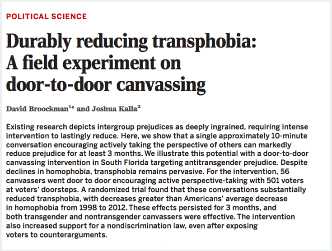
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

What is a causal effect?

factual vs. counterfactual

- Does increasing the minimum wage increase the unemployment rate?
    - Unemployment rate went up after the minimum wage increased
    - Would it have gone up if the minimum wage increase not occurred?
- Does having girls affect a judge's rulings in court?
    - A judge with a daughter gave a pro-choice ruling.
    - Would they have done that if had a son instead?
- **Fundamental problem of causal inference**:
    - Can never observe counterfactuals, must be inferred.

Political canvassing study



POLITICAL SCIENCE

Durably reducing transphobia: A field experiment on door-to-door canvassing

David Broockman[1*] and Joshua Kalla[2]

Existing research depicts intergroup prejudices as deeply ingrained, requiring intense intervention to lastingly reduce. Here, we show that a single approximately 10-minute conversation encouraging actively taking the perspective of others can markedly reduce prejudice for at least 3 months. We illustrate this potential with a door-to-door canvassing intervention in South Florida targeting antitransgender prejudice. Despite declines in homophobia, transphobia remains pervasive. For the intervention, 56 canvassers went door to door encouraging active perspective-taking with 501 voters at voters' doorsteps. A randomized trial found that these conversations substantially reduced transphobia, with decreases greater than Americans' average decrease in homophobia from 1998 to 2012. These effects persisted for 3 months, and both transgender and nontransgender canvassers were effective. The intervention also increased support for a nondiscrimination law, even after exposing voters to counterarguments.

- Can canvassers change minds about topics like transgender rights?
- Experimental setting:
    - Randomly assign canvassers to have a conversation about transgender right or a conversation about recycling.
    - Trans rights conversations focused on "perspective taking"
- Outcome of interest: support for trans rights policies.

Credit: Fabrice Florian via Flickr

A tale of two respondents

	Conversation Script	Support for Nondiscrimination Law
Respondent 1	Recycling	No
Respondent 2	Trans rights	Yes

Did the second respondent support the law **because** of the perspective-taking conversation?

Translating into math

Useful to have **compact** notation for referring to **treatment variable**:

$$T_i = \begin{cases} 1 & \text{if respondent } i \text{ had trans rights conversation} \\ 0 & \text{if respondent } i \text{ had recycling conversation} \end{cases}$$

Similar notation for the **outcome variable**:

$$Y_i = \begin{cases} 1 & \text{if respondent } i \text{ supports trans nondiscrimination laws} \\ 0 & \text{if respondent } i \text{ doesn't support nondiscrimination laws} \end{cases}$$

i is a placeholder to refer to a generic unit/respondent: Y_{42} is the outcome for the 42nd unit.

A tale of two respondents (redux)

	Conversation Script	Support for Nondiscrimination Law
Respondent 1	Recycling	No
Respondent 2	Trans rights	Yes

becomes...

i	T_i	Y_i
Respondent 1	0	0
Respondent 2	1	1

Causal effects & counterfactuals

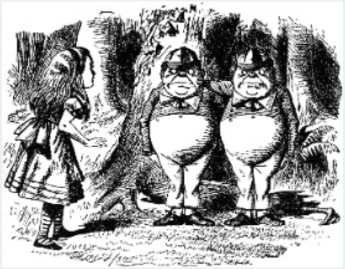
- What does "T_i causes Y_i" mean? $\rightsquigarrow$ **counterfactuals**, "what if"
- Would respondent change their support based on the conversation?
- Two **potential outcomes**:
    - $Y_i(1)$: would respondent *i* support ND laws if they had trans rights script?
    - $Y_i(0)$: would respondent *i* support ND laws if they had recycling script?
- **Causal effect**: $Y_i(1) - Y_i(0)$
    - $Y_i(1) - Y_i(0) = 0 \rightsquigarrow$ script has no effect on policy views
    - $Y_i(1) - Y_i(0) = -1 \rightsquigarrow$ trans rights script lower support for laws
    - $Y_i(1) - Y_i(0) = +1 \rightsquigarrow$ trans rights script increases support for laws

Potential outcomes

i	T_i	Y_i	$Y_i(1)$	$Y_i(0)$
Respondent 1	0	0	???	0
Respondent 2	1	1	1	???

- **Fundamental problem of causal inference**:
  - We only observe one of the two potential outcomes.
  - Observe $Y_i = Y_i(1)$ if $T_i = 1$ or $Y_i = Y_i(0)$ if $T_i = 0$
- To infer causal effect, we need to infer the missing counterfactuals!

How can we figure out counterfactuals?



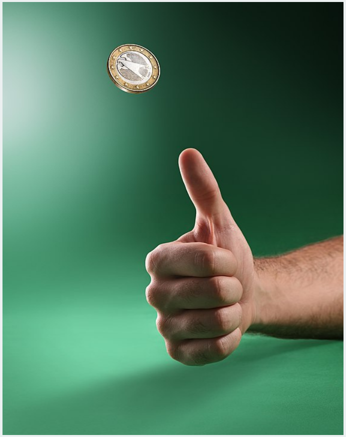
- Find a similar unit! $\rightsquigarrow$ **matching**
    - Mill's method of difference
- Does respondent support law because of the trans rights script?
    - $\rightsquigarrow$ find a identical respondent who got the recycling script.
- NJ increased the minimum wage. Causal effect on unemployment?
    - $\rightsquigarrow$ find a state similar to NJ that didn't increase minimum wage.

Imperfect matches

- The problem: imperfect matches!
- Say we match *i* (treated) and *j* (control)
- **Selection Bias:** $Y_i(1) \neq Y_j(1)$
- Those who take treatment may be different that those who take control.
- How can we correct for that?

2/ Randomized experiments

Match groups not individuals



- **Randomized control trial**: each unit's treatment assignment is determined by chance.
    - Flip a coin; draw red and blue chips from a hat; etc
- Randomization ensures **balance** between treatment and control group.
    - Treatment and control group are identical **on average**
    - Similar on both observable and unobservable characteristics.

A little more notation

- We will often refer to the **sample size** (number of units) as *n*.
- We often have *n* measurements of some variable: $(Y_1, Y_2, \ldots, Y_n)$
- How many in our sample support nondiscrimination laws?

$$Y_1 + Y_2 + Y_3 + \cdots + Y_n$$

- Notation is a bit clunky, so we often use the **Sigma notation**:

$$\sum_{i=1}^{n} Y_i = Y_1 + Y_2 + Y_3 + \cdots + Y_n$$

- $\Sigma_{i=1}^{n}$ means sum each value from Y_1 to Y_n

Averages

- The **sample average** or **sample mean** is simply the sum of all values divided by the number of values.
- Sigma notation allows us to write this in a compact way:

$$\overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

- Suppose we surveyed 6 people and 3 supported nondiscrim. laws:

$$\overline{Y} = \frac{1}{6}(1 + 1 + 1 + 0 + 0 + 0) = 0.5$$

Quantity of interest

- We want to estimate the average causal effects over all units:

$$\text{Sample Average Treatment Effect (SATE)} = \frac{1}{n}\sum_{i=1}^{n}\{Y_i(1) - Y_i(0)\}$$
$$= \frac{1}{n}\sum_{i=1}^{n} Y_i(1) - \frac{1}{n}\sum_{i=1}^{n} Y_i(0)$$

- Why can't we just calculate this quantity directly?
- What we can estimate instead:

$$\text{Difference in means} = \overline{Y}_{\text{treated}} - \overline{Y}_{\text{control}}$$

  - $\overline{Y}_{\text{treated}}$: sample average outcome for treated group
  - $\overline{Y}_{\text{control}}$: sample average outcome for control group
- When will the difference-in-means is a good estimate of the SATE?

Why randomization works

- Under an RCT, treatment and control groups are random samples.
- Average in the treatment group will be similar to average if all treated:

$$\overline{Y}_{\text{treated}} \approx \frac{1}{n}\sum_{i=1}^{n} Y_i(1)$$

- Average in the control group will be similar to average if all untreated:

$$\overline{Y}_{\text{control}} \approx \frac{1}{n}\sum_{i=1}^{n} Y_i(0)$$

- Implies difference-in-means should be close to SATE:

$$\overline{Y}_{\text{treated}} - \overline{Y}_{\text{control}} \approx \frac{1}{n}\sum_{i=1}^{n} Y_i(1) - \frac{1}{n}\sum_{i=1}^{n} Y_i(0) = \frac{1}{n}\sum_{i=1}^{n}\{Y_i(1) - Y_i(0)\} = \text{SATE}$$

Some potential problems with RCTs

- **Placebo effects**:
    - Respondents will be affected by any intervention, even if they shouldn't have any effect.
    - Reason to have control group be recycling script
- **Hawthorne effects**:
    - Respondents act differently just knowing that they are under study.

Balance checking

- Can we determine if randomization “worked”?
- If it did, we shouldn’t see large differences between treatment and control group on **pretreatment variable**.
  - Pretreatment variable are those that are unaffected by treatment.
- We can check in the actual data for some pretreatment variable X
  - $\overline{X}_{\text{treated}}$: average value of variable for treated group.
  - $\overline{X}_{\text{control}}$: average value of variable for control group.
  - Under randomization, $\overline{X}_{\text{treated}} - \overline{X}_{\text{control}} \approx 0$

Multiple treatments

- Instead of 1 treatment, we might have multiple **treatment arms**:
  - Control condition
  - Treatment A
  - Treatment B
  - Treatment C, etc
- In this case, we will look at multiple comparisons:
  - $\overline{Y}_{\text{treated, A}} - \overline{Y}_{\text{control}}$
  - $\overline{Y}_{\text{treated, B}} - \overline{Y}_{\text{control}}$
  - $\overline{Y}_{\text{treated, A}} - \overline{Y}_{\text{treated, B}}$
- If treatment arms are randomly assigned, these differences will be good estimators for each causal contrast.

3/ Calculating effects

Transphobia study data

```r
## reinstall gov50data if necessary
library(gov50data)
```

Variable Name	Description
`age`	Age of the R in years
`female`	1=R marked “Female” on voter reg., 0 otherwise
`voted_gen_14`	1 if R voted in the 2014 general election
`vote_gen_12`	1 if R voted in the 2012 general election
`treat_ind`	1 if R assigned to trans rights script, 0 for recycling
`racename`	name of racial identity indicated on voter file
`democrat`	1 if R is a registered Democrat
`nondiscrim_pre`	1 if R supports nondiscrim. law at baseline
`nondiscrim_post`	1 if R supports nondiscrim. law after 3 months

Peak at the data

```r
trans
```

```
## # A tibble: 565 x 9
##      age female voted_gen_14 voted~1 treat~2 racen~3 democ~4
##    <dbl>  <dbl>        <dbl>   <dbl>   <dbl> <chr>     <dbl>
##  1    29      0            0       1       0 Africa~       1
##  2    59      1            1       0       1 Africa~       1
##  3    35      1            1       1       1 Africa~       1
##  4    63      1            1       1       1 Africa~       1
##  5    65      0            1       1       1 Africa~       0
##  6    51      1            1       1       0 Caucas~       0
##  7    26      1            1       1       0 Africa~       1
##  8    62      1            1       1       1 Africa~       1
##  9    37      0            1       1       0 Caucas~       0
## 10    51      1            1       1       0 Caucas~       0
## # ... with 555 more rows, 2 more variables:
## #   nondiscrim_pre <dbl>, nondiscrim_post <dbl>, and
## #   abbreviated variable names 1: voted_gen_12,
## #   2: treat_ind, 3: racename, 4: democrat
```

Calculate the average outcomes in each group

```r
treat_mean <- trans |>
  filter(treat_ind == 1) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post))
treat_mean
```

```
## # A tibble: 1 x 1
##   nondiscrim_mean
##             <dbl>
## 1           0.687
```

```r
control_mean <- trans |>
  filter(treat_ind == 0) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post))
control_mean
```

```
## # A tibble: 1 x 1
##   nondiscrim_mean
##             <dbl>
## 1           0.648
```

Calculating the difference in means

```
treat_mean - control_mean
```

```
##   nondiscrim_mean
## 1           0.039
```

We'll see more ways to do this throughout the semester.

Checking balance on numeric covariates

We can use group_by to see how the mean of covariates varies by group:

```
trans |>
  group_by(treat_ind) |>
  summarize(age_mean = mean(age))
```

```
## # A tibble: 2 x 2
##   treat_ind age_mean
##       <dbl>    <dbl>
## 1         0     48.2
## 2         1     48.3
```

Checking balance on categorical covariates

Or we can group by treatment and a categorical control:

```
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n())
```

```
## # A tibble: 9 x 3
## # Groups:   treat_ind [2]
##   treat_ind racename             n
##       <dbl> <chr>            <int>
## 1         0 African American    58
## 2         0 Asian                2
## 3         0 Caucasian           77
## 4         0 Hispanic           150
## 5         1 African American    68
## 6         1 Asian                4
## 7         1 Caucasian           75
## 8         1 Hispanic           130
## 9         1 Native American      1
```

Hard to read!

pivot_wider

pivot_wider( ) takes data from a single column and moves it into multiple columns based on a grouping variable:

```r
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n()) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = n
  )
```

names_from tells us what variable will map onto the columns

values_from tell us what values should be mapped into those columns

```r
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n()) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = n
  )
```

```
## # A tibble: 5 x 3
##   racename           `0`   `1`
##   <chr>            <int> <int>
## 1 African American    58    68
## 2 Asian                2     4
## 3 Caucasian           77    75
## 4 Hispanic           150   130
## 5 Native American     NA     1
```

Calculating diff-in-means by group

```r
trans |>
  mutate(
    treat_ind = if_else(treat_ind == 1, "Treated", "Control"),
    party = if_else(democrat == 1, "Democrat", "Non-Democrat")
  ) |>
  group_by(treat_ind, party) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post)) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = nondiscrim_mean
  ) |>
  mutate(
    diff_in_means = Treated - Control
  )
```

```
## # A tibble: 2 x 4
##   party        Control Treated diff_in_means
##   <chr>          <dbl>   <dbl>         <dbl>
## 1 Democrat       0.704   0.754        0.0498
## 2 Non-Democrat   0.605   0.628        0.0234
## # A tibble: 2 x 4
##   party        Control Treated diff_in_means
##   <chr>          <dbl>   <dbl>         <dbl>
## 1 Democrat       0.704   0.754        0.0498
## 2 Non-Democrat   0.605   0.628        0.0234
```

Gov 50: 7. Observational Studies

Matthew Blackwell

Harvard University

Roadmap

1. Calculating effects

2. Observational Studies

1/ Calculating effects

Transphobia study data

```r
## reinstall gov50data if necessary
library(gov50data)
```

Variable Name	Description
`age`	Age of the R in years
`female`	1=R marked “Female” on voter reg., 0 otherwise
`voted_gen_14`	1 if R voted in the 2014 general election
`vote_gen_12`	1 if R voted in the 2012 general election
`treat_ind`	1 if R assigned to trans rights script, 0 for recycling
`racename`	name of racial identity indicated on voter file
`democrat`	1 if R is a registered Democrat
`nondiscrim_pre`	1 if R supports nondiscrim. law at baseline
`nondiscrim_post`	1 if R supports nondiscrim. law after 3 months

Peak at the data

```
trans
```

```
## # A tibble: 565 x 9
##      age female voted_gen_14 voted~1 treat~2 racen~3 democ~4
##    <dbl>  <dbl>        <dbl>   <dbl>   <dbl> <chr>     <dbl>
##  1    29      0            0       1       0 Africa~       1
##  2    59      1            1       0       1 Africa~       1
##  3    35      1            1       1       1 Africa~       1
##  4    63      1            1       1       1 Africa~       1
##  5    65      0            1       1       1 Africa~       0
##  6    51      1            1       1       0 Caucas~       0
##  7    26      1            1       1       0 Africa~       1
##  8    62      1            1       1       1 Africa~       1
##  9    37      0            1       1       0 Caucas~       0
## 10    51      1            1       1       0 Caucas~       0
## # ... with 555 more rows, 2 more variables:
## #   nondiscrim_pre <dbl>, nondiscrim_post <dbl>, and
## #   abbreviated variable names 1: voted_gen_12,
## #   2: treat_ind, 3: racename, 4: democrat
```

Calculate the average outcomes in each group

```r
treat_mean <- trans |>
  filter(treat_ind == 1) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post))
treat_mean
```

```
## # A tibble: 1 x 1
##   nondiscrim_mean
##             <dbl>
## 1           0.687
```

```r
control_mean <- trans |>
  filter(treat_ind == 0) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post))
control_mean
```

```
## # A tibble: 1 x 1
##   nondiscrim_mean
##             <dbl>
## 1           0.648
```

Calculating the difference in means

```
treat_mean - control_mean
```

```
##   nondiscrim_mean
## 1           0.039
```

We’ll see more ways to do this throughout the semester.

Checking balance on numeric covariates

We can use `group_by` to see how the mean of covariates varies by group:

```r
trans |>
  group_by(treat_ind) |>
  summarize(age_mean = mean(age))
```

```
## # A tibble: 2 x 2
##   treat_ind age_mean
##       <dbl>    <dbl>
## 1         0     48.2
## 2         1     48.3
```

Checking balance on categorical covariates

Or we can group by treatment and a categorical control:

```
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n())
```

```
## # A tibble: 9 x 3
## # Groups:   treat_ind [2]
##   treat_ind racename             n
##       <dbl> <chr>            <int>
## 1         0 African American    58
## 2         0 Asian                2
## 3         0 Caucasian           77
## 4         0 Hispanic           150
## 5         1 African American    68
## 6         1 Asian                4
## 7         1 Caucasian           75
## 8         1 Hispanic           130
## 9         1 Native American      1
```

Hard to read!

pivot_wider

pivot_wider() takes data from a single column and moves it into multiple columns based on a grouping variable:

```
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n()) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = n
  )
```

names_from tells us what variable will map onto the columns

values_from tell us what values should be mapped into those columns

```r
trans |>
  group_by(treat_ind, racename) |>
  summarize(n = n()) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = n
  )
```

```
## # A tibble: 5 x 3
##   racename            `0`   `1`
##   <chr>             <int> <int>
## 1 African American     58    68
## 2 Asian                 2     4
## 3 Caucasian            77    75
## 4 Hispanic            150   130
## 5 Native American      NA     1
```

Calculating diff-in-means by group

```
trans |>
  mutate(
    treat_ind = if_else(treat_ind == 1, "Treated", "Control"),
    party = if_else(democrat == 1, "Democrat", "Non-Democrat")
  ) |>
  group_by(treat_ind, party) |>
  summarize(nondiscrim_mean = mean(nondiscrim_post)) |>
  pivot_wider(
    names_from = treat_ind,
    values_from = nondiscrim_mean
  ) |>
  mutate(
    diff_in_means = Treated - Control
  )
```

```
## # A tibble: 2 x 4
##   party        Control Treated diff_in_means
##   <chr>          <dbl>   <dbl>         <dbl>
## 1 Democrat       0.704   0.754        0.0498
## 2 Non-Democrat   0.605   0.628        0.0234
## # A tibble: 2 x 4
##   party        Control Treated diff_in_means
##   <chr>          <dbl>   <dbl>         <dbl>
## 1 Democrat       0.704   0.754        0.0498
## 2 Non-Democrat   0.605   0.628        0.0234
```

2/ Observational Studies

Do newspaper endorsements matter?

- Can newspaper endorsements change voters' minds?
- Why not compare vote choice of readers of different papers?
  - Problem: readers choose papers based on their previous beliefs.
  - Liberals $\rightsquigarrow$ New York Times, conservatives $\rightsquigarrow$ Wall Street Journal.
- Study for today: British newspapers switching their endorsements.
  - Some newspapers endorsing Tories in 1992 switched to Labour in 1997.
  - **Treated group**: readers of Tory $\rightarrow$ Labour papers.
  - **Control group**: readers of papers who didn't switch.

Data

Name	Description
`to_labour`	Read a newspaper that switched endorsement to Labour between 1992 and 1997 (1=Yes, 0=No)?
`vote_lab_92`	Did respondent vote for Labour in 1992 election (1=Yes, 0=No)?
`vote_lab_97`	Did respondent vote for Labour in 1997 election (1=Yes, 0=No)?
`age`	Age of respondent
`male`	Does the respondent identify as Male (1=Yes, 0=No)?
`parent_labour`	Did the respondent's parents vote for Labour (1=Yes, 0=No)?
`work_class`	Does the respondent identify as working class (1=Yes, 0=No)?

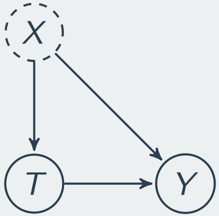
```r
library(tidyverse)
library(gov50data)
newspapers
```

```
## # A tibble: 1,593 x 7
##    to_labour vote_lab_92 vote_~1 age    male paren~2 work_~3
##        <dbl>       <dbl>   <dbl> <dbl> <dbl>   <dbl>   <dbl>
##  1         0           1       1 33        0       1       1
##  2         0           1       0 51        0       1       0
##  3         0           0       0 46        0       1       1
##  4         0           1       1 45        1       1       1
##  5         0           1       1 29        0       1       1
##  6         0           1       1 47        1       1       1
##  7         0           1       1 34        1       0       1
##  8         0           1       1 31        0       1       1
##  9         0           1       1 24        1       1       1
## 10         1           1       1 48        0       1       1
## # ... with 1,583 more rows, and abbreviated variable names
## #   1: vote_lab_97, 2: parent_labour, 3: work_class
```

Observational studies

- Example of an **observational study**:
    - We as researchers observe a naturally assigned treatment
    - Very common: often can't randomize for ethical/logistical reasons.
- **Internal validity**: are the causal assumption satisfied? Can we interpret this as a causal effect?
    - RCTs usually have higher internal validity.
    - Observational studies less so because treatment and control groups may differ in ways that are hard to measure
- **External validity**: can the conclusions/estimated effects be generalized beyond this study?
    - RCTs weaker here because often very expensive to conduct on representative samples.
    - Observational studies often have larger/more representative samples that improve external validity.

Confounding


$$\begin{tikzcd} & X \arrow[dl] \arrow[dr] & \\ T \arrow[rr] & & Y \end{tikzcd}$$

- **Confounder**: pre-treatment variable affecting treatment & the outcome.
    - Leftists (X) more likely to read newspapers switching to Labour (T).
    - Leftists (X) also more likely to vote for Labour (Y).
- **Confounding bias** in the estimated SATE due to these differences
    - $\overline{Y}_{\text{control}}$ not a good proxy for $\frac{1}{n}\sum_{i=1}^{n} Y_i(0)$ in treated group.
    - one type: **selection bias** from self-selection into treatment

Research designs

- How can we find a good comparison group?
- Depends on the data we have available.
- Three general types of observational study **reseach designs**:
    1. **Cross-sectional design**: compare outcomes treated and control units at one point in time.
    2. **Before-and-after design**: compare outcomes before and after a unit has been treated, but need over-time data on treated group.
    3. **Difference-in-differences design**: use before/after information for the treated and control group; need over-time on treated & control group.

Cross-sectional design

- Compare treatment and control groups after treatment happens.
    - Readers of switching papers vs readers of non-switching papers in 1997.
- Treatment & control groups assumed identical on average as in RCT.
    - Sometimes called **unconfoundedness** or **as-if randomized**.
- Cross-section comparison estimate:

$$\overline{Y}^{\text{after}}_{\text{treated}} - \overline{Y}^{\text{after}}_{\text{control}}$$

- Could there be confounders?

Cross-sectional design in R

```r
switched <- newspapers |>
  filter(to_labour == 1) |>
  summarize(mean(vote_lab_97))


no_change <- newspapers %>%
  filter(to_labour == 0) |>
  summarize(mean(vote_lab_97))

switched - no_change
```

```
##   mean(vote_lab_97)
## 1              0.14
```

Statistical control

- **Statistical control**: adjust for confounders using statistical procedures.
    - Can help to reduce confounding bias.
- One type of statistical control: **subclassification**
    - Compare treated and control groups within levels of a confounder.
    - Remaining effect can't be due to the confounder.
- Threat to inference: we can only control for observed variables $\rightsquigarrow$ threat of **unmeasured confounding**

Statistical control in R

```r
newspapers %>%
  group_by(parent_labour, to_labour) %>%
  summarize(avg_vote = mean(vote_lab_97)) %>%
  pivot_wider(
    names_from = to_labour,
    values_from = avg_vote
  ) %>%
  mutate(diff_by_parent = `1` - `0`)
```

```
## # A tibble: 2 x 4
## # Groups:   parent_labour [2]
##   parent_labour   `0`   `1` diff_by_parent
##           <dbl> <dbl> <dbl>          <dbl>
## 1             0 0.279 0.434          0.155
## 2             1 0.597 0.698          0.101
```

Before-and-after comparison

- Compare readers of party-switching newspapers before & after switch.
- Advantage: all person-specific features held fixed
    - comparing within a person over time.
- Before-and-after estimate:

$$\overline{Y}^{\text{after}}_{\text{treated}} - \overline{Y}^{\text{before}}_{\text{treated}}$$

- Threat to inference: **time-varying confounders**
    - Time trend: Labour just did better overall in 1997 compared to 1992.

Before and after in R

```r
newspapers |>
  mutate(
    vote_change = vote_lab_97 - vote_lab_92
  ) |>
  summarize(avg_change = mean(vote_change))
```

```
## # A tibble: 1 x 1
##   avg_change
##        <dbl>
## 1      0.119
```

Differences in differences

- Key idea: use the before-and-after difference of **control group** to infer what would have happend to **treatment group** without treatment.
- DiD estimate:

$$\underbrace{\left(\overline{Y}_{\text{treated}}^{\text{after}} - \overline{Y}_{\text{treated}}^{\text{before}}\right)}_{\text{trend in treated group}} - \underbrace{\left(\overline{Y}_{\text{control}}^{\text{after}} - \overline{Y}_{\text{control}}^{\text{before}}\right)}_{\text{trend in control group}}$$

- Change in treated group above and beyond the change in control group.
- **Parallel time trend assumption**
  - Changes in vote of readers of non-switching papers roughly the same as changes that readers of switching papers would have been if they read non-switching papers.
  - Threat to inference: non-parallel trends.

Difference-in-differences in R

```r
newspapers |>
  mutate(
    vote_change = vote_lab_97 - vote_lab_92,
    to_labour = if_else(to_labour == 1, "switched", "unswitched")
  ) |>
  group_by(to_labour) |>
  summarize(avg_change = mean(vote_change)) |>
  pivot_wider(
    names_from = to_labour,
    values_from = avg_change
  ) |>
  mutate(DID = switched - unswitched)
```

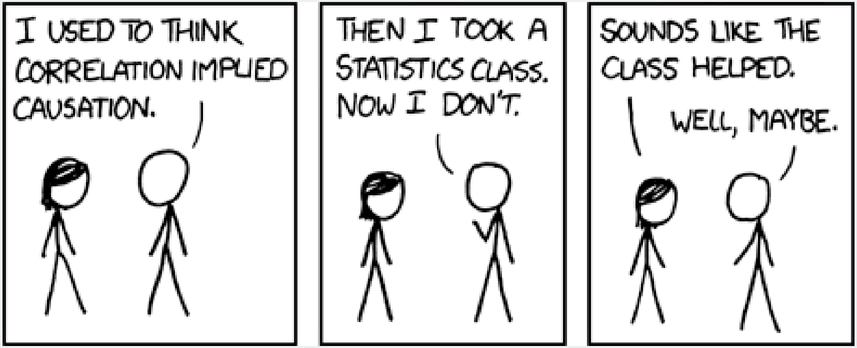
```
## # A tibble: 1 x 3
##   switched unswitched    DID
##      <dbl>      <dbl>  <dbl>
## 1    0.190      0.110 0.0796
```

Summarizing approaches

1. **Cross-sectional comparison**
    - Compare treated units with control units after treatment
    - Assumption: treated and controls units are comparable
    - Possible confounding
2. **Before-and-after comparison**
    - Compare the same units before and after treatment
    - Assumption: no time-varying confounding
3. **Differences-in-differences**
    - Assumption: parallel trends assumptions
    - Under this assumption, it accounts for unit-specific and time-varying confounding.

- All rely on assumptions that can't be verified to handle confounding.
- RCTs handle confounding by design.

Causality understanding check


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Gov 50: 8. Summarizing Data

Matthew Blackwell

Harvard University

Roadmap

1. Descriptive Statistics
2. Missing data
3. Proportion tables

1/ Descriptive Statistics

Lots of data

```r
library(tidyverse)
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

Lots and lots of data

```r
head(gapminder$gdpPercap, n = 200)
```

```
##   [1]   779   821   853   836   740   786   978   852   649
##  [10]   635   727   975  1601  1942  2313  2760  3313  3533
##  [19]  3631  3739  2497  3193  4604  5937  2449  3014  2551
##  [28]  3247  4183  4910  5745  5681  5023  4797  5288  6223
##  [37]  3521  3828  4269  5523  5473  3009  2757  2430  2628
##  [46]  2277  2773  4797  5911  6857  7133  8053  9443 10079
##  [55]  8998  9140  9308 10967  8798 12779 10040 10950 12217
##  [64] 14526 16789 18334 19477 21889 23425 26998 30688 34435
##  [73]  6137  8843 10751 12835 16662 19749 21597 23688 27042
##  [82] 29096 32418 36126  9867 11636 12753 14805 18269 19340
##  [91] 19211 18524 19036 20292 23404 29796   684   662   686
## [100]   721   630   660   677   752   838   973  1136  1391
## [109]  8343  9715 10991 13149 16672 19118 20980 22526 25576
## [118] 27561 30486 33693  1063   960   949  1036  1086  1029
## [127]  1278  1226  1191  1233  1373  1441  2677  2128  2181
## [136]  2587  2980  3548  3157  2754  2962  3326  3413  3822
## [145]   974  1354  1710  2172  2860  3528  4127  4314  2547
## [154]  4766  6019  7446   851   918   984  1215  2264  3215
## [163]  4551  6206  7954  8647 11004 12570  2109  2487  3337
## [172]  3430  4986  6660  7031  7807  6950  7958  8131  9066
```

How to summarize data

- How should we summarize the wages data? Many possibilities!
    - Up to now: focus on **averages** or means of variables.
- Two salient features of a variable that we want to know:
    - **Central tendency**: where is the middle/typical/average value.
    - **Spread** around the center: are all values to the center or spread out?

Center of the data

- "Center" of the data: typical/average value.
- **Mean**: sum of the values divided by the number of observations

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- **Median**:

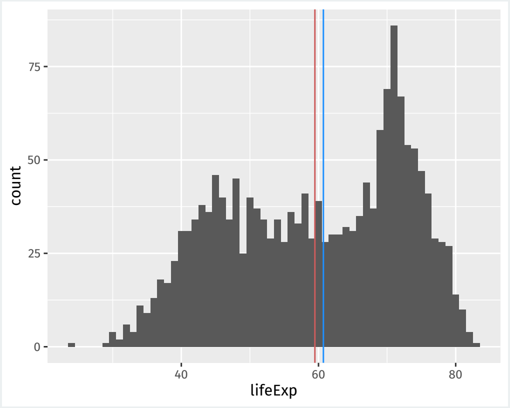
$$\text{median} = \begin{cases} \text{middle value} & \text{if number of entries is odd} \\ \frac{\text{sum of two middle values}}{2} & \text{if number of entries is even} \end{cases}$$

- In **R**: `mean()` and `median()`.

Mean vs median

- Median more robust to **outliers**:
  - Example 1: data = $\{0, 1, 2, 3, 5\}$. Mean? Median?
  - Example 2: data = $\{0, 1, 2, 3, 100\}$. Mean? Median?
- What does Mark Zuckerberg do to the mean vs median income?

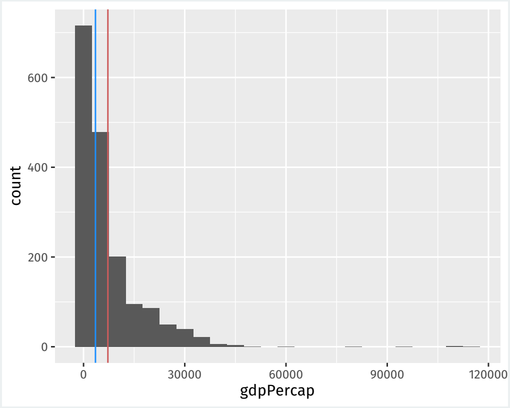
```r
ggplot(gapminder, aes(x = lifeExp)) +
  geom_histogram(binwidth = 1) +
  geom_vline(aes(xintercept = mean(lifeExp)), color = "indianred") +
  geom_vline(aes(xintercept = median(lifeExp)), color = "dodgerblue")
```



```r
summary(gapminder$lifeExp)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     23.6    48.2    60.7    59.5    70.8    82.6
```

```r
ggplot(gapminder, aes(x = gdpPercap)) +
  geom_histogram(binwidth = 5000) +
  geom_vline(aes(xintercept = mean(gdpPercap)), color = "indianred") +
  geom_vline(aes(xintercept = median(gdpPercap)), color = "dodgerblue")
```

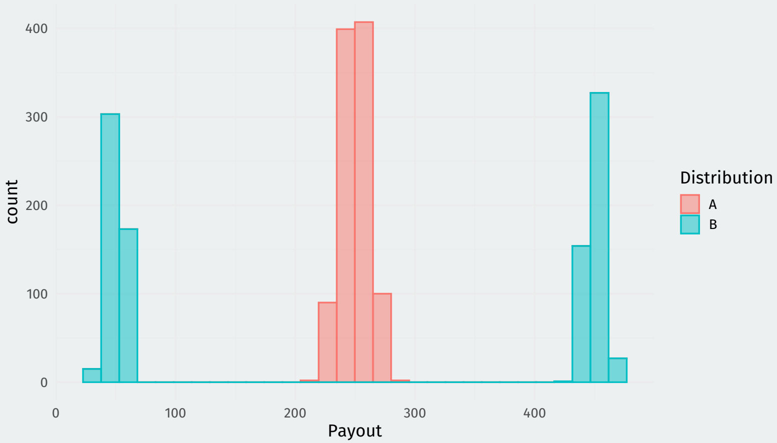


```r
summary(gapminder$gdpPercap)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      241     1202     3532     7215     9325   113523
```

Which distribution would you prefer?

Lottery where we randomly draw one value from A or B:



They have the same mean, so why do we care about the difference? **Spread!!**

Spread of the data

- Are the values of the variable close to the center?
- **Range**: $[\min(X),\ \max(X)]$
- **Quantile** (quartile, percentile, etc): divide data into equal sized groups.
  - 25th percentile = lower quartile (25% of the data below this value)
  - 50th percentile = median (50% of the data below this value)
  - 75th percentile = upper quartile (75% of the data below this value)
- **Interquartile range** (IQR): a measure of variability
  - How spread out is the middle half of the data?
  - Is most of the data really close to the median or are the values spread out?
- **R** function: `range()`, `summary()`, `IQR()`

Standard deviation

- **Standard deviation**: On average, how far away are data points from the mean?

$$\text{standard deviation} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

- Steps:

  1. Subtract each data point by the mean.
  2. Square each resulting difference.
  3. Take the sum of these values
  4. Divide by $n - 1$ (or n, doesn't matter much)
  5. Take the square root.

- **Variance** = standard deviation^2

- Why not just take the average deviations from mean without squaring?

2/ Missing data

Missing data

- **Nonresponse**: respondent can't or won't answer question.
    - Sensitive questions $\rightsquigarrow$ **social desirability bias**
    - Some countries lack official statistics like unemployment.
    - Leads to missing data.
- Missing data in R: a special value `NA`
- Have already seen how to use `na.rm = TRUE`

CCES data

```r
library(gov50data)
cces_2020
```

```
## # A tibble: 51,551 x 6
##    gender race  educ                 pid3    turno~1 pres_~2
##    <fct>  <fct> <fct>                <fct>     <dbl> <fct>  
##  1 Male   White 2-year               Republ~       1 Donald~
##  2 Female White Post-grad            Democr~      NA <NA>   
##  3 Female White 4-year               Indepe~       1 Joe Bi~
##  4 Female White 4-year               Democr~       1 Joe Bi~
##  5 Male   White 4-year               Indepe~       1 Other  
##  6 Male   White Some college         Republ~       1 Donald~
##  7 Male   Black Some college         Not su~      NA <NA>   
##  8 Female White Some college         Indepe~       1 Donald~
##  9 Female White High school graduate Republ~       1 Donald~
## 10 Female White 4-year               Democr~       1 Joe Bi~
## # ... with 51,541 more rows, and abbreviated variable names
## #   1: turnout_self, 2: pres_vote
```

drop_na() to remove rows with missing values

```r
cces_2020 |>
  drop_na()
```

```
## # A tibble: 45,651 x 6
##    gender race  educ                 pid3    turno~1 pres_~2
##    <fct>  <fct> <fct>                <fct>     <dbl> <fct>
##  1 Male   White 2-year               Republ~       1 Donald~
##  2 Female White 4-year               Indepe~       1 Joe Bi~
##  3 Female White 4-year               Democr~       1 Joe Bi~
##  4 Male   White 4-year               Indepe~       1 Other
##  5 Male   White Some college         Republ~       1 Donald~
##  6 Female White Some college         Indepe~       1 Donald~
##  7 Female White High school graduate Republ~       1 Donald~
##  8 Female White 4-year               Democr~       1 Joe Bi~
##  9 Female White 4-year               Democr~       1 Joe Bi~
## 10 Female White 4-year               Democr~       1 Joe Bi~
## # ... with 45,641 more rows, and abbreviated variable names
## #   1: turnout_self, 2: pres_vote
```

Drop rows based on certain variables

```r
cces_2020 |>
  dim_desc()
```

```
## [1] "[51,551 x 6]"
```

```r
cces_2020 |>
  drop_na() |>
  dim_desc()
```

```
## [1] "[45,651 x 6]"
```

```r
cces_2020 |>
  drop_na(turnout_self) |>
  dim_desc()
```

```
## [1] "[48,462 x 6]"
```

Available-case vs complete-case analysis

Available-case analysis: use the data you have for that variable:

```
cces_2020 |>
  summarize(mean(turnout_self, na.rm = TRUE)) |>
  pull()
```

```
## [1] 0.942
```

Complete-case analysis: only use units that have data on all variables

```
cces_2020 |>
  drop_na() |>
  summarize(mean(turnout_self)) |>
  pull()
```

```
## [1] 0.999
```

(also called **listwise deletion**)

`is.na()` to detect missingness

Trying to detect missingness with `==` doesn't work:

```
c(5, 6, NA, 0) == NA
```

```
## [1] NA NA NA NA
```

Use `is.na()` instead:

```
is.na(c(5, 6, NA, 0))
```

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

Can use `sum()` or `mean()` on this to get number/proportion missing:

```
sum(is.na(c(5, 6, NA, 0)))
```

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

Nonresponse bias

Nonresponse can create bias if lower turnout ⇨ more non-response:

```r
cces_2020 |>
  group_by(pid3) |>
  summarize(
    mean_turnout = mean(turnout_self, na.rm = TRUE),
    missing_turnout = mean(is.na(turnout_self))
  )
```

```
## # A tibble: 5 x 3
##   pid3        mean_turnout missing_turnout
##   <fct>              <dbl>           <dbl>
## 1 Democrat           0.963          0.0280
## 2 Republican         0.953          0.0403
## 3 Independent        0.924          0.0718
## 4 Other              0.957          0.0709
## 5 Not sure           0.630          0.431
```

3/ Proportion tables

Review of getting counts

First, let's review how to get counts:

```r
cces_2020 |>
  group_by(pres_vote) |>
  summarize(n = n())
```

```
## # A tibble: 7 x 2
##   pres_vote                        n
##   <fct>                        <int>
## 1 Joe Biden (Democrat)         26188
## 2 Donald J. Trump (Republican) 17702
## 3 Other                         1458
## 4 I did not vote in this race    100
## 5 I did not vote                  13
## 6 Not sure                       190
## 7 <NA>                          5900
```

First attempt to create proportions

```r
cces_2020 |>
  group_by(pres_vote) |>
  summarize(prop = n() / sum(n()))
```

```
## # A tibble: 7 x 2
##   pres_vote                       prop
##   <fct>                          <dbl>
## 1 Joe Biden (Democrat)               1
## 2 Donald J. Trump (Republican)       1
## 3 Other                              1
## 4 I did not vote in this race        1
## 5 I did not vote                     1
## 6 Not sure                           1
## 7 <NA>                               1
```

Inside `summarize()` all operations are done within groups!

Mutate after summarizing

```r
cces_2020 |>
  group_by(pres_vote) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n))
```

```
## # A tibble: 7 x 3
##   pres_vote                          n     prop
##   <fct>                          <int>    <dbl>
## 1 Joe Biden (Democrat)           26188 0.508
## 2 Donald J. Trump (Republican) 17702 0.343
## 3 Other                           1458 0.0283
## 4 I did not vote in this race     100 0.00194
## 5 I did not vote                   13 0.000252
## 6 Not sure                         190 0.00369
## 7 <NA>                            5900 0.114
```

Grouping is silently dropped after `summarize()`

Multiple grouping variables

What happens with multiple grouping variables

```
cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  group_by(pid3, pres_vote) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n))
```

```
## # A tibble: 10 x 4
## # Groups:   pid3 [5]
##    pid3        pres_vote                         n   prop
##    <fct>       <fct>                         <int>  <dbl>
##  1 Democrat    Joe Biden (Democrat)          17649 0.968
##  2 Democrat    Donald J. Trump (Republican)    581 0.0319
##  3 Republican  Joe Biden (Democrat)            856 0.0712
##  4 Republican  Donald J. Trump (Republican) 11164 0.929
##  5 Independent Joe Biden (Democrat)           6601 0.571
##  6 Independent Donald J. Trump (Republican)  4951 0.429
##  7 Other       Joe Biden (Democrat)            735 0.487
##  8 Other       Donald J. Trump (Republican)    774 0.513
##  9 Not sure    Joe Biden (Democrat)            347 0.599
## 10 Not sure    Donald J. Trump (Republican)    232 0.401
```

With multiple grouping variables, `summarize()` drops the last one.

Dropping all groups

If we want the proportion of all rows, need to drop all groups.

```
cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  group_by(pid3, pres_vote) |>
  summarize(n = n(), .groups = "drop") |>
  mutate(prop = n / sum(n))
```

```
## # A tibble: 10 x 4
##    pid3        pres_vote                       n    prop
##    <fct>       <fct>                       <int>   <dbl>
##  1 Democrat    Joe Biden (Democrat)        17649 0.402
##  2 Democrat    Donald J. Trump (Republican)  581 0.0132
##  3 Republican  Joe Biden (Democrat)          856 0.0195
##  4 Republican  Donald J. Trump (Republican) 11164 0.254
##  5 Independent Joe Biden (Democrat)         6601 0.150
##  6 Independent Donald J. Trump (Republican)  4951 0.113
##  7 Other       Joe Biden (Democrat)          735 0.0167
##  8 Other       Donald J. Trump (Republican)  774 0.0176
##  9 Not sure    Joe Biden (Democrat)          347 0.00791
## 10 Not sure    Donald J. Trump (Republican)  232 0.00529
```

Gov 50: 9. Survey Sampling

Matthew Blackwell

Harvard University

Roadmap

1. Proportion tables

2. Measurement

1/ Proportion tables

CCES Data

```r
library(gov50data)
cces_2020
```

```
## # A tibble: 51,551 x 6
##    gender race  educ                 pid3    turno~1 pres_~2
##    <fct>  <fct> <fct>                <fct>     <dbl> <fct>
##  1 Male   White 2-year               Republ~       1 Donald~
##  2 Female White Post-grad            Democr~      NA <NA>
##  3 Female White 4-year               Indepe~       1 Joe Bi~
##  4 Female White 4-year               Democr~       1 Joe Bi~
##  5 Male   White 4-year               Indepe~       1 Other
##  6 Male   White Some college         Republ~       1 Donald~
##  7 Male   Black Some college         Not su~      NA <NA>
##  8 Female White Some college         Indepe~       1 Donald~
##  9 Female White High school graduate Republ~       1 Donald~
## 10 Female White 4-year               Democr~       1 Joe Bi~
## # ... with 51,541 more rows, and abbreviated variable names
## #   1: turnout_self, 2: pres_vote
```

Mutate after summarizing

```r
cces_2020 |>
  group_by(pres_vote) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n))
```

```
## # A tibble: 7 x 3
##   pres_vote                           n     prop
##   <fct>                           <int>    <dbl>
## 1 Joe Biden (Democrat)            26188 0.508
## 2 Donald J. Trump (Republican)    17702 0.343
## 3 Other                            1458 0.0283
## 4 I did not vote in this race       100 0.00194
## 5 I did not vote                     13 0.000252
## 6 Not sure                          190 0.00369
## 7 <NA>                             5900 0.114
```

Another approach

```r
cces_2020 |>
  group_by(pres_vote) |>
  summarize(prop = n() / nrow(cces_2020))
```

```
## # A tibble: 7 x 2
##   pres_vote                          prop
##   <fct>                             <dbl>
## 1 Joe Biden (Democrat)           0.508
## 2 Donald J. Trump (Republican) 0.343
## 3 Other                          0.0283
## 4 I did not vote in this race  0.00194
## 5 I did not vote                 0.000252
## 6 Not sure                       0.00369
## 7 <NA>                           0.114
```

Doesn’t work if you have filtered the data in any way during the pipe

Multiple grouping variables

What happens with multiple grouping variables

```r
vote_by_party <- cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  mutate(pres_vote = if_else(pres_vote == "Joe Biden (Democrat)",
                             "Biden", "Trump")) |>
  group_by(pid3, pres_vote) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n)) |>
  select(-n)

vote_by_party
```

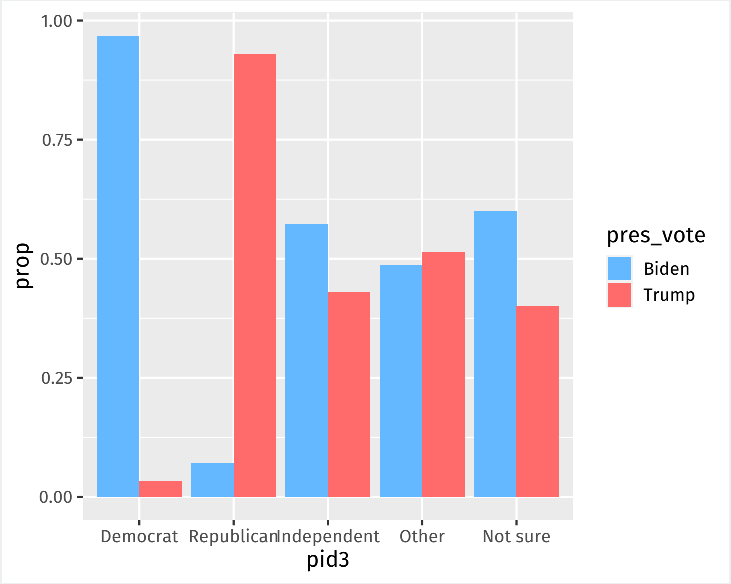
```
## # A tibble: 10 x 3
## # Groups:   pid3 [5]
##    pid3        pres_vote   prop
##    <fct>       <chr>      <dbl>
##  1 Democrat    Biden     0.968 
##  2 Democrat    Trump     0.0319
##  3 Republican  Biden     0.0712
##  4 Republican  Trump     0.929 
##  5 Independent Biden     0.571 
##  6 Independent Trump     0.429 
##  7 Other       Biden     0.487 
##  8 Other       Trump     0.513 
##  9 Not sure    Biden     0.599 
## 10 Not sure    Trump     0.401 
```

With multiple grouping variables, `summarize()` drops the last one.

Visualizing the cross-tab

We can visualize this using the `fill` aesthetic and `position="dodge"`:

```
ggplot(vote_by_party,
       aes(x = pid3, y = prop, fill = pres_vote)) +
  geom_col(position = "dodge") +
  scale_fill_manual(values = c(Biden = "steelblue1", Trump = "indianred1"))
```


1.00
0.75
0.50
0.25
0.00
prop
Democrat
Republican
Independent
Other
Not sure
pid3
pres_vote
Biden
Trump

Pivoting to create cross-tab

```r
cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  mutate(pres_vote = if_else(pres_vote == "Joe Biden (Democrat)",
                             "Biden", "Trump")) |>
  group_by(pid3, pres_vote) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n)) |>
  select(-n) |>
  pivot_wider(
    names_from = pid3,
    values_from = prop
  )
```

```
## # A tibble: 2 x 6
##   pres_vote Democrat Republican Independent Other `Not sure`
##   <chr>        <dbl>      <dbl>       <dbl> <dbl>      <dbl>
## 1 Biden       0.968      0.0712       0.571 0.487      0.599
## 2 Trump       0.0319     0.929        0.429 0.513      0.401
```

What if we want row proportions?

Switch the grouping variables to switch denominator:

```
cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  mutate(pres_vote = if_else(pres_vote == "Joe Biden (Democrat)",
                             "Biden", "Trump")) |>
  group_by(pres_vote, pid3) |>
  summarize(n = n()) |>
  mutate(prop = n / sum(n)) |>
  select(-n) |>
  pivot_wider(
    names_from = pid3,
    values_from = prop
  )
```

```
## # A tibble: 2 x 6
## # Groups:   pres_vote [2]
##   pres_vote Democrat Republican Independent  Other Not sur~1
##   <chr>        <dbl>      <dbl>       <dbl>  <dbl>     <dbl>
## 1 Biden       0.674      0.0327       0.252 0.0281    0.0133
## 2 Trump       0.0328     0.631        0.280 0.0437    0.0131
## # ... with abbreviated variable name 1: `Not sure`
```

Proportion of all observations

If we want the proportion of all rows, drop all groups

```r
cces_2020 |>
  filter(pres_vote %in% c("Joe Biden (Democrat)",
                          "Donald J. Trump (Republican)")) |>
  mutate(pres_vote = if_else(pres_vote == "Joe Biden (Democrat)",
                             "Biden", "Trump")) |>
  group_by(pid3, pres_vote) |>
  summarize(n = n(), .groups = "drop") |>
  mutate(prop = n / sum(n)) |>
  select(-n) |>
  pivot_wider(
    names_from = pid3,
    values_from = prop
  )
```

```
## # A tibble: 2 x 6
##   pres_vote Democrat Republican Independent  Other Not sur~1
##   <chr>        <dbl>      <dbl>       <dbl>  <dbl>     <dbl>
## 1 Biden       0.402      0.0195       0.150 0.0167   0.00791
## 2 Trump       0.0132     0.254        0.113 0.0176   0.00529
## # ... with abbreviated variable name 1: `Not sure`
```

2/ Measurement

Where does data come from?

- Social science is about developing and testing **causal theories**:
  - Does minimum wage change levels of employment?
  - Does outgroup contact influence views on immigration?
- Theories are made up of **concepts**:
  - Minimum wage, level of employment, outgroup contact, views on immigration.
  - We took these for granted when talking about causality.
- Need **operational definition** to concretely measure these concepts

Concepts vary in how observable they are

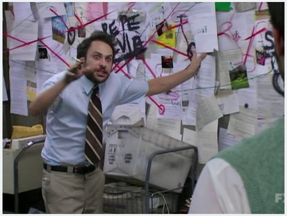
Kinds of measurement arranged by how direct we can measure them:

Observable in the world

- Minimum wage laws
- Sensor measurements
- Election results

Observable by survey

- Age of a person
- Employment status
- Presidential approval



Not directly observable

- A person's ideology
- Levels of democracy
- Extent of gerrymandering

Example

- Concept: presidential approval.
- Conceptual definition:
    - Extent to which US adults support the actions and policies of the current US president.
- Operational definition:
    - “On a scale from 1 to 5, where 1 is least supportive and 5 is more supportive, how much would you say you support the job that Joe Biden is doing as president?”

Measurement error

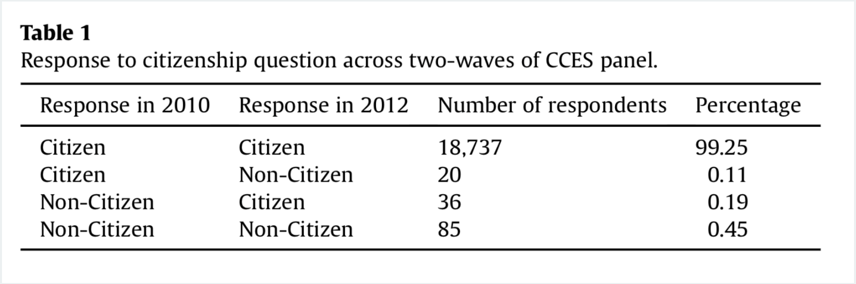
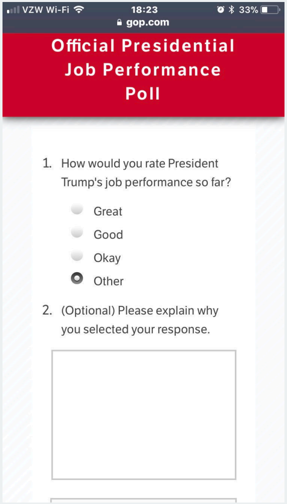


Table 1
Response to citizenship question across two-waves of CCES panel.

Response in 2010	Response in 2012	Number of respondents	Percentage
Citizen	Citizen	18,737	99.25
Citizen	Non-Citizen	20	0.11
Non-Citizen	Citizen	36	0.19
Non-Citizen	Non-Citizen	85	0.45

- **Measurement error**: chance variation in our measurements.
    - individual measurement = exact value + chance error
    - chance errors tend to cancel out when we take averages.
    - why? often data entry errors or faulty memories.

Bias

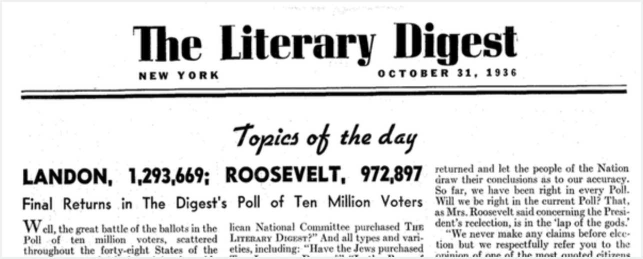


Official Presidential Job Performance Poll

1. How would you rate President Trump's job performance so far?
   - Great
   - Good
   - Okay
   - [✓ Other]
2. (Optional) Please explain why you selected your response.

- **Bias**: systematic errors for all units in the same direction.
- individual measurement = exact value + bias + chance error.
- "What did you eat yesterday?" $\rightsquigarrow$ underreporting

1936 Literary Digest Poll



The Literary Digest

NEW YORK — OCTOBER 31, 1936

Topics of the day

LANDON, 1,293,669; ROOSEVELT, 972,897

Final Returns in The Digest's Poll of Ten Million Voters

Well, the great battle of the ballots in the Poll of ten million voters, scattered throughout the forty-eight States of the

lican National Committee purchased The Literary Digest?" And all types and varieties, including: "Have the Jews purchased

returned and let the people of the Nation draw their conclusions as to our accuracy. So far, we have been right in every Poll. Will we be right in the current Poll? That, as Mrs. Roosevelt said concerning the President's reelection, is in the 'lap of the gods.'

"We never make any claims before election but we respectfully refer you to the opinion of one of the most quoted citizens

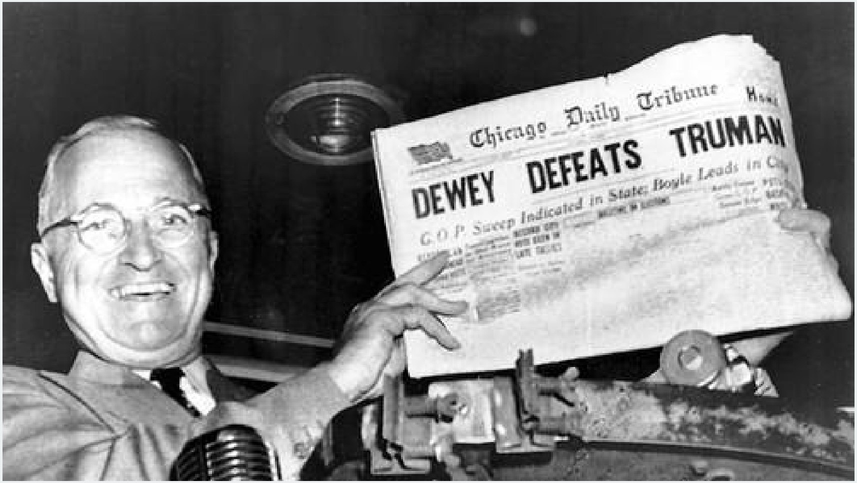
- Literary Digest predicted elections using mail-in polls.
- Source of addresses: automobile registrations, phone books, etc.
- In 1936, sent out 10 million ballots, over 2.3 million returned.
- George Gallup used only 50,000 respondents.

Poll fail

	FDR's Vote Share
Literary Digest	43%
George Gallup	56%
Actual Outcome	62%

- **Selection bias**: ballots skewed toward the wealthy (with cars, phones)
    - Only 1 in 4 households had a phone in 1936.
- **Nonresponse bias**: respondents differ from nonrespondents.
- $\rightsquigarrow$ when selection procedure is biased, adding more units won't help!

1948 Election



The Polling Disaster

	Truman	Dewey	Thurmond	Wallace
Crossley	45	50	2	3
Gallup	44	50	2	4
Roper	38	53	5	4
Actual	50	45	3	2

- **Quota sampling**: fixed quota of certain respondents for each interviewer
    - If black women make up 5% of the population, stop interviewing them once they make up 5% of your sample.
- Sample resembles the population on these characteristics
- Potential unobserved confounding $\rightsquigarrow$ **selection bias**
- Republicans easier to find within quotas (phones, listed addresses)

Sample surveys

- **Probability sampling** to ensure representativeness
    - Definition: every unit in the population has a known, non-zero probability of being selected into sample.
- **Simple random sampling**: every unit has an **equal** selection probability.
- Random digit dialing:
    - Take a particular area code + exchange: 617-495-XXXX.
    - Randomly choose each digit in XXXX to call a particular phone.
    - Every phone in America has an equal chance of being included in sample.

Sampling lingo

- **Target population**: set of people we want to learn about.
  - Ex: people who will vote in the next election.
- **Sampling frame**: list of people from which we will actually sample.
  - Frame bias: list of registered voters (frame) might include nonvoters!
- **Sample**: set of people contacted.
- **Respondents**: subset of sample that actually responds to the survey.
  - Unit non-response: sample $\neq$ respondents.
  - Not everyone picks up their phone.
- **Completed items**: subset of questions that respondents answer.
  - Item non-response: refusing to disclose their vote preference.

Difficulties of sampling

- Problems of telephone survey
  - Cell phones (double counting for the wealthy)
  - Caller ID screening (unit non-response)
  - Response rates down to 9%!
- An alternative: Internet surveys
  - Opt-in panels, respondent-driven sampling $\rightsquigarrow$ **non-probability sampling**
  - Cheaper, but non-representative
  - Digital divide: rich vs. poor, young vs. old
  - Correct for potential sampling bias via statistical methods.

Gov 50: 10. Summarizing Bivariate Relationships

Matthew Blackwell

Harvard University

Roadmap

1. Z-scores and standardization
2. Correlation
3. Writing our own functions

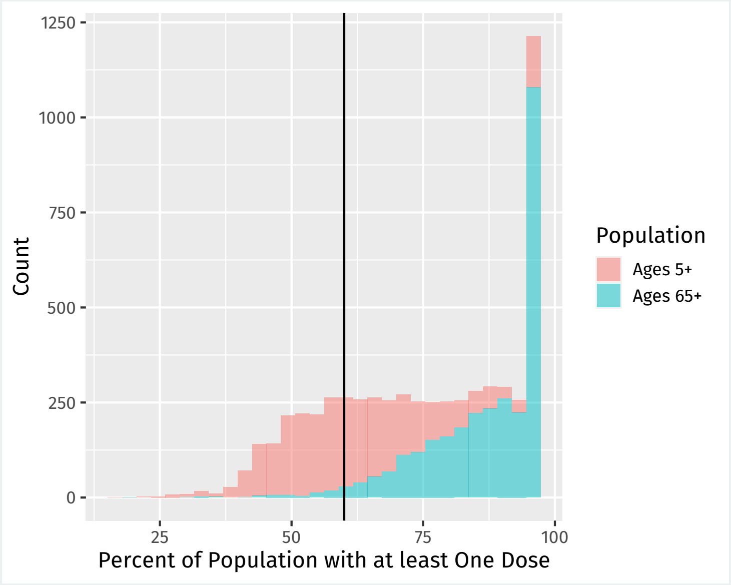
1/ Z-scores and standardization

COVID vaccination rates and votes

```r
library(tidyverse)
library(gov50data)
covid_votes
```

```
## # A tibble: 3,114 x 8
##    fips  county         state one_d~1 one_d~2 boost~3 dem_p~4
##    <chr> <chr>          <chr>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 26039 Crawford Cou~ MI       55.7    77.3   31.2      43.8
##  2 40015 Caddo County  OK       83.3    95     30.3      46.4
##  3 17007 Boone County  IL       71.1    94.5   35.1      41.8
##  4 12055 Highlands Co~ FL       68.9    93.7   24.7      40.3
##  5 34029 Ocean County  NJ       71      95     32.1      47.2
##  6 01067 Henry County  AL       58.5    85.5   18.2      40.1
##  7 27037 Dakota County MN       81      95     49.5      46.9
##  8 27115 Pine County   MN       56.5    85     31.7      47.0
##  9 51750 Radford city  VA       41.5    73.8    1.79     46.4
## 10 22009 Avoyelles Pa~ LA       59.7    80.1   21.9      45.7
## # ... with 3,104 more rows, 1 more variable:
## #   dem_pct_2020 <dbl>, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct,
## #   3: booster_5plus_pct, 4: dem_pct_2000
```

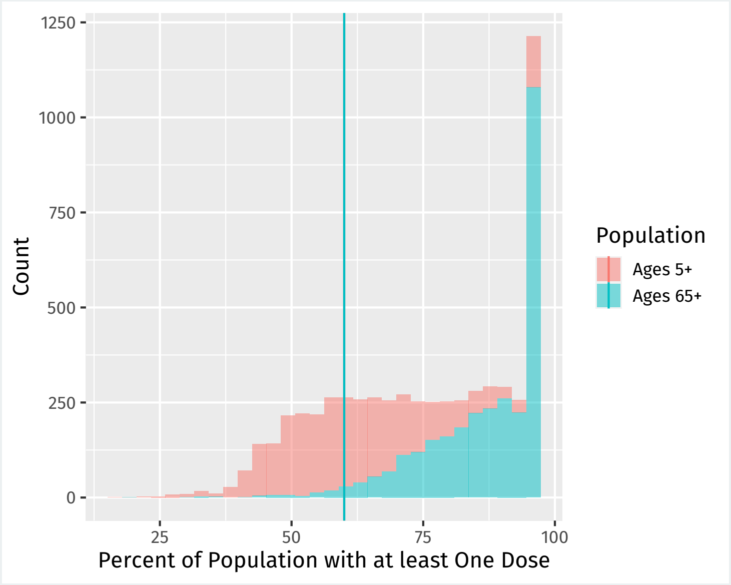
Is 60% vaccinated a lot?



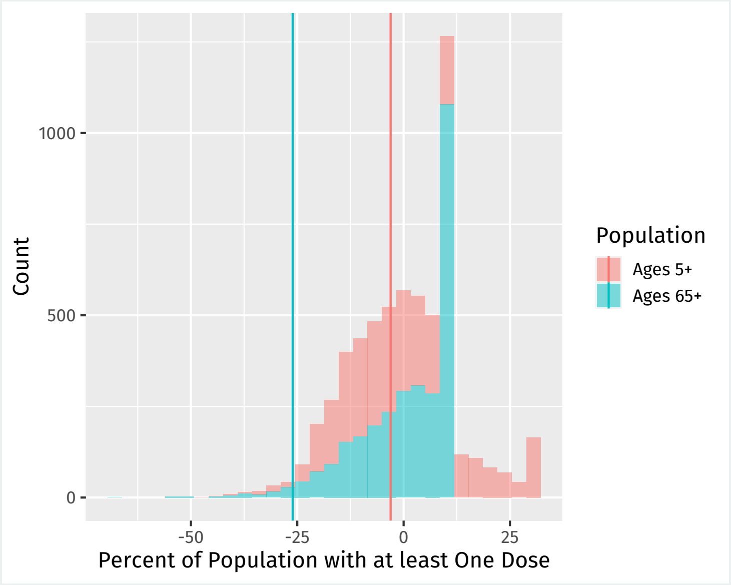
How large is large?

- How large 60% vaccinated is depends on the distribution!
    - Clear to see from the histogram
    - Middling for the 5+ group, but very low for the 65+ group.
- Can we transform the values of our variables to be **common units**?
- Yes, with two transformations:
    - **Centering**: subtract the mean of the variable from each value.
    - **Scaling**: dividing deviations from the mean by the standard deviation.

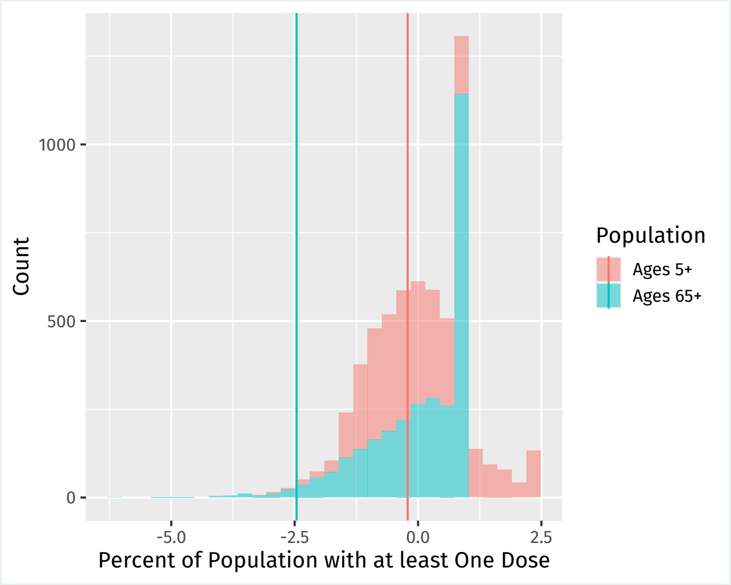
Original distributions



Centered distributions



Centered and scaled distributions



Z-scores

- Centering tells us immediately if a value is above or below the mean.
- Scaling tells us how many standard deviations away from the mean it is.
- Combine them with the **z-score** transformation:

$$\text{z-score of } x_i = \frac{x_i - \text{mean of } x}{\text{standard deviation of } x}$$

- Useful heuristic: data more than 3 SDs away from mean are rare.

z-score example

```r
covid_votes |>
  mutate(one_dose_centered = one_dose_5plus_pct -
           mean(one_dose_5plus_pct, na.rm = TRUE)) |>
  select(fips:state, one_dose_5plus_pct, one_dose_centered)
```

```
## # A tibble: 3,114 x 5
##    fips  county           state one_dose_5plus_pct one_dos~1
##    <chr> <chr>            <chr>              <dbl>     <dbl>
##  1 26039 Crawford County  MI                  55.7     -7.35
##  2 40015 Caddo County     OK                  83.3     20.2 
##  3 17007 Boone County     IL                  71.1      8.05
##  4 12055 Highlands County FL                  68.9      5.85
##  5 34029 Ocean County     NJ                  71        7.95
##  6 01067 Henry County     AL                  58.5     -4.55
##  7 27037 Dakota County    MN                  81       17.9 
##  8 27115 Pine County      MN                  56.5     -6.55
##  9 51750 Radford city     VA                  41.5    -21.6 
## 10 22009 Avoyelles Parish LA                  59.7     -3.35
## # ... with 3,104 more rows, and abbreviated variable name
## #   1: one_dose_centered
```

z-score example

```r
covid_votes |>
  mutate(
    one_dose_z =
      (one_dose_5plus_pct - mean(one_dose_5plus_pct, na.rm = TRUE)) /
      sd(one_dose_5plus_pct, na.rm = TRUE))   |>
  select(fips:state, one_dose_5plus_pct, one_dose_z)
```

```
## # A tibble: 3,114 x 5
##    fips  county           state one_dose_5plus_pct one_dos~1
##    <chr> <chr>            <chr>              <dbl>     <dbl>
##  1 26039 Crawford County  MI                  55.7    -0.508
##  2 40015 Caddo County     OK                  83.3     1.40
##  3 17007 Boone County     IL                  71.1     0.556
##  4 12055 Highlands County FL                  68.9     0.404
##  5 34029 Ocean County     NJ                  71       0.549
##  6 01067 Henry County     AL                  58.5    -0.314
##  7 27037 Dakota County    MN                  81       1.24
##  8 27115 Pine County      MN                  56.5    -0.452
##  9 51750 Radford city     VA                  41.5    -1.49
## 10 22009 Avoyelles Parish LA                  59.7    -0.231
## # ... with 3,104 more rows, and abbreviated variable name
## #   1: one_dose_z
```

2/ Correlation

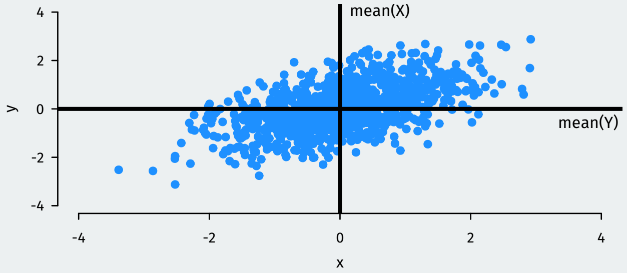
Correlation

- How do variables move together on average?
- When x_i is big, what is y_i likely to be?
  - Positive correlation: when x_i is big, y_i is also big
  - Negative correlation: when x_i is big, y_i is small
  - High magnitude of correlation: data cluster tightly around a line.
- The technical definition of the **correlation coefficient**:

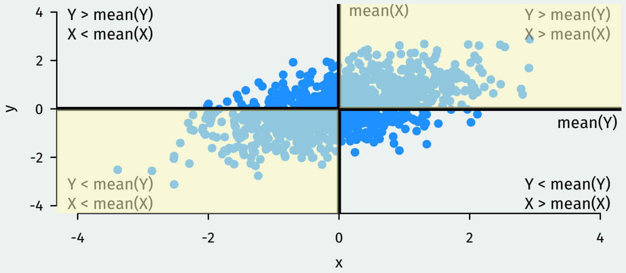
$$\frac{1}{n-1}\sum_{i=1}^{n}\left[(\text{z-score for } x_i) \times (\text{z-score for } y_i)\right]$$

- Interpretation:
  - Correlation is between -1 and 1
  - Correlation of 0 means no linear association.
  - Positive correlations $\rightsquigarrow$ positive associations.
  - Negative correlations $\rightsquigarrow$ negative associations.
  - Closer to -1 or 1 means stronger association.

Correlation intuition

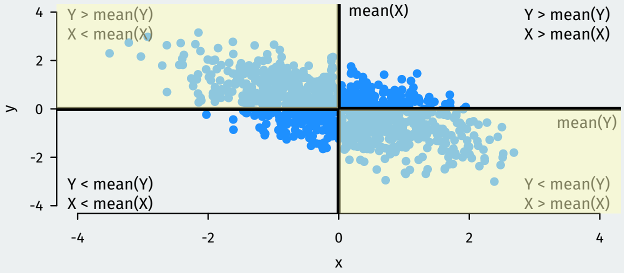


Correlation intuition



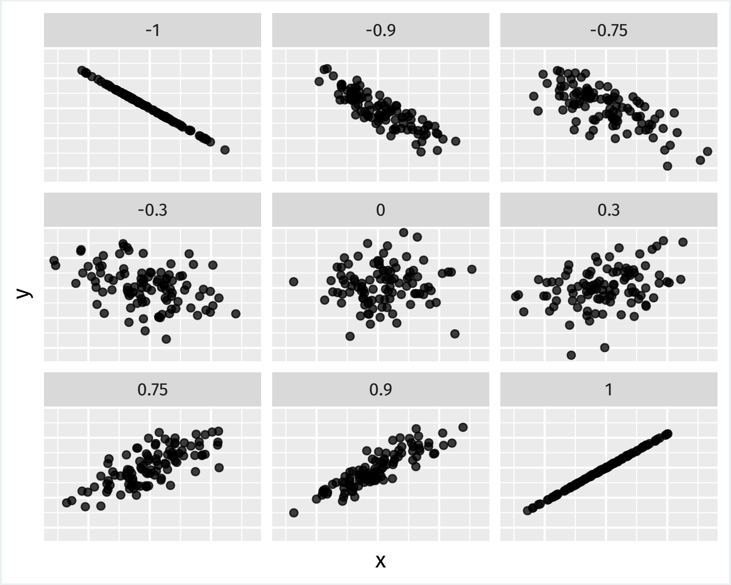
- Large values of X tend to occur with large values of Y:
    - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{pos. num.}) \times (\text{pos. num}) = +$
- Small values of X tend to occur with small values of Y:
    - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{neg. num.}) \times (\text{neg. num}) = +$
- If these dominate $\rightsquigarrow$ positive correlation.

Correlation intuition



- Large values of X tend to occur with small values of Y:
    - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{pos. num.}) \times (\text{neg. num}) = -$
- Small values of X tend to occur with large values of Y:
    - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{neg. num.}) \times (\text{pos. num}) = -$
- If these dominate $\rightsquigarrow$ negative correlation.

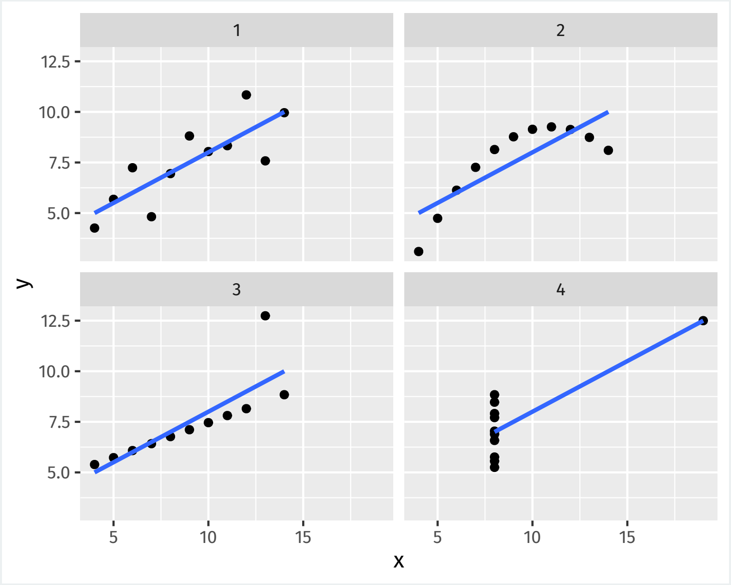
Correlation examples



Properties of correlation coefficient

- Correlation measures **linear** association.
- Order doesn't matter: `cor(x,y) = cor(y,x)`
- Not affected by changes of scale:
  - `cor(x,y) = cor(ax+b, cy+d)`
  - Celsius vs. Fahreneheit; dollars vs. pesos; cm vs. in.

All 4 relationships have 0.816 correlation



Correlation in R

Use the `cor()` function:

```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020)
```

```
## [1] NA
```

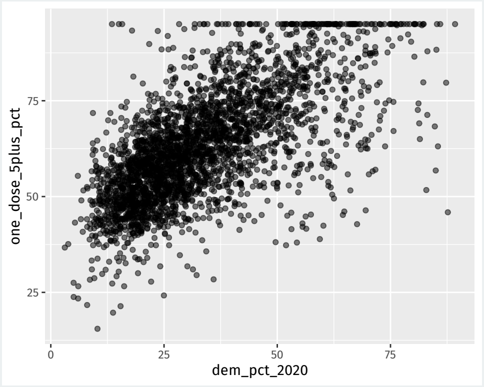
Missing values: set the `use = "pairwise"` → available case analysis

```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020,
    use = "pairwise")
```

```
## [1] 0.666
```

Comparing correlations

```r
covid_votes |>
  ggplot(aes(x = dem_pct_2020, y = one_dose_5plus_pct)) +
  geom_point(alpha = 0.5)
```

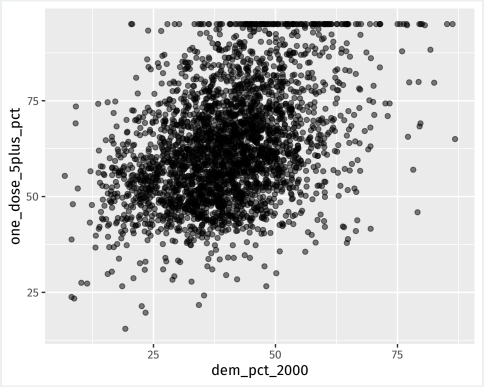


```r
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020,
    use = "pairwise")
```

```
## [1] 0.666
```

Comparing correlations

```r
covid_votes |>
  ggplot(aes(x = dem_pct_2000, y = one_dose_5plus_pct)) +
  geom_point(alpha = 0.5)
```

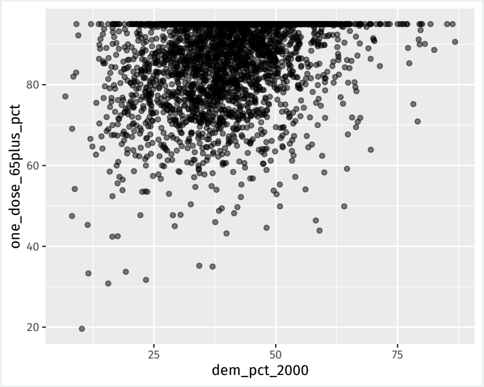


```r
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2000,
    use = "pairwise")
```

```
## [1] 0.394
```

Comparing correlations

```r
covid_votes |>
  ggplot(aes(x = dem_pct_2000, y = one_dose_65plus_pct)) +
  geom_point(alpha = 0.5)
```



```r
cor(covid_votes$one_dose_65plus_pct, covid_votes$dem_pct_2000,
    use = "pairwise")
```

```
## [1] 0.263
```

3/ Writing our own functions

Why write functions?

Copy-pasting code tedious and prone to failure:

```r
covid_votes |>
  mutate(
    one_dose_5p_z =
      (one_dose_5plus_pct - mean(one_dose_5plus_pct, na.rm = TRUE)) /
      sd(one_dose_5plus_pct, na.rm = TRUE),
    one_dose_65p_z =
      (one_dose_65plus_pct - mean(one_dose_65plus_pct, na.rm = TRUE)) /
      sd(one_dose_65plus_pct, na.rm = TRUE),
    booster_z =
      (booster_5plus_pct - mean(booster_5plus_pct, na.rm = TRUE)) /
      sd(booster_5plus_pct, na.rm = TRUE),
    dem_pct_2000_z =
      (dem_pct_2000 - mean(dem_pct_2000, na.rm = TRUE)) /
      sd(dem_pct_2000, na.rm = TRUE),
    dem_pct_2020_z =
      (dem_pct_2020 - mean(dem_pct_2020, na.rm = TRUE)) /
      sd(dem_pct_2020, na.rm = TRUE)
  )
```

Writing a new function

Notice that all of the mutations follow the same template:

```
(█ - mean(█, na.rm = TRUE)) / sd(█, na.rm = TRUE)
```

Only one thing varies: the column of data, represented with █

Components of a function

We create functions like so:

```
name <- function(arguments) {
  body
}
```

Three components:

1. **Name**: the name of the function that we'll use to call it. Maybe `z_score`?
2. **Arguments**: things that we want to vary across calls of our function. We'll use x.
3. **Body**: the code that the function performs.

Our first function

Convert our template to a function:

```r
z_score <- function(x) {
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
}
```

Check that it seems to work:

```r
z_score(c(1,2, 3, 4, 5))
```

```
## [1] -1.265 -0.632  0.000  0.632  1.265
```

Cleaning up our code

```r
covid_votes |>
  mutate(
    one_dose_5p_z = z_score(one_dose_5plus_pct),
    one_dose_65p_z = z_score(one_dose_65plus_pct),
    booster_z = z_score(booster_5plus_pct),
    dem_pct_2000_z = z_score(dem_pct_2000),
    dem_pct_2020_z = z_score(dem_pct_2020)
  )
```

`across()` function

If we want to replace our variables with z-scores, we can use the `across()` function to perform many mutations at once:

```
covid_votes |>
  mutate(across(one_dose_5plus_pct:dem_pct_2020, z_score))
```

```
## # A tibble: 3,114 x 8
##    fips  county         state one_d~1 one_d~2 boost~3 dem_p~4
##    <chr> <chr>          <chr>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 26039 Crawford Cou~ MI     -0.508 -0.829    0.531  0.340
##  2 40015 Caddo County  OK      1.40   0.843    0.439  0.556
##  3 17007 Boone County  IL      0.556  0.795    0.927  0.163
##  4 12055 Highlands Co~ FL      0.404  0.720   -0.135  0.0402
##  5 34029 Ocean County  NJ      0.549  0.843    0.623  0.624
##  6 01067 Henry County  AL     -0.314 -0.0545  -0.799  0.0255
##  7 27037 Dakota County MN      1.24   0.843    2.40   0.598
##  8 27115 Pine County   MN     -0.452 -0.102    0.577  0.612
##  9 51750 Radford city  VA     -1.49  -1.16    -2.47   0.556
## 10 22009 Avoyelles Pa~ LA     -0.231 -0.564   -0.424  0.501
## # ... with 3,104 more rows, 1 more variable:
## #   dem_pct_2020 <dbl>, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct,
```

Alternative approach

We could also target all the numeric variables:

```
covid_votes |>
  mutate(across(where(is.numeric), z_score))
```

```
## # A tibble: 3,114 x 8
##    fips  county         state one_d~1 one_d~2 boost~3 dem_p~4
##    <chr> <chr>          <chr>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 26039 Crawford Cou~ MI     -0.508 -0.829    0.531  0.340
##  2 40015 Caddo County  OK      1.40   0.843    0.439  0.556
##  3 17007 Boone County  IL      0.556  0.795    0.927  0.163
##  4 12055 Highlands Co~ FL      0.404  0.720   -0.135  0.0402
##  5 34029 Ocean County  NJ      0.549  0.843    0.623  0.624
##  6 01067 Henry County  AL     -0.314 -0.0545  -0.799  0.0255
##  7 27037 Dakota County MN      1.24   0.843    2.40   0.598
##  8 27115 Pine County   MN     -0.452 -0.102    0.577  0.612
##  9 51750 Radford city  VA     -1.49  -1.16    -2.47   0.556
## 10 22009 Avoyelles Pa~ LA     -0.231 -0.564   -0.424  0.501
## # ... with 3,104 more rows, 1 more variable:
## #   dem_pct_2020 <dbl>, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct,
## #   3: booster_5plus_pct, 4: dem_pct_2000
```

Alternative approach

We could also target only the first dose variables:

```
covid_votes |>
  mutate(across(starts_with("one_dose"), z_score))
```

```
## # A tibble: 3,114 x 8
##    fips  county         state one_d~1 one_d~2 boost~3 dem_p~4
##    <chr> <chr>          <chr>   <dbl>   <dbl>   <dbl>   <dbl>
##  1 26039 Crawford Cou~ MI      -0.508 -0.829     31.2    43.8
##  2 40015 Caddo County  OK       1.40   0.843     30.3    46.4
##  3 17007 Boone County  IL       0.556  0.795     35.1    41.8
##  4 12055 Highlands Co~ FL       0.404  0.720     24.7    40.3
##  5 34029 Ocean County  NJ       0.549  0.843     32.1    47.2
##  6 01067 Henry County  AL      -0.314 -0.0545    18.2    40.1
##  7 27037 Dakota County MN       1.24   0.843     49.5    46.9
##  8 27115 Pine County   MN      -0.452 -0.102     31.7    47.0
##  9 51750 Radford city  VA      -1.49  -1.16       1.79   46.4
## 10 22009 Avoyelles Pa~ LA      -0.231 -0.564     21.9    45.7
## # ... with 3,104 more rows, 1 more variable:
## #   dem_pct_2020 <dbl>, and abbreviated variable names
## #   1: one_dose_5plus_pct, 2: one_dose_65plus_pct,
## #   3: booster_5plus_pct, 4: dem_pct_2000
```

Adding arguments to our function

What if we want to be able to control `na.rm` in the calls to `mean()` and `sd()` in our `z_score` function? Add an argument!

```r
z_score2 <- function(x, na.rm = FALSE) {
  (x - mean(x, na.rm = na.rm)) / sd(x, na.rm = na.rm)
}
```

```r
head(z_score2(covid_votes$one_dose_5plus_pct))
```

```
## [1] NA NA NA NA NA NA
```

```r
head(z_score2(covid_votes$one_dose_5plus_pct, na.rm = TRUE))
```

```
## [1] -0.508  1.398  0.556  0.404  0.549 -0.314
```

Gov 50: 11. Tidying and Joining Data

Matthew Blackwell

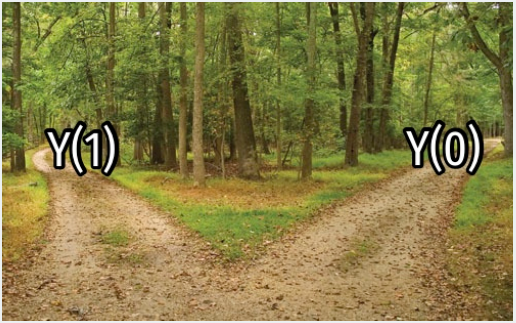
Harvard University

Roadmap

1. Causality review
2. Pivoting data longer
3. Joining data sets

1/ Causality review

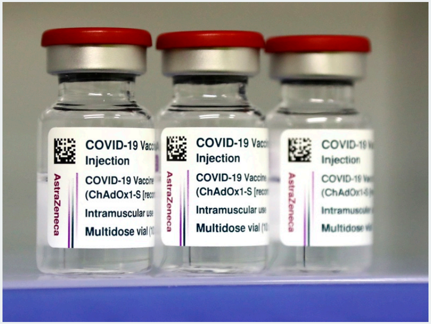
Potential outcomes



Potential outcomes:

- $Y_i(1)$ is the value that the outcome would take if gave unit i **treatment** and changed nothing else about them.
- $Y_i(0)$ is the value that the outcome would take if gave unit i **no treatment** and changed nothing else about them.
- Not the **possible values** of the outcome

COVID-19 vaccine trials



Treatment: $T_i = 1$ if vaccinated, $T_i = 0$ if not

Outcome: $Y_i = 1$ if acquired COVID after 12 weeks, $Y_i = 0$ if not

1. What are the potential outcomes $Y_i(1)$ and $Y_i(0)$?
2. Why not compare early volunteers for the vaccine to the overall population?

2/ Pivoting data longer

Mortality data

```r
library(tidyverse)
library(gov50data)
mortality
```

```
## # A tibble: 217 x 52
##    country        count~1 indic~2 `1972` `1973` `1974` `1975`
##    <chr>          <chr>   <chr>    <dbl>  <dbl>  <dbl>  <dbl>
##  1 Aruba          ABW     Mortal~   NA     NA     NA     NA
##  2 Afghanistan    AFG     Mortal~  291    285.   280.   274.
##  3 Angola         AGO     Mortal~   NA     NA     NA     NA
##  4 Albania        ALB     Mortal~   NA     NA     NA     NA
##  5 Andorra        AND     Mortal~   NA     NA     NA     NA
##  6 United Arab ~  ARE     Mortal~   80.1   72.6   65.7   59.4
##  7 Argentina      ARG     Mortal~   69.7   68.2   66.1   63.3
##  8 Armenia        ARM     Mortal~   NA     NA     NA     NA
##  9 American Sam~  ASM     Mortal~   NA     NA     NA     NA
## 10 Antigua and ~  ATG     Mortal~   26.9   25.1   23.5   22.1
## # ... with 207 more rows, 45 more variables: `1976` <dbl>,
## #   `1977` <dbl>, `1978` <dbl>, `1979` <dbl>, `1980` <dbl>,
## #   `1981` <dbl>, `1982` <dbl>, `1983` <dbl>, `1984` <dbl>,
## #   `1985` <dbl>, `1986` <dbl>, `1987` <dbl>, `1988` <dbl>,
## #   `1989` <dbl>, `1990` <dbl>, `1991` <dbl>, `1992` <dbl>,
```

Pivoting longer

Mortality data in a “wide” format (years in columns).

We can convert this to country-year rows with `pivot_longer()`.

```
mydata |>
  pivot_longer(
    cols = <<variables to pivot>>,
    names_to = <<new variable to put column names>>,
    values_to = <<new variable to put column values>>
  )
```

Pivoting the mortality data

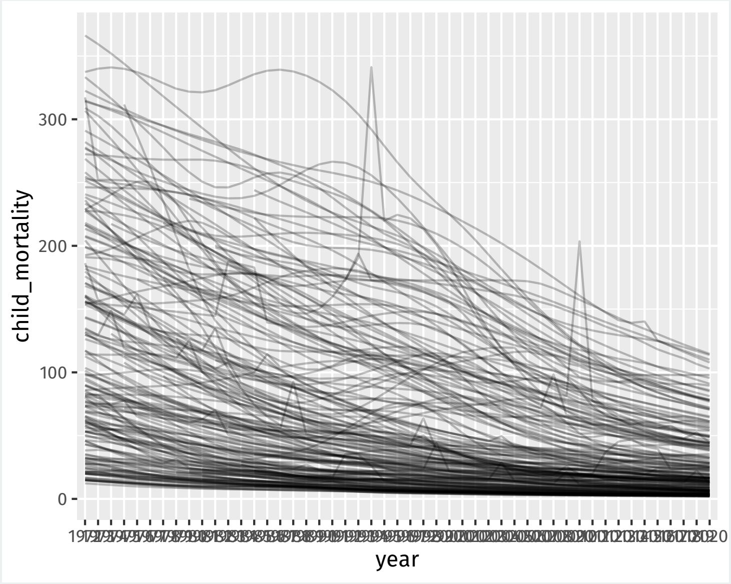
```r
mortality |>
  select(-indicator) |>
  pivot_longer(
    cols = `1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  )
```

```
## # A tibble: 10,633 x 4
##    country country_code year  child_mortality
##    <chr>   <chr>        <chr>           <dbl>
##  1 Aruba   ABW          1972               NA
##  2 Aruba   ABW          1973               NA
##  3 Aruba   ABW          1974               NA
##  4 Aruba   ABW          1975               NA
##  5 Aruba   ABW          1976               NA
##  6 Aruba   ABW          1977               NA
##  7 Aruba   ABW          1978               NA
##  8 Aruba   ABW          1979               NA
##  9 Aruba   ABW          1980               NA
## 10 Aruba   ABW          1981               NA
## # ... with 10,623 more rows
```

Let’s do line plots!

```r
mortality |>
  select(-indicator) |>
  pivot_longer(
    cols = `1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  ) |>
  ggplot(mapping = aes(x = year, y = child_mortality, group = country)) +
  geom_line(alpha = 0.25)
```

Hmm, what's going on?



Making sure year is numeric

By default, pivoted column names are characters, but we can transform them:

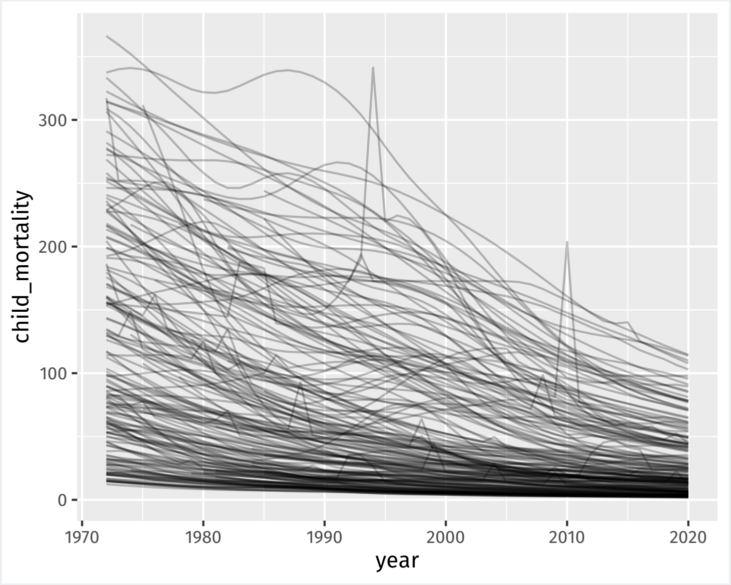
```
mortality_long <- mortality |>
  select(-indicator) |>
  pivot_longer(
    cols = `1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  ) |>
  mutate(year = as.integer(year))
mortality_long
```

```
## # A tibble: 10,633 x 4
##    country country_code  year child_mortality
##    <chr>   <chr>        <int>           <dbl>
##  1 Aruba   ABW           1972              NA
##  2 Aruba   ABW           1973              NA
##  3 Aruba   ABW           1974              NA
##  4 Aruba   ABW           1975              NA
##  5 Aruba   ABW           1976              NA
##  6 Aruba   ABW           1977              NA
```

Let’s (re)do line plots!

```r
mortality_long |>
  ggplot(mapping = aes(x = year, y = child_mortality, group = country)) +
  geom_line(alpha = 0.25)
```

There we go


child_mortality
300
200
100
0
1970
1980
1990
2000
2010
2020
year

Spotify data

```
spotify
```

```
## # A tibble: 490 x 54
##    Track ~1 Artist week1 week2 week3 week4 week5 week6 week7
##    <chr>    <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 The Box  Roddy~     1     1     1     1     1     1     1
##  2 ROXANNE  Arizo~     2     4     5     4     4     4     6
##  3 Yummy    Justi~     3     6    17    17    17    24    15
##  4 Circles  Post ~     4     7     9    10     7    10    11
##  5 BOP      DaBaby     5     5     7     5    11    12    18
##  6 Falling  Trevo~     6     8    10     7     6     8    10
##  7 Dance M~ Tones~     7    13    13    12    12    13    17
##  8 Bandit ~ Juice~     8    11    14    14    15    20    27
##  9 Futsal ~ Lil U~     9     9    19    21    24    32    40
## 10 everyth~ Billi~    10    17    28     9     8    11    14
## # ... with 480 more rows, 45 more variables: week8 <dbl>,
## #   week9 <dbl>, week10 <dbl>, week11 <dbl>, week12 <dbl>,
## #   week13 <dbl>, week14 <dbl>, week15 <dbl>, week16 <dbl>,
## #   week17 <dbl>, week18 <dbl>, week19 <dbl>, week20 <dbl>,
## #   week21 <dbl>, week22 <dbl>, week23 <dbl>, week24 <dbl>,
## #   week25 <dbl>, week26 <dbl>, week27 <dbl>, week28 <dbl>,
## #   week29 <dbl>, week30 <dbl>, week31 <dbl>, ...
```

Pivoting not ideal

Last approach isn’t ideal because of the week prefix:

```
spotify |>
  pivot_longer(
    cols = c(-`Track Name`, -Artist),
    names_to = ”week_of_year”,
    values_to = ”rank”
  )
```

```
## # A tibble: 25,480 x 4
##    `Track Name` Artist      week_of_year  rank
##    <chr>        <chr>       <chr>        <dbl>
##  1 The Box      Roddy Ricch week1            1
##  2 The Box      Roddy Ricch week2            1
##  3 The Box      Roddy Ricch week3            1
##  4 The Box      Roddy Ricch week4            1
##  5 The Box      Roddy Ricch week5            1
##  6 The Box      Roddy Ricch week6            1
##  7 The Box      Roddy Ricch week7            1
##  8 The Box      Roddy Ricch week8            1
##  9 The Box      Roddy Ricch week9            1
## 10 The Box      Roddy Ricch week10           1
```

Removing a column name prefix

When the data in the column name has a fixed prefix, we can use the `names_prefix` to remove it when moving the data to rows

```
spotify |>
  pivot_longer(
    cols = c(-`Track Name`, -Artist),
    names_to = "week_of_year",
    values_to = "rank",
    names_prefix = "week"
  ) |>
  mutate(
    week_of_year = as.integer(week_of_year)
  )
```

Removing a column name prefix

```
## # A tibble: 25,480 x 4
##    `Track Name` Artist      week_of_year  rank
##    <chr>        <chr>              <int> <dbl>
##  1 The Box      Roddy Ricch            1     1
##  2 The Box      Roddy Ricch            2     1
##  3 The Box      Roddy Ricch            3     1
##  4 The Box      Roddy Ricch            4     1
##  5 The Box      Roddy Ricch            5     1
##  6 The Box      Roddy Ricch            6     1
##  7 The Box      Roddy Ricch            7     1
##  8 The Box      Roddy Ricch            8     1
##  9 The Box      Roddy Ricch            9     1
## 10 The Box      Roddy Ricch           10     1
## # ... with 25,470 more rows
```

3/ Joining data sets

Gapminder data

```r
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

Joining data sets

What if we want to add the `child_mortality` variable to the `gampinder` data?

Just add the columns? Rows are not aligned properly!

```
gapminder |>
  select(country, year) |>
  head()
```

```
## # A tibble: 6 x 2
##   country      year
##   <fct>       <int>
## 1 Afghanistan  1952
## 2 Afghanistan  1957
## 3 Afghanistan  1962
## 4 Afghanistan  1967
## 5 Afghanistan  1972
## 6 Afghanistan  1977
```

```
mortality_long |>
  select(country, year) |>
  head()
```

```
## # A tibble: 6 x 2
##   country  year
##   <chr>   <int>
## 1 Aruba    1972
## 2 Aruba    1973
## 3 Aruba    1974
## 4 Aruba    1975
## 5 Aruba    1976
## 6 Aruba    1977
```

Key variables

A **primary key** is a variable or set of variables that uniquely identifies rows in the data.

- `{country, year}` in the `gapminder` data

A **foreign key** is the corresponding variable(s) in another table.

- `{country, year}` in the `mortality_long` data

If we align the two tables based on these variables, we can add variables from one to the other.

Checking that the keys are unique

Things get weird if these keys are not unique. Let's check.

Checking primary key is unique:

```
gapminder |>
  count(country, year) |>
  filter(n > 1)
```

```
## # A tibble: 0 x 3
```

Checking foreign key:

```
mortality_long |>
  count(country, year) |>
  filter(n > 1)
```

```
## # A tibble: 0 x 3
```

`left_join()`: add variables to primary table

`left_join()` keeps all rows from the first argument/piped data:

```
gapminder |>
  left_join(mortality_long) |>
  select(country, year, lifeExp, pop, gdpPercap, child_mortality) |>
  head(n = 6)
```

```
## Joining, by = c("country", "year")
```

```
## # A tibble: 6 x 6
##   country      year lifeExp      pop gdpPercap child_morta~1
##   <chr>       <int>   <dbl>    <int>     <dbl>         <dbl>
## 1 Afghanistan  1952    28.8  8425333      779.           NA
## 2 Afghanistan  1957    30.3  9240934      821.           NA
## 3 Afghanistan  1962    32.0 10267083      853.           NA
## 4 Afghanistan  1967    34.0 11537966      836.           NA
## 5 Afghanistan  1972    36.1 13079460      740.          291
## 6 Afghanistan  1977    38.4 14880372      786.          262.
## # ... with abbreviated variable name 1: child_mortality
```

Rows in primary table not in foreign table: new values are NA.

inner_join(): add and filter

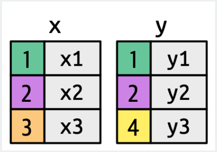
inner_join() adds the variables from the foreign table and filters to rows in both tables:

```
gapminder |>
  inner_join(mortality_long) |>
  select(country, year, lifeExp, pop, gdpPercap, child_mortality) |>
  head(n = 6)
```

```
## Joining, by = c("country", "year")

## # A tibble: 6 x 6
##   country      year lifeExp      pop gdpPercap child_morta~1
##   <chr>       <int>   <dbl>    <int>     <dbl>         <dbl>
## 1 Afghanistan  1972    36.1 13079460      740.           291
## 2 Afghanistan  1977    38.4 14880372      786.          262.
## 3 Afghanistan  1982    39.9 12881816      978.          231.
## 4 Afghanistan  1987    40.8 13867957      852.          198.
## 5 Afghanistan  1992    41.7 16317921      649.          166.
## 6 Afghanistan  1997    41.8 22227415      635.          142.
## # ... with abbreviated variable name 1: child_mortality
```

How inner joins work

Two data sets:



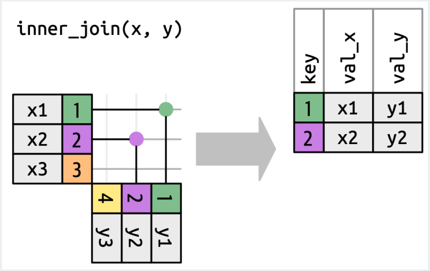
x

1	x1
2	x2
3	x3

y

1	y1
2	y2
4	y3

Find matching keys:



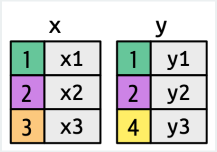
```
inner_join(x, y)
```

key	val_x	val_y
1	x1	y1
2	x2	y2

Credit: R for Data Science

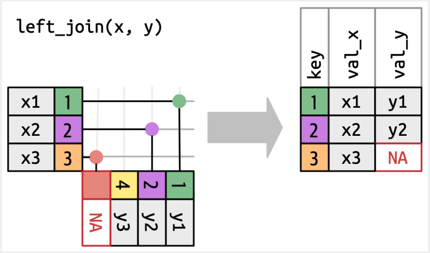
How left joins work

Two data sets:



x
1
x1
2
x2
3
x3
y
1
y1
2
y2
4
y3

Keep all x keys:



Credit: R for Data Science

More complicated example

```r
library(nycflights13)
flights2 <- flights |>
  select(year, time_hour, origin, dest, tailnum, carrier)
flights2
```

```
## # A tibble: 336,776 x 6
##     year time_hour           origin dest  tailnum carrier
##    <int> <dttm>              <chr>  <chr> <chr>   <chr>  
##  1  2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA     
##  2  2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA     
##  3  2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA     
##  4  2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6     
##  5  2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL     
##  6  2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA     
##  7  2013 2013-01-01 06:00:00 EWR    FLL   N516JB  B6     
##  8  2013 2013-01-01 06:00:00 LGA    IAD   N829AS  EV     
##  9  2013 2013-01-01 06:00:00 JFK    MCO   N593JB  B6     
## 10  2013 2013-01-01 06:00:00 LGA    ORD   N3ALAA  AA     
## # ... with 336,766 more rows
```

Planes data

```
planes2 <- planes |>
  select(tailnum, year, type, engine, seats)
planes2
```

```
## # A tibble: 3,322 x 5
##    tailnum  year type                    engine    seats
##    <chr>   <int> <chr>                   <chr>     <int>
##  1 N10156   2004 Fixed wing multi engine Turbo-fan    55
##  2 N102UW   1998 Fixed wing multi engine Turbo-fan   182
##  3 N103US   1999 Fixed wing multi engine Turbo-fan   182
##  4 N104UW   1999 Fixed wing multi engine Turbo-fan   182
##  5 N10575   2002 Fixed wing multi engine Turbo-fan    55
##  6 N105UW   1999 Fixed wing multi engine Turbo-fan   182
##  7 N107US   1999 Fixed wing multi engine Turbo-fan   182
##  8 N108UW   1999 Fixed wing multi engine Turbo-fan   182
##  9 N109UW   1999 Fixed wing multi engine Turbo-fan   182
## 10 N110UW   1999 Fixed wing multi engine Turbo-fan   182
## # ... with 3,312 more rows
```

year here is manufacture year.

What happens with naive join?

```r
flights2 |>
  left_join(planes2)
```

```
## Joining, by = c("year", "tailnum")

## # A tibble: 336,776 x 9
##     year time_hour           origin dest  tailnum carrier type  engine
##    <int> <dttm>              <chr>  <chr> <chr>   <chr>   <chr> <chr>
##  1  2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA      <NA>  <NA>
##  2  2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA      <NA>  <NA>
##  3  2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA      <NA>  <NA>
##  4  2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6      <NA>  <NA>
##  5  2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL      <NA>  <NA>
##  6  2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA      <NA>  <NA>
##  7  2013 2013-01-01 06:00:00 EWR    FLL   N516JB  B6      <NA>  <NA>
##  8  2013 2013-01-01 06:00:00 LGA    IAD   N829AS  EV      <NA>  <NA>
##  9  2013 2013-01-01 06:00:00 JFK    MCO   N593JB  B6      <NA>  <NA>
## 10  2013 2013-01-01 06:00:00 LGA    ORD   N3ALAA  AA      <NA>  <NA>
## # ... with 336,766 more rows, and 1 more variable: seats <int>
```

Specify the joining variables

```r
flights2 |>
  left_join(planes2, by = "tailnum")
```

```
## # A tibble: 336,776 x 10
##    year.x time_hour           origin dest  tailnum carrier year.y
##     <int> <dttm>              <chr>  <chr> <chr>   <chr>    <int>
##  1   2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA        1999
##  2   2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA        1998
##  3   2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA        1990
##  4   2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6        2012
##  5   2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL        1991
##  6   2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA        2012
##  7   2013 2013-01-01 06:00:00 EWR    FLL   N516JB  B6        2000
##  8   2013 2013-01-01 06:00:00 LGA    IAD   N829AS  EV        1998
##  9   2013 2013-01-01 06:00:00 JFK    MCO   N593JB  B6        2004
## 10   2013 2013-01-01 06:00:00 LGA    ORD   N3ALAA  AA          NA
## # ... with 336,766 more rows, and 3 more variables: type <chr>,
## #   engine <chr>, seats <int>
```

Change variables names

```r
flights2 |>
  left_join(planes2 |> rename(manufacture_year = year))
```

```
## Joining, by = "tailnum"
```

```
## # A tibble: 336,776 x 10
##     year time_hour           origin dest  tailnum carrier manufactur~1
##    <int> <dttm>              <chr>  <chr> <chr>   <chr>          <int>
##  1  2013 2013-01-01 05:00:00 EWR    IAH   N14228  UA              1999
##  2  2013 2013-01-01 05:00:00 LGA    IAH   N24211  UA              1998
##  3  2013 2013-01-01 05:00:00 JFK    MIA   N619AA  AA              1990
##  4  2013 2013-01-01 05:00:00 JFK    BQN   N804JB  B6              2012
##  5  2013 2013-01-01 06:00:00 LGA    ATL   N668DN  DL              1991
##  6  2013 2013-01-01 05:00:00 EWR    ORD   N39463  UA              2012
##  7  2013 2013-01-01 06:00:00 EWR    FLL   N516JB  B6              2000
##  8  2013 2013-01-01 06:00:00 LGA    IAD   N829AS  EV              1998
##  9  2013 2013-01-01 06:00:00 JFK    MCO   N593JB  B6              2004
## 10  2013 2013-01-01 06:00:00 LGA    ORD   N3ALAA  AA                NA
## # ... with 336,766 more rows, 3 more variables: type <chr>,
## #   engine <chr>, seats <int>, and abbreviated variable name
## #   1: manufacture_year
```

Gov 50: 12. Prediction and Iteration

Matthew Blackwell

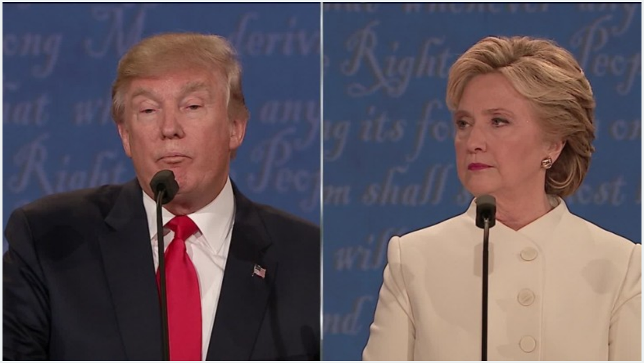
Harvard University

Roadmap

1. Prediction
2. Loops
3. Evaluating the predictions
4. Time-series plot

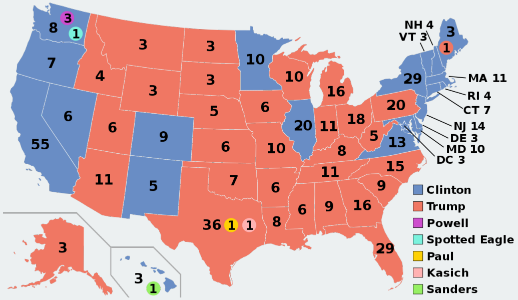
1/ Prediction

2016 US Presidential Election



- 2016 election popular vote:
    - Clinton: 65,853,516 (48.2%)
    - Trump: 62,984,825 (46.1%)
- Why did Trump win? **Electoral college**
    - Trump: 304, Clinton: 227
- Election determined by 77,744 votes (margins in WI, MI, and PA)
    - 0.056% of the electorate (~136 million)

Predicting US Presidential Elections



- **Electoral college system**
    - Must win an absolute majority of 538 electoral votes
    - 538 = 435 (House of Representatives) + 100 (Senators) + 3 (DC)
    - Must win at least 270 votes
    - nobody wins an absolute majority $\rightsquigarrow$ House vote
- Must predict winner of each state

Prediction strategy

- Predict state-level support for each candidate using polls
- Allocate electoral college votes of that state to its predicted winner
- Aggregate EC votes across states to determine the predicted winner
- Coding strategy:
    1. For each state, subset to polls within that state.
    2. Further subset the latest polls
    3. Average the latest polls to estimate support for each candidate
    4. Allocate the electoral votes to the candidate who has greatest support
    5. Repeat this for all states and aggregate the electoral votes
- Sounds like a lot of subsets, ugh...

2/ Loops

A simple example

What if we wanted to know the number of unique values of each column of the `cces_2020` data?

```r
library(gov50data)
cces_2020
```

```
## # A tibble: 51,551 x 6
##    gender race  educ                 pid3    turno~1 pres_~2
##    <fct>  <fct> <fct>                <fct>     <dbl> <fct>
##  1 Male   White 2-year               Republ~       1 Donald~
##  2 Female White Post-grad            Democr~      NA <NA>
##  3 Female White 4-year               Indepe~       1 Joe Bi~
##  4 Female White 4-year               Democr~       1 Joe Bi~
##  5 Male   White 4-year               Indepe~       1 Other
##  6 Male   White Some college         Republ~       1 Donald~
##  7 Male   Black Some college         Not su~      NA <NA>
##  8 Female White Some college         Indepe~       1 Donald~
##  9 Female White High school graduate Republ~       1 Donald~
## 10 Female White 4-year               Democr~       1 Joe Bi~
## # ... with 51,541 more rows, and abbreviated variable names
## #   1: turnout_self, 2: pres_vote
```

Manually changing values

```r
length(unique(cces_2020$gender))
```

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

```r
length(unique(cces_2020$race))
```

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

```r
length(unique(cces_2020$educ))
```

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

```r
length(unique(cces_2020$pid3))
```

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

```r
length(unique(cces_2020$turnout_self))
```

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

```r
length(unique(cces_2020$pres_vote))
```

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

Subsetting with brackets

Note that we can also access variables with `[[]]`:

```r
unique(cces_2020$gender)
```

```
## [1] Male   Female
## Levels: Male Female skipped not asked
```

```r
unique(cces_2020[[1]])
```

```
## [1] Male   Female
## Levels: Male Female skipped not asked
```

```r
unique(cces_2020$pid3)
```

```
## [1] Republican  Democrat    Independent Not sure
## [5] Other
## 7 Levels: Democrat Republican Independent ... not asked
```

```r
unique(cces_2020[[4]])
```

```
## [1] Republican  Democrat    Independent Not sure
## [5] Other
## 7 Levels: Democrat Republican Independent ... not asked
```

Manually changing values, alternative

```r
length(unique(cces_2020[[1]]))
```

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

```r
length(unique(cces_2020[[2]]))
```

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

```r
length(unique(cces_2020[[3]]))
```

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

```r
length(unique(cces_2020[[4]]))
```

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

```r
length(unique(cces_2020[[5]]))
```

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

```r
length(unique(cces_2020[[6]]))
```

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

Recognizing the template

What if you had more values? Not efficient!

Recognize the template:

```
length(unique(cces_2020[[<<column number>>]]))
```

Can we give R this template and a set of column numbers have it do our task repeatedly?

Loops in R

for loop provide a way to execute these templates multiple times:

```
output <- rep(NA, times = ncol(cces_2020))     # 1. output
for (i in seq_along(cces_2020)) {              # 2. sequence
  output[i] <- length(unique(cces_2020[[i]]))  # 3. body
}
output
```

```
## [1] 2 8 6 5 3 7
```

- Elements of a loop:
  1. `output`: vector to hold the
  2. `i`: placeholder name we'll use to swap values between iterations.
  3. `seq_along(cces_2020)`: vector of values we want the placeholder to take.
  4. body: a set of expressions that will be repeatedly evaluated.
  5. `{}`: curly braces to define beginning and end of the loop.
- Indentation is important for readability of the code.

2020 polling prediction

Election data: `pres20`

Name	Description
`state`	abbreviated name of state
`biden`	Biden's vote share (percentage)
`trump`	Trump's vote share (percentage)
`ev`	number of electoral college votes for the state

Polling data `polls20`:

Name	Description
`state`	state in which poll was conducted
`end_date`	end date the period when poll was conducted
`daysleft`	number of days between end date and election day
`pollster`	name of organization conducting poll
`sample_size`	name of organization conducting poll
`biden`	predicted support for Biden (percentage)
`trump`	predicted support for Trump (percentage)

Some preprocessing

```r
library(gov50data)

# calculate Trump's margin of victory
polls20 <- polls20 |>
  mutate(margin = biden - trump)
pres20 <- pres20 |>
  mutate(margin = biden - trump)

glimpse(polls20)
```

```
## Rows: 2,445
## Columns: 8
## $ end_date    <date> 2020-11-02, 2020-11-02, 2020-11-02, 2~
## $ state       <chr> "FL", "PA", "FL", "FL", "NV", "GA", "S~
## $ days_left   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ pollster    <chr> "The Political Matrix/The Listener Gro~
## $ sample_size <dbl> 966, 499, 400, 1054, 1024, 1041, 817, ~
## $ biden       <dbl> 44.2, 48.4, 47.0, 47.3, 48.4, 45.4, 39~
## $ trump       <dbl> 48.0, 49.2, 48.2, 49.4, 49.1, 49.7, 51~
## $ margin      <dbl> -3.8, -0.8, -1.2, -2.1, -0.7, -4.3, -1~
```

Reminder of our goal

- Coding strategy:
    1. For each state, subset to polls within that state.
    2. Further subset the latest polls
    3. Average the latest polls to estimate support for each candidate
    4. Allocate the electoral votes to the candidate who has greatest support
    5. Repeat this for all states and aggregate the electoral votes

Poll prediction for each state

```r
poll_pred <- rep(NA, 51) # place holder

# get list of unique state names to iterate over
state_names <- sort(unique(polls20$state))

# add labels to holder
names(poll_pred) <- state_names

for (i in 1:51) {
  state_data <- subset(polls20, subset = (state == state_names[i]))

  latest <- state_data$days_left == min(state_data$days_left)

  poll_pred[i] <- mean(state_data$margin[latest])
}

head(poll_pred)
```

```
##      AK     AL     AR     AZ     CA     CO
##   -9.00 -26.00 -23.00   4.25  26.00  11.00
```

Tidyverse alternative version

```r
poll_pred <- polls20 |>
  group_by(state) |>
  filter(days_left == min(days_left)) |>
  summarize(margin_pred = mean(margin))
poll_pred
```

```
## # A tibble: 51 x 2
##    state margin_pred
##    <chr>       <dbl>
##  1 AK         -9
##  2 AL        -26
##  3 AR        -23
##  4 AZ          4.25
##  5 CA         26
##  6 CO         11
##  7 CT         22
##  8 DC         89
##  9 DE         22
## 10 FL          0.0800
## # ... with 41 more rows
```

3/ Evaluating the predictions

Polling errors

Prediction error = actual outcome − predicted outcome

```
poll_pred <- poll_pred |>
  left_join(pres20) |>
  mutate(errors = margin - margin_pred)
poll_pred
```

```
## # A tibble: 51 x 8
##    state margin_pred    ev biden trump  other  margin errors
##    <chr>       <dbl> <dbl> <dbl> <dbl>  <dbl>   <dbl>  <dbl>
##  1 AK         -9         3  42.8 52.8  0.732  -10.1   -1.06
##  2 AL        -26         9  36.6 62.0  0.699  -25.5    0.538
##  3 AR        -23         6  34.8 62.4  0.257  -27.6   -4.62
##  4 AZ          4.25     11  49.4 49.1  0.263    0.309 -3.94
##  5 CA         26        55  63.5 34.3  0.244   29.2    3.16
##  6 CO         11         9  55.0 41.6  0.161   13.4    2.41
##  7 CT         22         7  59.3 39.2  0.129   20.1   -1.93
##  8 DC         89         3  92.1  5.40 0.491   86.8   -2.25
##  9 DE         22         3  58.7 39.8  0.0780  19.0   -3.03
## 10 FL          0.0800   29  47.9 51.2  0.0835  -3.36  -3.44
## # ... with 41 more rows
```

Assessing the prediction error

Bias: average prediction error

```r
mean(poll_pred$errors)
```

```
## [1] -3.98
```

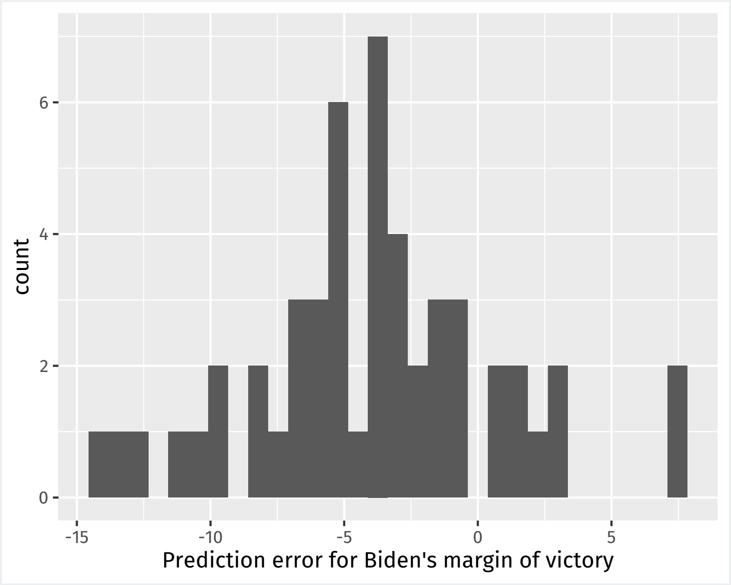
Root mean-square error: average magnitude of the prediction error

```r
sqrt(mean(poll_pred$errors^2))
```

```
## [1] 6.07
```

Histogram of the errors

```r
ggplot(poll_pred, aes(x = errors)) +
  geom_histogram() +
  labs(
    x = "Prediction error for Biden's margin of victory"
  )
```

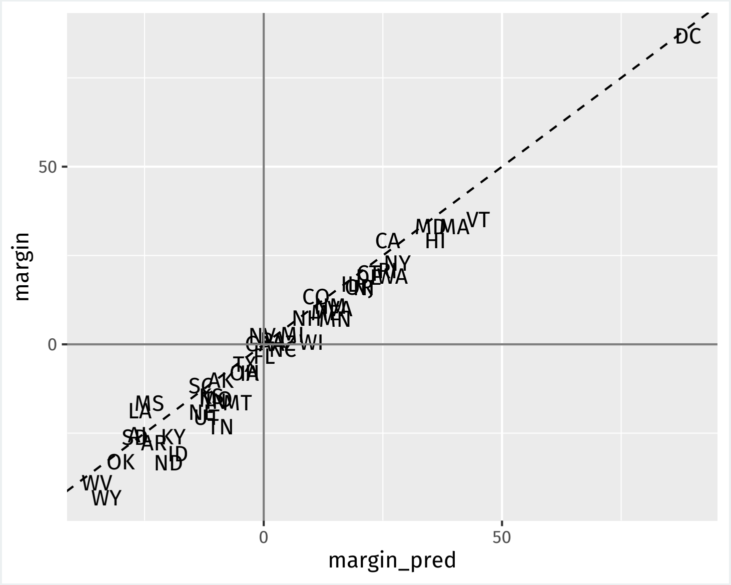

count
0
2
4
6
-15
-10
-5
0
5
Prediction error for Biden's margin of victory

Comparing polls to outcome

Sometimes we want plot text labels instead of point and we use `geom_text` and the `label` aesthetic:

```r
## merge the actual results
ggplot(poll_pred, aes(x = margin_pred, y = margin)) +
  geom_text(aes(label = state)) +
  geom_abline(xintercept = 0, slope = 1, linetype = 2) +
  geom_hline(yintercept = 0, color = ”grey50”) +
  geom_vline(xintercept = 0, color = ”grey50”)
```

Comparing polls to outcome



Classification

Election prediction: need to predict winner in each state:

```r
poll_pred |>
  filter(margin > 0) |>
  summarize(sum(ev)) |> pull()
```

```
## [1] 306
```

```r
poll_pred |>
  filter(margin_pred > 0) |>
  summarize(sum(ev)) |> pull()
```

```
## [1] 328
```

- Prediction of binary outcome variable = **classification problem**
- Wrong prediction $\rightsquigarrow$ misclassification
    1. **true positive**: predict Trump wins when he actually wins.
    2. **false positive**: predict Trump wins when he actually loses.
    3. **true negative**: predict Trump loses when he actually loses.
    4. **false negative**: predict Trump loses when he actually wins.
- Sometimes false negatives are more/less important: e.g., civil war.

Classification based on polls

Accuracy: `sign()` returns 1 for a positive number, `-1` for a negative number, and `0` for 0.

```
poll_pred |>
  summarize(prop_correct = mean(sign(margin_pred) == sign(margin))) |>
  pull()
```

```
## [1] 0.922
```

Which states did polls call wrong?

```
poll_pred |>
  filter(sign(margin_pred) != sign(margin))
```

```
## # A tibble: 4 x 8
##   state margin_pred    ev biden trump  other margin errors
##   <chr>       <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
## 1 FL         0.0800    29  47.9  51.2 0.0835 -3.36   -3.44
## 2 GA        -1.15      16  49.5  49.2 0.0759  0.236   1.39
## 3 NC         3.95      15  48.6  49.9 0.296  -1.35   -5.30
## 4 NV        -0.350      6  50.1  47.7 0.759   2.39    2.74
```

4/ Time-series plot

National polls

We often want to show a time series of the national-level polls to get a sense of the popular vote:

```
national_polls20
```

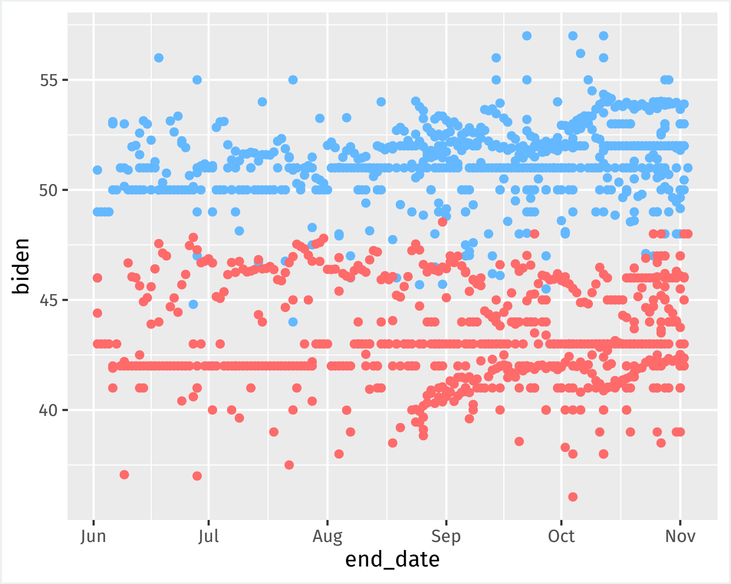
```
## # A tibble: 654 x 5
##    end_date   pollster                           sampl~1 biden trump
##    <date>     <chr>                                <dbl> <dbl> <dbl>
##  1 2020-11-03 Lake Research                         2400  51    48  
##  2 2020-11-02 Research Co.                          1025  50    42  
##  3 2020-11-02 YouGov                                1363  53    43  
##  4 2020-11-02 Ipsos                                  914  52    45  
##  5 2020-11-02 SurveyMonkey                         28240  52    46  
##  6 2020-11-02 HarrisX                               2297  52    48  
##  7 2020-11-02 TIPP                                  1212  50.4  46.0
##  8 2020-11-02 USC Dornsife                          5423  53.9  42.4
##  9 2020-11-01 John Zogby Strategies/EMI~            1008  49.6  43.8
## 10 2020-11-01 Swayable                              5174  51.8  46.1
## # ... with 644 more rows, and abbreviated variable name
## #   1: sample_size
```

Plotting the raw results

```r
national_polls20 |>
  ggplot(aes(x = end_date)) +
  geom_point(aes(y = biden), color = ”steelblue1”) +
  geom_point(aes(y = trump), color = ”indianred1”)
```

Plotting the raw results

Fairly messy:



Clean the mess by taking moving averages

Goal: plot the average of polls in the last 7 days (very difficult with `dplyr`).

Loop over each day in the data and do:

1. Subset to all polls in the previous 7 days of that day.
2. Calculate the average of these polls for Biden and Trump.
3. Save the result as a 1-row tibble.

Dates in R

You can get R to properly understand dates and do arithmetic with them:

```r
head(national_polls20$end_date)
```

```
## [1] "2020-11-03" "2020-11-02" "2020-11-02" "2020-11-02"
## [5] "2020-11-02" "2020-11-02"
```

```r
head(national_polls20$end_date + 3)
```

```
## [1] "2020-11-06" "2020-11-05" "2020-11-05" "2020-11-05"
## [5] "2020-11-05" "2020-11-05"
```

Lubridate to create dates

We can covert a string to a date using the `lubridate` package:

```
"2020-11-03" + 3  ## R doesn't know this is a date yet!
```

```
## Error in "2020-11-03" + 3: non-numeric argument to binary operator
```

```
lubridate::ymd("2020-11-03") + 3
```

```
## [1] "2020-11-06"
```

```
lubridate::mdy("11/03/2020") + 3
```

```
## [1] "2020-11-06"
```

Getting a vector of dates

Setup the vector of dates to cover:

```r
election_day <- lubridate::ymd("2020-11-03")
all_dates <- seq(from = min(national_polls20$end_date) + 1,
                 to = election_day,
                 by = "days")
head(all_dates)
```

```
## [1] "2020-06-03" "2020-06-04" "2020-06-05" "2020-06-06"
## [5] "2020-06-07" "2020-06-08"
```

Moving window loop

```
output <- vector("list", length = length(all_dates))

for (i in seq_along(all_dates)) {
  this_date <- all_dates[[i]]

  this_week <- national_polls20 |>
    filter(
      this_date - end_date >= 0,      # this_date is after end_date
      this_date - end_date < 7       # within a week
    )

  output[[i]] <- this_week |>
    summarize(
      date = this_date,
      biden = mean(biden, na.rm = TRUE),
      trump = mean(trump, na.rm = TRUE)
    )
}
output <- bind_rows(output)
```

Result

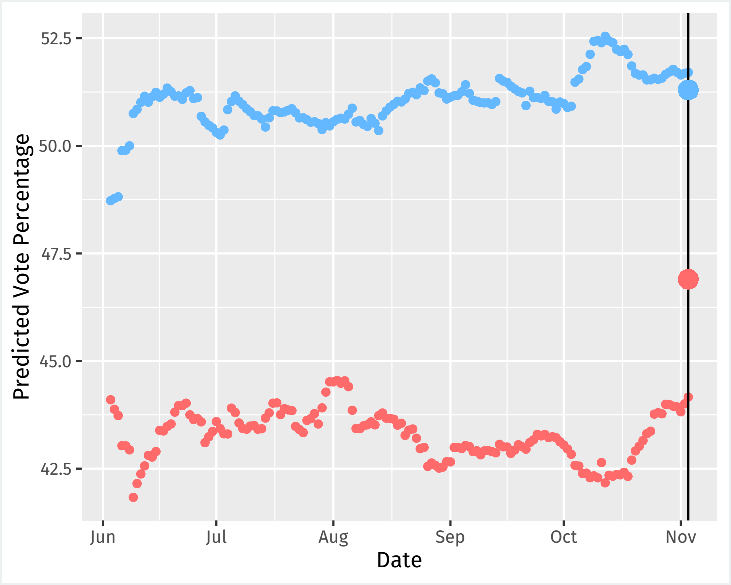
```
output
```

```
## # A tibble: 154 x 3
##    date       biden trump
##    <date>     <dbl> <dbl>
##  1 2020-06-03  48.7  44.1
##  2 2020-06-04  48.8  43.9
##  3 2020-06-05  48.8  43.7
##  4 2020-06-06  49.9  43.0
##  5 2020-06-07  49.9  43.0
##  6 2020-06-08  50    42.9
##  7 2020-06-09  50.8  41.8
##  8 2020-06-10  50.8  42.2
##  9 2020-06-11  51.0  42.4
## 10 2020-06-12  51.2  42.6
## # ... with 144 more rows
```

Let's plot

```r
output |>
  ggplot(aes(x = date)) +
  geom_point(aes(y = biden), color = "steelblue1") +
  geom_point(aes(y = trump), color = "indianred1") +
  geom_vline(xintercept = election_day) +
  geom_point(aes(x = election_day, y = 51.3), color = "steelblue1", size =
  geom_point(aes(x = election_day, y = 46.9), color = "indianred1", size =
  labs(
    x = "Date",
    y = "Predicted Vote Percentage"
  )
```

Let's plot



52.5
50.0
47.5
45.0
42.5
Predicted Vote Percentage
Jun
Jul
Aug
Sep
Oct
Nov
Date

Gov 50: 13. Regression

Matthew Blackwell

Harvard University

Roadmap

1. Prediction
2. Modeling with a line
3. Linear regression in R

1/ Prediction

Predicting my weight

Predicting weight with activity: `health` data

Name	Description
`date`	date of measurements
`active_calories`	calories burned
`steps`	number of steps taken (in 1,000s)
`weight`	weight (lbs)
`steps_lag`	steps on day before (in 1,000s)
`calories_lag`	calories burned on day before

Predicting using bivariate relationship

- Goal: what's our best guess about Y_i if we know what X_i is?
    - what's our best guess about my weight this morning if I know how many steps I took yesterday?
- Terminology:
    - **Dependent/outcome variable**: what we want to predict (weight).
    - **Independent/explanatory variable**: what we're using to predict (steps).

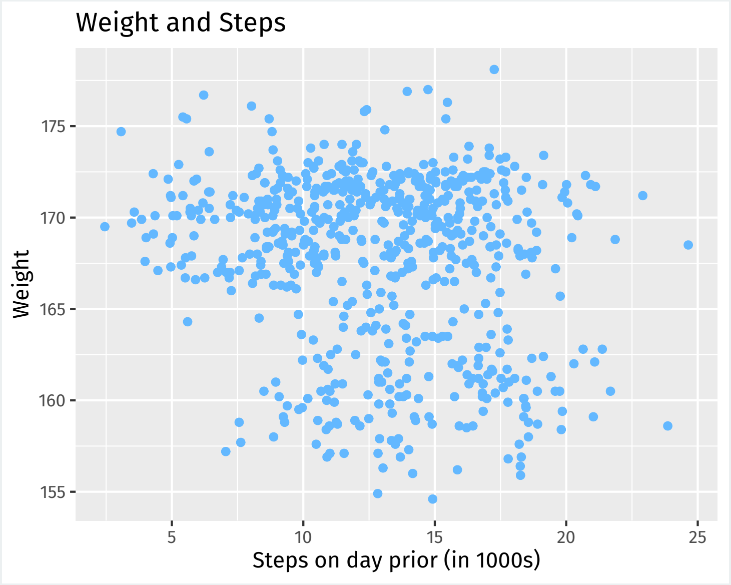
Weight data

- Load the data:

```r
library(gov50data)
health <- drop_na(health)
```

- Plot the data:

```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = ”steelblue1”) +
  labs(
    x = ”Steps on day prior (in 1000s)”,
    y = ”Weight”,
    title = ”Weight and Steps”
  )
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Prediction one variable with another

- Prediction with access to just Y: average of the Y values.
- Prediction with another variable: for any value of X, what's the best guess about Y?
    - Need a function $y = f(x)$ that maps values of X into predictions.
    - **Machine learning**: fancy ways to determine $f(x)$
- Example: what if did 5,000 steps today? What's my best guess about weight?

Start with looking at a narrow strip of X

Let's find all values that round to 5,000 steps:

```
health |>
  filter(round(steps_lag) == 5)
```

```
## # A tibble: 12 x 6
##    date       active.calories steps weight steps_lag calor~1
##    <date>               <dbl> <dbl>  <dbl>     <dbl>   <dbl>
##  1 2015-09-08           1111. 15.2    169.      5.02    410.
##  2 2015-12-12            728. 14.7    167.      5.36    259.
##  3 2015-12-28            430.  8.94   170.      5.19    314
##  4 2016-01-29            475.  8.26   171.      4.95    314.
##  5 2016-02-14            264.  5.42   172.      4.86    297.
##  6 2016-02-15            892. 13.1    171.      5.42    264.
##  7 2016-05-02            627. 11.8    170.      5.04    283.
##  8 2016-06-27            352.  7.21   169.      4.93    212.
##  9 2016-07-22            766. 14.8    167.      4.96    251.
## 10 2016-11-25            452   9.4    173.      5.26    295
## 11 2016-11-28            577. 11.8    171.      4.97    304.
## 12 2016-12-30            621. 12.4    176.      5.42    371.
## # ... with abbreviated variable name 1: calorie_lag
```

Best guess about Y for this X

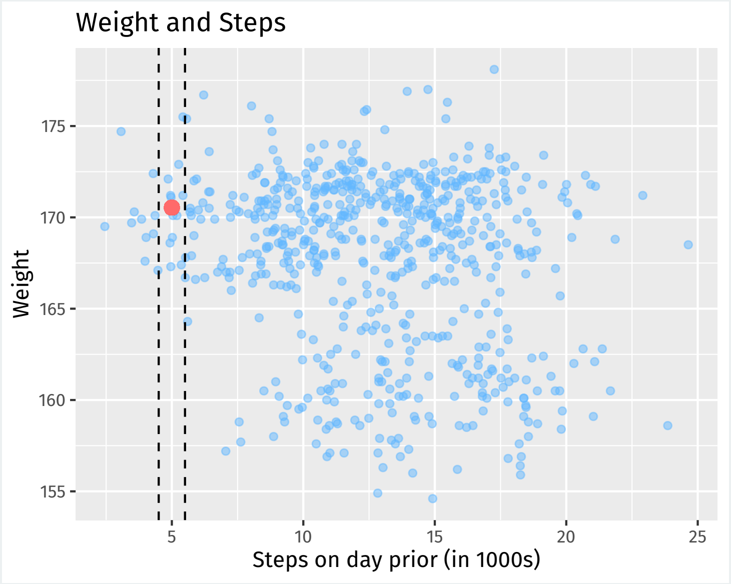
Best prediction about weight for a step count of roughly 5,000 is the average weight for observations around that value:

```
mean_wt_5k_steps <- health |>
  filter(round(steps_lag) == 5) |>
  summarize(mean(weight)) |>
  pull()
mean_wt_5k_steps
```

```
## [1] 171
```

Plotting the best guess

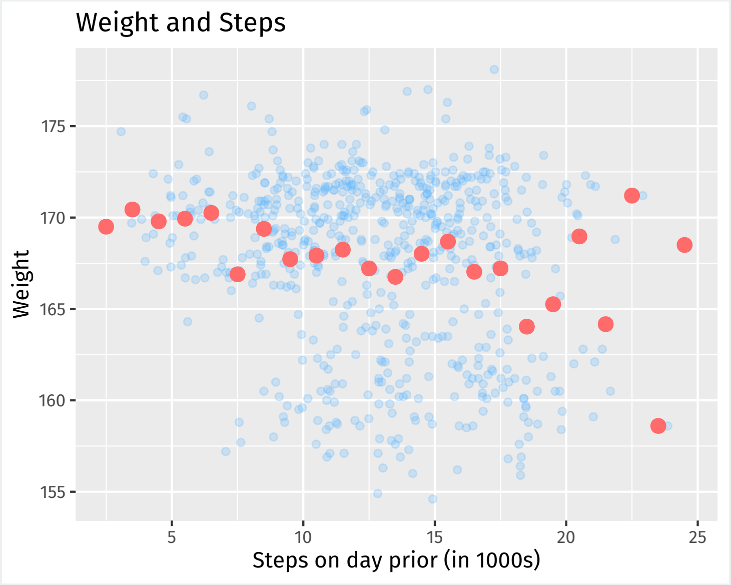
```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.5) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_vline(xintercept = c(4.5, 5.5), linetype = "dashed") +
  geom_point(aes(x = 5, y = mean_wt_5k_steps), color = "indianred1",
             size = 3)
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Binned means

We can use a `stat_summary_bin()` to add these binned means all over the scatter plot:

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 3,
                   geom = "point", binwidth = 1)
```

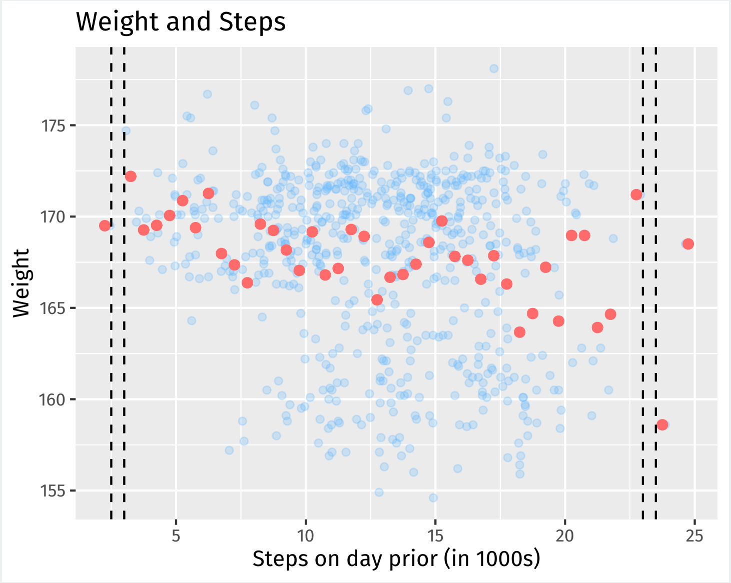

Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Smaller bins

But what happens when we make the bins too small?

```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 2,
                   geom = "point", binwidth = 0.5) +
  geom_vline(xintercept = c(2.5, 3, 23, 23.5), linetype = "dashed")
```

Gaps and bumps:



2/ Modeling with a line

Using a line to predict

- Can we smooth out these binned means and close gaps? **A model.**
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

- Problem: for any line we draw, not all the data is on the line.
    - Some points will be above the line, some below.
    - Need a way to account for **chance variation** away from the line.

Linear regression model

- Model for the line of best fit:

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \cdot X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit.
- **Chance error** ϵ_i: accounts for the fact that the line doesn't perfectly fit the data.
    - Each observation allowed to be off the regression line.
    - Chance errors are 0 on average.
- Useful fiction: this model represents the **data generating process**
    - George Box: "all models are wrong, some are useful"

Interpreting the regression line

$$Y_i = \alpha + \beta \cdot X_i + \epsilon_i$$

- **Intercept** α: average value of Y when X is 0
    - Average weight when I take 0 steps the day prior.
- **Slope** β: average change in Y when X increases by one unit.
    - Average decrease in weight for each additional 1,000 steps.

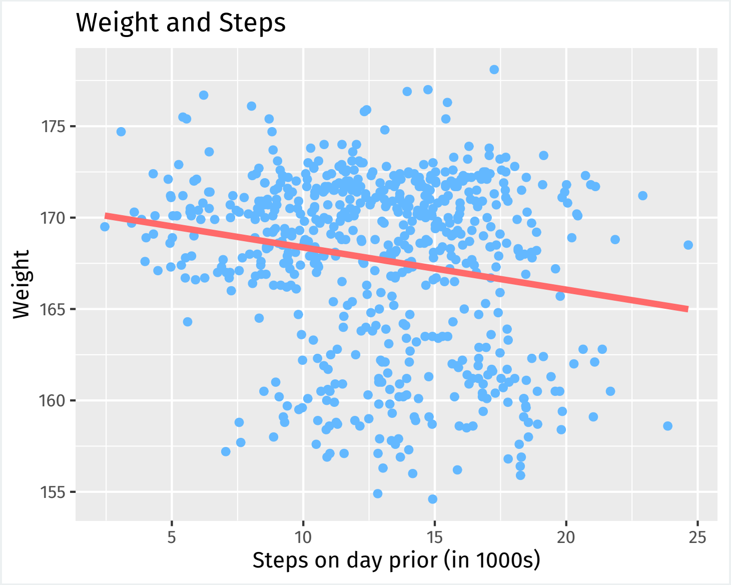
Estimated coefficients

- Parameters: α, β
  - Unknown features of the **data-generating process**.
  - Chance error makes these impossible to observe directly.
- Estimates: $\hat{\alpha}, \hat{\beta}$
  - An **estimate** is our best guess about some parameter.
- **Regression line**: $\widehat{Y} = \hat{\alpha} + \hat{\beta} \cdot x$
  - Average value of Y when X is equal to x.
  - Represents the best guess or **predicted value** of the outcome at x.

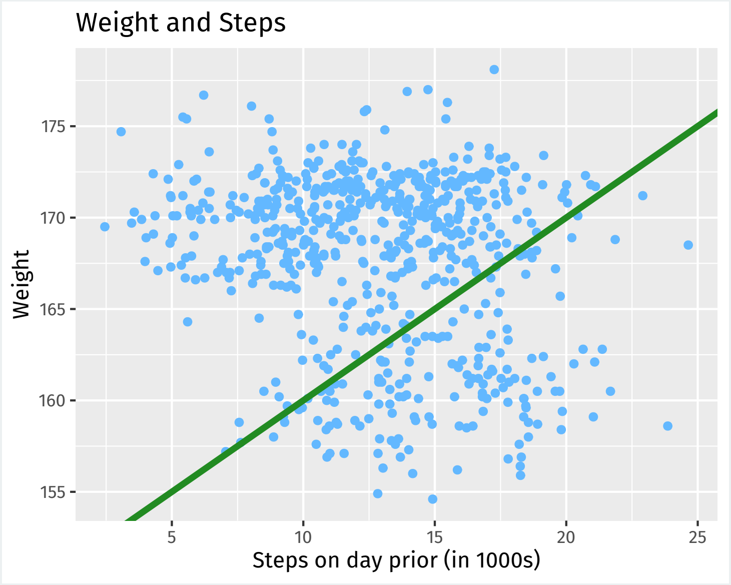
Line of best fit

```r
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1") +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```

Line of best fit



Why not this line?



Prediction error

Let's understand the **prediction error** for a line with intercept a and slope b.

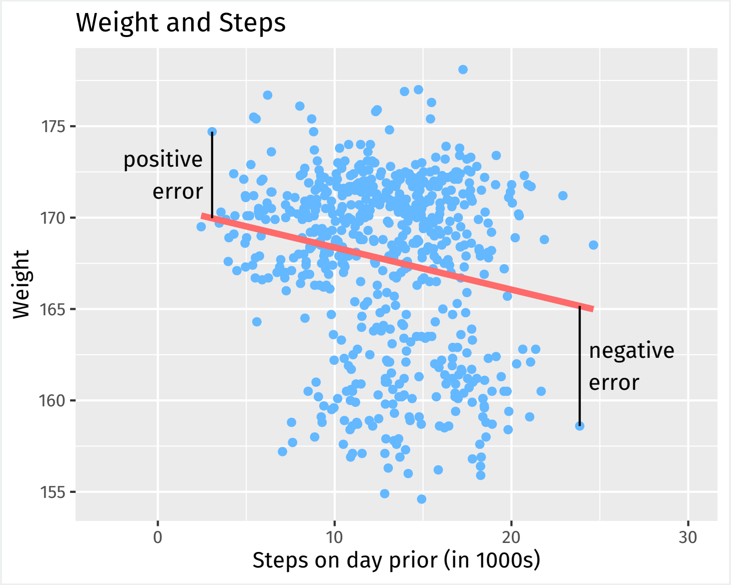
Fitted/predicted value for unit i:

$$a + b \cdot X_i$$

Preidiction error (residual):

$$\text{error = actual - predicted} = Y_i - (a + b \cdot X_i)$$

Prediction errors/residuals



Least squares

- Get these estimates by the **least squares method**.
- Minimize the **sum of the squared residuals** (SSR):

$$\text{SSR} = \sum_{i=1}^{n} (\text{prediction error}_i)^2 = \sum_{i=1}^{n} (Y_i - a - b \cdot X_i)^2$$

- Finds the line that minimizes the magnitude of the prediction errors!

3/ Linear regression in R

Linear regression in R

- R will calculate least squares line for a data set using `lm( )`.
  - Syntax: `lm(y ~ x, data = mydata)`
  - `y` is the name of the dependent variance
  - `x` is the name of the independent variable
  - `mydata` is the data.frame where they live

```
fit <- lm(weight ~ steps_lag, data = health)
fit
```

```
##
## Call:
## lm(formula = weight ~ steps_lag, data = health)
##
## Coefficients:
## (Intercept)    steps_lag
##     170.675       -0.231
```

Coefficients

Use `coef()` to extract estimated coefficients:

```
coef(fit)
```

```
## (Intercept)   steps_lag
##     170.675      -0.231
```

Interpretation: a 1-unit increase in X (1,000 steps) is associated with a decrease in the average weight of 0.231 pounds.

Question: what would this model predict about the change in average weight for a 10,000 step increase in steps?

broom package

The broom package can provide nice summaries of the regression output.

`augment()` can show fitted values, residuals and other unit-level statistics:

```
library(broom)
augment(fit) |> head()
```

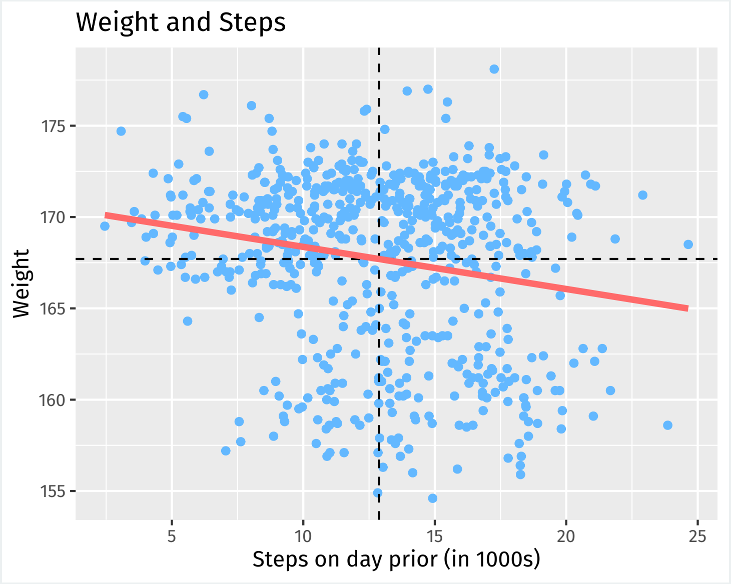
```
## # A tibble: 6 x 8
##   weight steps_lag .fitted  .resid    .hat .sigma    .cooksd
##    <dbl>     <dbl>   <dbl>   <dbl>   <dbl>  <dbl>      <dbl>
## 1   169.      17.5    167.  2.46   0.00369   4.68    5.13e-4
## 2   168       18.4    166.  1.57   0.00463   4.68    2.64e-4
## 3   167.      19.6    166.  1.05   0.00609   4.68    1.54e-4
## 4   168.      10.4    168. -0.0750 0.00217   4.68    2.80e-7
## 5   168.      18.7    166.  1.44   0.00496   4.68    2.38e-4
## 6   166.       9.14   169. -2.27   0.00296   4.68    3.49e-4
## # ... with 1 more variable: .std.resid <dbl>
```

Properties of least squares

Least squares line always goes through $(\overline{X}, \overline{Y})$.

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1") +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  geom_hline(yintercept = mean(health$weight), linetype = "dashed") +
  geom_vline(xintercept = mean(health$steps_lag), linetype = "dashed") +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```

Least squares line always goes through $(\overline{X}, \overline{Y})$.



Properties of least squares line

Estimated slope is related to correlation:

$$\hat{\beta} = (\text{correlation of } X \text{ and } Y) \times \frac{\text{SD of } Y}{\text{SD of } X}$$

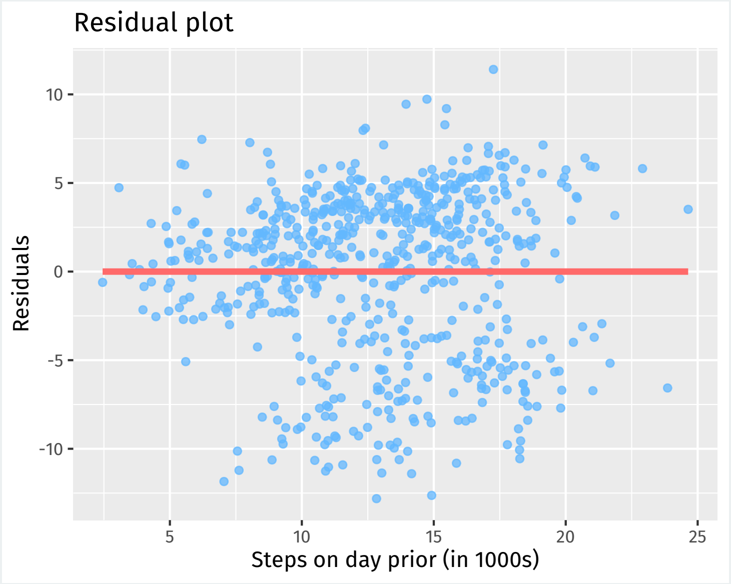
Mean of residuals is always 0.

```r
augment(fit) |>
  summarize(mean(.resid))
```

```
## # A tibble: 1 x 1
##   `mean(.resid)`
##            <dbl>
## 1      -1.21e-13
```

Plotting the residuals

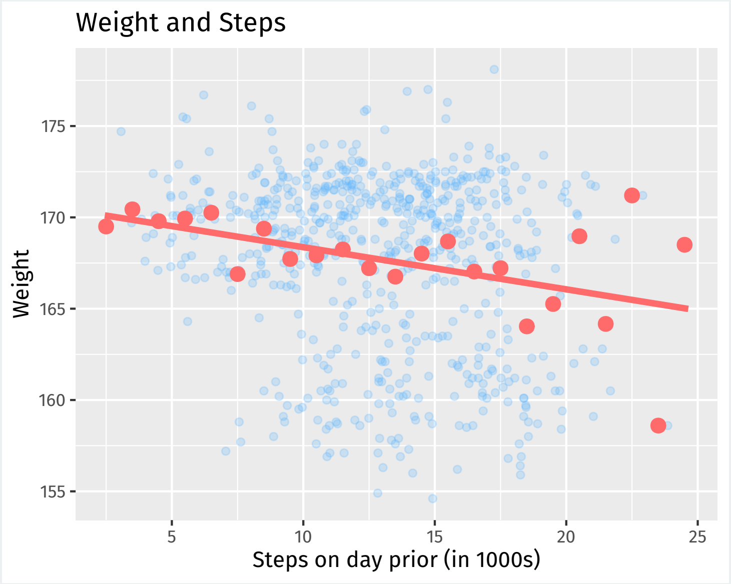
```r
augment(fit) |>
  ggplot(aes(x = steps_lag, y = .resid)) +
  geom_point(color = "steelblue1", alpha = 0.75) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Residuals",
    title = "Residual plot"
  ) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```


Residual plot
Residuals
10
5
0
-5
-10
5
10
15
20
25
Steps on day prior (in 1000s)

Smoothed graph of averages

Another way to think of the regression line is a smoothed version of the binned means plot:

```
ggplot(health, aes(x = steps_lag, y = weight)) +
  geom_point(color = "steelblue1", alpha = 0.25) +
  labs(
    x = "Steps on day prior (in 1000s)",
    y = "Weight",
    title = "Weight and Steps"
  ) +
  stat_summary_bin(fun = "mean", color = "indianred1", size = 3,
                   geom = "point", binwidth = 1) +
  geom_smooth(method = "lm", se = FALSE, color = "indianred1", size = 1.5)
```


Weight and Steps
Weight
175
170
165
160
155
5
10
15
20
25
Steps on day prior (in 1000s)

Gov 50: 14. More Regression and Model Fit

Matthew Blackwell

Harvard University

Roadmap

1. Model fit

2. Multiple regression

1/ Model fit

Presidential popularity and the midterms

- Does popularity of the president or recent changes in the economy better predict midterm election outcomes?

Name	Description
`year`	midterm election year
`president`	name of president
`party`	Democrat or Republican
`approval`	Gallup approval rating at midterms
`rdi_change`	% change in real disposable income over the year before midterms
`seat_change`	change in the number of House seats for the president's party

```r
library(gov50data)
midterms
```

```
## # A tibble: 20 x 6
##     year president  party approval seat_change rdi_change
##    <dbl> <chr>      <chr>    <dbl>       <dbl>      <dbl>
##  1  1946 Truman     D           33         -55     NA    
##  2  1950 Truman     D           39         -29      8.2  
##  3  1954 Eisenhower R           61          -4      1    
##  4  1958 Eisenhower R           57         -47      1.1  
##  5  1962 Kennedy    D           61          -4      5    
##  6  1966 Johnson    D           44         -47      5.3  
##  7  1970 Nixon      R           58          -8      6.6  
##  8  1974 Ford       R           54         -43      6.4  
##  9  1978 Carter     D           49         -11      7.7  
## 10  1982 Reagan     R           42         -28      4.8  
## 11  1986 Reagan     R           63          -5      5.1  
## 12  1990 H.W. Bush  R           58          -8      5.6  
## 13  1994 Clinton    D           46         -53      3.9  
## 14  1998 Clinton    D           66           5      5.6  
## 15  2002 W. Bush    R           63           6      2.6  
## 16  2006 W. Bush    R           38         -30      5.7  
## 17  2010 Obama      D           45         -63      3.5  
## 18  2014 Obama      D           40         -13      4.6  
## 19  2018 Trump      R           38         -42      4.1  
## 20  2022 Biden      D           42          NA     -0.003
```

Fitting the approval model

```
fit.app <- lm(seat_change ~ approval, data = midterms)
fit.app
```

```
##
## Call:
## lm(formula = seat_change ~ approval, data = midterms)
##
## Coefficients:
## (Intercept)     approval
##      -96.58         1.42
```

For a one-point increase in presidential approval, the predicted seat change increases by 1.42

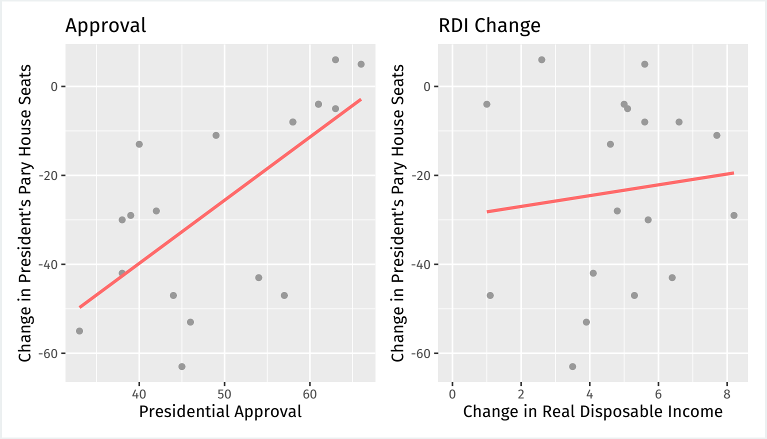
Fitting the income model

```
fit.rdi <- lm(seat_change ~ rdi_change, data = midterms)
fit.rdi
```

```
##
## Call:
## lm(formula = seat_change ~ rdi_change, data = midterms)
##
## Coefficients:
## (Intercept)   rdi_change
##      -29.41         1.21
```

For a one-point increase in the change in real disposable income, the predicted seat change increases by 1.21

Comparing models



- How well do the models "fit the data"?
    - How well does the model predict the outcome variable in the data?

Model fit

Model prediction error:

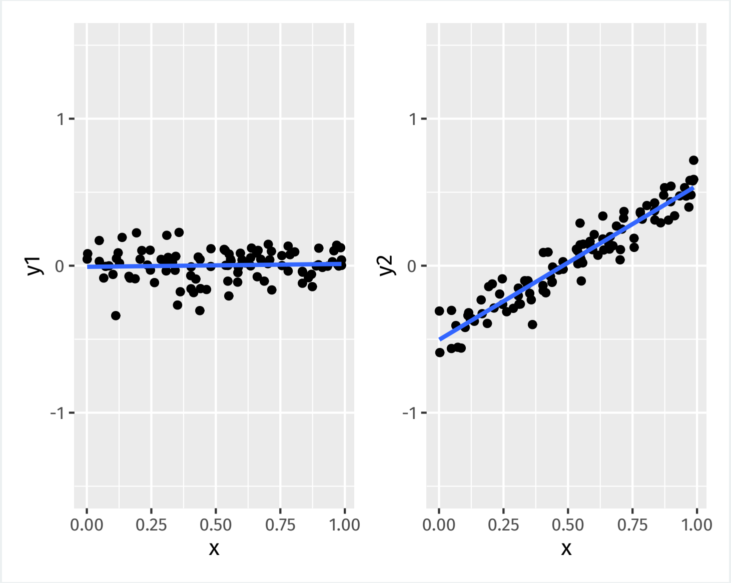
$$\text{prediction error} = \sum_{i=1}^{n} (\text{actual}_i - \text{predicted}_i)^2$$

Prediction error for regression: **Sum of squared residuals**

$$\text{SSR} = \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i \right)^2$$

Lower SSR is better, right?

These two regression lines have approximately the same SSR:



Benchmarking model fit

Benchmarking our predictions using the **proportional reduction in error**:

$$\frac{\text{reduction in prediction error using model}}{\text{baseline prediction error}}$$

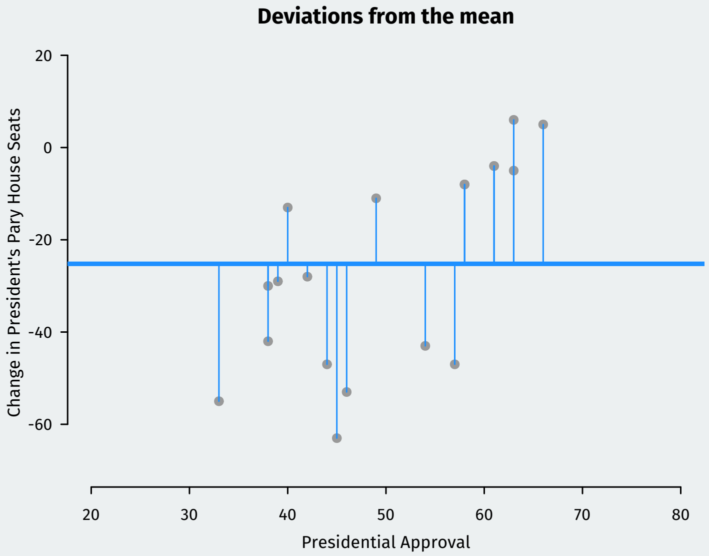
Baseline prediction error without a regression is using the mean of Y to predict. This is called the **Total sum of squares**:

$$\text{TSS} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2$$

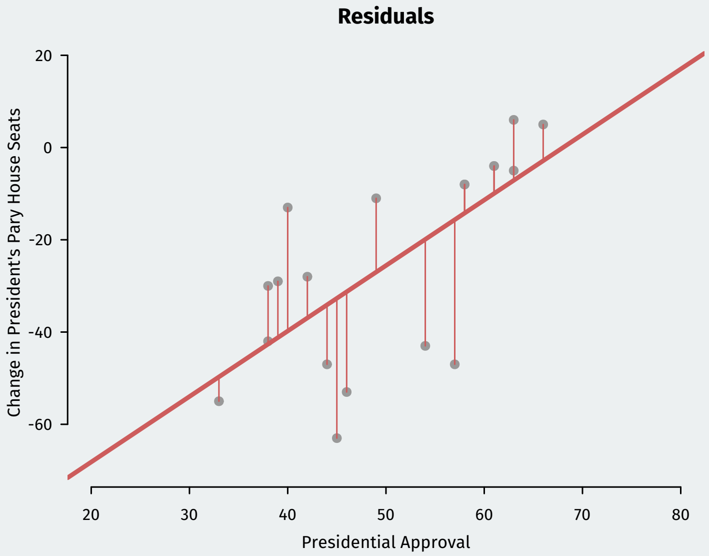
Leads to the **coefficient of determination**, R^2, one summary of LS model fit:

$$R^2 = \frac{TSS - SSR}{TSS} = \frac{\text{how much smaller LS prediction errors are vs mean}}{\text{prediction error using the mean}}$$

Total SS vs SSR



Total SS vs SSR



Model fit in R

- To access R^2 from the `lm()` output, use the `summary()` function:

```r
fit.app.sum <- summary(fit.app)
fit.app.sum$r.squared
```

```
## [1] 0.45
```

- Compare to the fit using change in income:

```r
fit.rdi.sum <- summary(fit.rdi)
fit.rdi.sum$r.squared
```

```
## [1] 0.012
```

- Which does a better job predicting midterm election outcomes?

Accessing model fit via `broom` package

We can also access summary statistics like model fit using the `glance()` function from `broom`:

```r
library(broom)
glance(fit.app)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r~1 sigma stati~2 p.value    df logLik   AIC
##       <dbl>   <dbl> <dbl>   <dbl>   <dbl> <dbl>  <dbl> <dbl>
## 1     0.450   0.418  16.9    13.9 0.00167     1  -79.6  165.
## # ... with 4 more variables: BIC <dbl>, deviance <dbl>,
## #   df.residual <int>, nobs <int>, and abbreviated variable
## #   names 1: adj.r.squared, 2: statistic
```

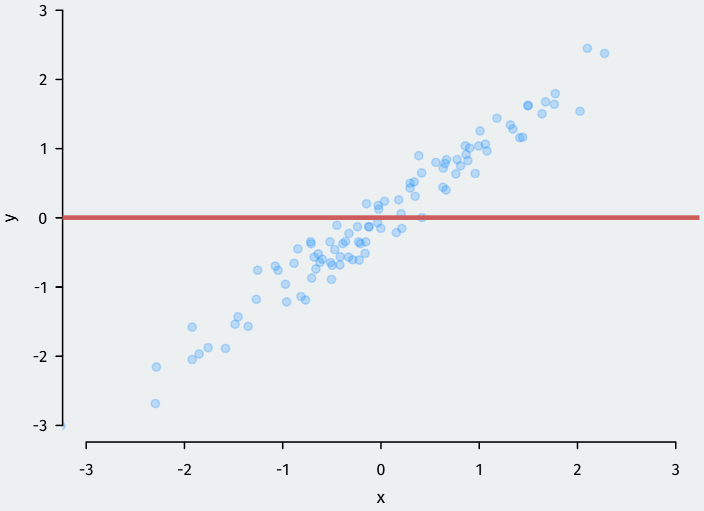
Fake data, better fit

- Little hard to see what's happening in that example.
- Let's look at fake variables x and y:

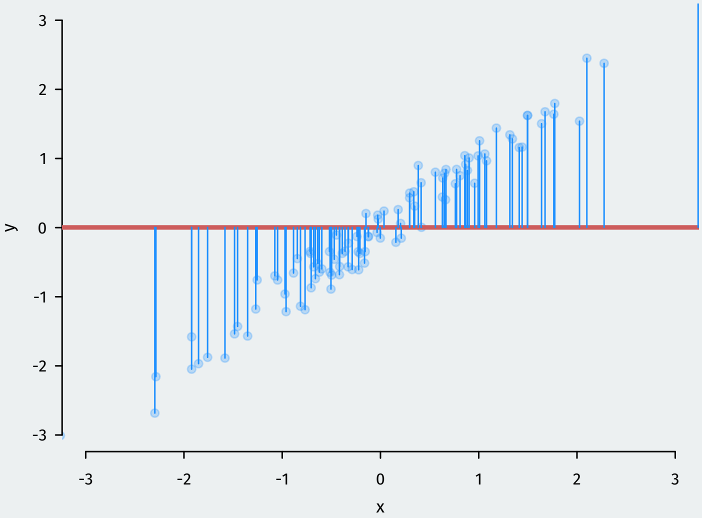
```r
fit.x <- lm(y ~ x)
```

- Very good model fit: $R^2 \approx 0.95$

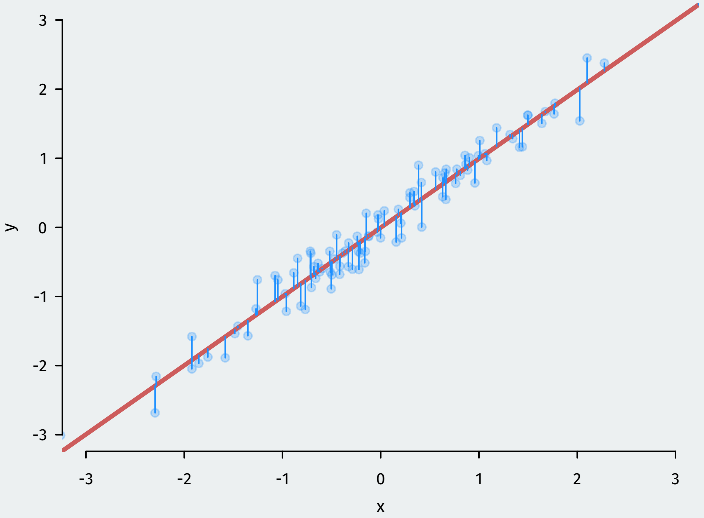
Fake data, better fit



Fake data, better fit

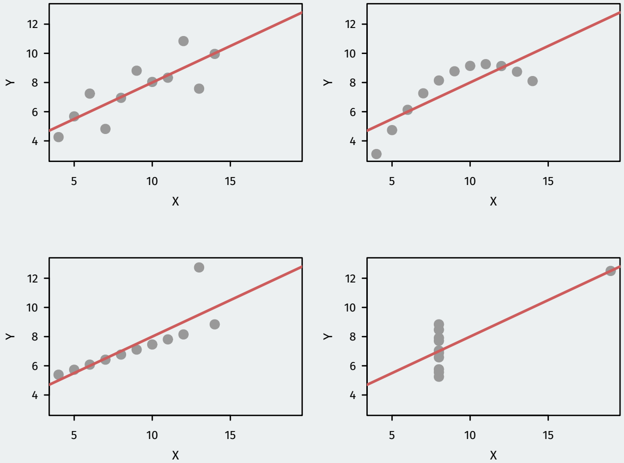


Fake data, better fit



Is R-squared useful?

- Can be very misleading. Each of these samples have the same R^2 even though they are vastly different:



Overfitting

- **In-sample fit**: how well your model predicts the data used to estimate it.
    - R^2 is a measure of in-sample fit.
- **Out-of-sample fit**: how well your model predicts new data.
- **Overfitting**: OLS optimizes in-sample fit; may do poorly out of sample.
    - Example: predicting winner of Democratic presidential primary with gender of the candidate.
    - Until 2016, gender was a **perfect** predictor of who wins the primary.
    - Prediction for 2016 based on this: Bernie Sanders as Dem. nominee.
    - Bad out-of-sample prediction due to overfitting!

2/ Multiple regression

Multiple predictors

What if we want to predict Y as a function of many variables?

$$\texttt{seat\_change}_i = \alpha + \beta_1 \texttt{approval}_i + \beta_2 \texttt{rdi\_change}_i + \epsilon_i$$

Why?

- Better predictions (at least in-sample).
- Better interpretation as **ceteris paribus** relationships:
  - β_1 is the relationship between `approval` and `seat_change` holding `rdi_change` constant.
  - **Statistical control** in a cross-sectional study.

Multiple regression in R

```r
mult.fit <- lm(seat_change ~ approval + rdi_change,
               data = midterms)
mult.fit
```

```
##
## Call:
## lm(formula = seat_change ~ approval + rdi_change, data = midterms)
##
## Coefficients:
## (Intercept)     approval   rdi_change
##     -117.23         1.53         3.22
```

- $\hat{\alpha} =$ -117.2: average seat change president has 0% approval and no change in income levels.
- $\hat{\beta}_1 =$ 1.53: average increase in seat change for additional percentage point of approval, **holding RDI change fixed**
- $\hat{\beta}_2 =$ 3.217: average increase in seat change for each additional percentage point increase of RDI, **holding approval fixed**

Least squares with multiple regression

- How do we estimate the coefficients?
- The same exact way as before: minimize prediction error!
- Residuals (aka prediction error) with multiple predictors:

$$Y_i - \widehat{Y}_i = \texttt{seat\_change}_i - \hat{\alpha} - \hat{\beta}_1 \texttt{approval}_i - \hat{\beta}_2 \texttt{rdi\_change}_i$$

- Find the coefficients that minimizes the **sum of the squared residuals**:

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = (Y_i - \hat{\alpha} - \hat{\beta}_1 X_{i1} - \hat{\beta}_2 X_{i2})^2$$

Model fit with multiple predictors

- R^2 mechanically increases when you add a variables to the regression.
    - But this could be overfitting!!
- Solution: penalize regression models with more variables.
    - Occam's razor: **simpler models are preferred**
- Adjusted R^2: lowers regular R^2 for each additional covariate.
    - If the added covariates doesn't help predict, adjusted R^2 goes down!

Comparing model fits

```r
glance(fit.app) |>
  select(r.squared, adj.r.squared, sigma)
```

```
## # A tibble: 1 x 3
##   r.squared adj.r.squared sigma
##       <dbl>         <dbl> <dbl>
## 1     0.450         0.418  16.9
```

```r
glance(mult.fit) |>
  select(r.squared, adj.r.squared, sigma)
```

```
## # A tibble: 1 x 3
##   r.squared adj.r.squared sigma
##       <dbl>         <dbl> <dbl>
## 1     0.468         0.397  16.7
```

Gov 50: 15. Multiple Regression and Interpretation

Matthew Blackwell

Harvard University

Roadmap

1. Multiple regression
2. Categorical independent variables

1/ Multiple regression

Multiple predictors

What if we want to predict Y as a function of many variables?

$$\texttt{seat\_change}_i = \alpha + \beta_1 \texttt{approval}_i + \beta_2 \texttt{rdi\_change}_i + \epsilon_i$$

Why?

- Better predictions (at least in-sample).
- Better interpretation as **ceteris paribus** relationships:
    - β_1 is the relationship between `approval` and `seat_change` holding `rdi_change` constant.
    - **Statistical control** in a cross-sectional study.

Multiple regression in R

```r
mult.fit <- lm(seat_change ~ approval + rdi_change,
               data = midterms)
mult.fit
```

```
##
## Call:
## lm(formula = seat_change ~ approval + rdi_change, data = midterms)
##
## Coefficients:
## (Intercept)     approval   rdi_change
##     -117.23         1.53         3.22
```

- $\hat{\alpha} =$ -117.2: average seat change president has 0% approval and no change in income levels.
- $\hat{\beta}_1 =$ 1.53: average increase in seat change for additional percentage point of approval, **holding RDI change fixed**
- $\hat{\beta}_2 =$ 3.217: average increase in seat change for each additional percentage point increase of RDI, **holding approval fixed**

Least squares with multiple regression

- How do we estimate the coefficients?
- The same exact way as before: minimize prediction error!
- Residuals (aka prediction error) with multiple predictors:

$$Y_i - \widehat{Y}_i = \texttt{seat\_change}_i - \hat{\alpha} - \hat{\beta}_1 \texttt{approval}_i - \hat{\beta}_2 \texttt{rdi\_change}_i$$

- Find the coefficients that minimizes the **sum of the squared residuals**:

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = (Y_i - \hat{\alpha} - \hat{\beta}_1 X_{i1} - \hat{\beta}_2 X_{i2})^2$$

Model fit with multiple predictors

- R^2 mechanically increases when you add a variables to the regression.
  - But this could be overfitting!!
- Solution: penalize regression models with more variables.
  - Occam's razor: **simpler models are preferred**
- Adjusted R^2: lowers regular R^2 for each additional covariate.
  - If the added covariates doesn't help predict, adjusted R^2 goes down!

Comparing model fits

```r
library(broom)
fit.app <- lm(seat_change ~ approval, data = midterms)
glance(fit.app) |>
  select(r.squared, adj.r.squared, sigma)
```

```
## # A tibble: 1 x 3
##   r.squared adj.r.squared sigma
##       <dbl>         <dbl> <dbl>
## 1     0.450         0.418  16.9
```

```r
glance(mult.fit) |>
  select(r.squared, adj.r.squared, sigma)
```

```
## # A tibble: 1 x 3
##   r.squared adj.r.squared sigma
##       <dbl>         <dbl> <dbl>
## 1     0.468         0.397  16.7
```

Predicted values from R

We could plug in values into the equation, but R can do this for us. The `{modelr}` package gives some functions that allow us to predictions in a tidy way:

Let's use `add_predictions()` to predict the 2022 results

```
library(modelr)

midterms |>
  filter(year == 2022) |>
  add_predictions(mult.fit)
```

```
## # A tibble: 1 x 7
##    year president party approval seat_change rdi_cha~1  pred
##   <dbl> <chr>     <chr>    <dbl>       <dbl>     <dbl> <dbl>
## 1  2022 Biden     D           42          NA    -0.003 -53.2
## # ... with abbreviated variable name 1: rdi_change
```

Predictions from several models

The `gather_predictions()` will return one row for each model passed to it with the prediction for that model:

```
midterms |>
  filter(year == 2022) |>
  gather_predictions(fit.app, mult.fit)
```

```
## # A tibble: 2 x 8
##   model     year presi~1 party appro~2 seat_~3 rdi_c~4  pred
##   <chr>    <dbl> <chr>   <chr>   <dbl>   <dbl>   <dbl> <dbl>
## 1 fit.app   2022 Biden   D          42      NA  -0.003 -36.9
## 2 mult.fit  2022 Biden   D          42      NA  -0.003 -53.2
## # ... with abbreviated variable names 1: president,
## #   2: approval, 3: seat_change, 4: rdi_change
```

Predictions from new data

What about predicted values not in data?

```
tibble(approval = c(50, 75), rdi_change = 0) |>
  gather_predictions(fit.app, mult.fit)
```

```
## # A tibble: 4 x 4
##   model    approval rdi_change   pred
##   <chr>       <dbl>      <dbl>  <dbl>
## 1 fit.app        50          0 -25.6
## 2 fit.app        75          0   9.92
## 3 mult.fit       50          0 -40.9
## 4 mult.fit       75          0  -2.79
```

Predictions from augment()

We can also get predicted values from the augment() function using the newdata argument:

```r
newdata <- tibble(approval = c(50, 75), rdi_change = 0)

augment(mult.fit, newdata = newdata)
```

```
## # A tibble: 2 x 3
##   approval rdi_change .fitted
##      <dbl>      <dbl>   <dbl>
## 1       50          0  -40.9
## 2       75          0   -2.79
```

2/ Categorical independent variables

Political effects of gov't programs

- *Progesa*: Mexican conditional cash transfer program (CCT) from ~2000
    - Welfare $$ given if kids enrolled in schools, get regular check-ups, etc.
- Do these programs have political effects?
    - Program had support from most parties.
    - Was implemented in a nonpartisan fashion.
    - Would the incumbent presidential party be rewarded?

The data

- Randomized roll-out of the CCT program:
    - treatment: receive CCT 21 months before 2000 election
    - control: receive CCT 6 months before 2000 election
- Does having CCT longer mobilize voters for incumbent PRI party?

Name	Description
`treatment`	early Progresa (1) or late Progresa (0)
`pri2000s`	PRI votes in the 2000 election as a share of adults in precinct
`t2000`	turnout in the 2000 election as share of adults in precinct

```r
library(qss)
data("progresa", package = "qss")
cct <- as_tibble(progresa) |>
  select(treatment, pri2000s, t2000)
cct
```

```
## # A tibble: 417 x 3
##    treatment pri2000s t2000
##        <int>    <dbl> <dbl>
##  1         1     40.8  55.8
##  2         1     22.4  31.2
##  3         1     38.9  47.0
##  4         1     31.2  45.0
##  5         0     76.9 100
##  6         0     23.9  37.4
##  7         1     47.3  64.9
##  8         1     21.4  58.1
##  9         1     56.5  71.3
## 10         1     36.6  51.2
## # ... with 407 more rows
```

Difference in means estimates

Does CCT affect turnout?

```r
cct |> group_by(treatment) |>
  summarize(t2000 = mean(t2000)) |>
  pivot_wider(names_from = treatment, values_from = t2000) |>
  mutate(ATE = `1` - `0`)
```

```
## # A tibble: 1 x 3
##     `0`   `1`   ATE
##   <dbl> <dbl> <dbl>
## 1  63.8  68.1  4.27
```

Does CCT affect PRI (incumbent) votes?

```r
cct |> group_by(treatment) |>
  summarize(pri2000s = mean(pri2000s)) |>
  pivot_wider(names_from = treatment, values_from = pri2000s) |>
  mutate(ATE = `1` - `0`)
```

```
## # A tibble: 1 x 3
##     `0`   `1`   ATE
##   <dbl> <dbl> <dbl>
## 1  34.5  38.1  3.62
```

Binary independent variables

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

- When independent variable X_i is **binary**:
  - Intercept $\hat{\alpha}$ is the average outcome in the $X = 0$ group.
  - Slope $\hat{\beta}$ is the difference-in-means of Y between $X = 1$ group and $X = 0$ group.

$$\hat{\beta} = \overline{Y}_{\text{treated}} - \overline{Y}_{\text{control}}$$

- If there are other independent variables, this becomes the difference-in-means controlling for those covariates.

Linear regression for experiments

- Under **randomization**, we can estimate the ATE with regression:

```
cct |> group_by(treatment) |>
  summarize(pri2000s = mean(pri2000s)) |>
  pivot_wider(names_from = treatment, values_from = pri2000s) |>
  mutate(ATE = `1` - `0`)
```

```
## # A tibble: 1 x 3
##     `0`   `1`   ATE
##   <dbl> <dbl> <dbl>
## 1  34.5  38.1  3.62
```

```
lm(pri2000s ~ treatment, data = cct) |> coef()
```

```
## (Intercept)   treatment
##       34.49        3.62
```

Categorical variables in regression

- We often have **categorical variables**:
  - Race/ethnicity: white, Black, Latino, Asian.
  - Partisanship: Democrat, Republican, Independent
- Strategy for including in a regression: create a **series of binary variables**

Unit	Party	Democrat	Republican	Independent
1	Democrat	1	0	0
2	Democrat	1	0	0
3	Independent	0	0	1
4	Republican	0	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Then include **all but one** of these binary variables:

$$\text{turnout}_i = \alpha + \beta_1 \text{Republican}_i + \beta_2 \text{Independent}_i + \varepsilon_i$$

Interpreting categorical variables

$$\text{turnout}_i = \alpha + \beta_1 \text{Republican}_i + \beta_2 \text{Independent}_i + \varepsilon_i$$

- $\hat{\alpha}$: average outcome in the **omitted group/baseline** (Democrats).
- $\hat{\beta}$ coefficients: average difference between each group and the baseline.
    - $\hat{\beta}_1$: average difference in turnout between Republicans and Democrats
    - $\hat{\beta}_2$: average difference in turnout between Independents and Democrats

CCES data

```r
library(gov50data)
cces_2020
```

```
## # A tibble: 51,551 x 6
##    gender race  educ                 pid3    turno~1 pres_~2
##    <fct>  <fct> <fct>                <fct>     <dbl> <fct>
##  1 Male   White 2-year               Republ~       1 Donald~
##  2 Female White Post-grad            Democr~      NA <NA>
##  3 Female White 4-year               Indepe~       1 Joe Bi~
##  4 Female White 4-year               Democr~       1 Joe Bi~
##  5 Male   White 4-year               Indepe~       1 Other
##  6 Male   White Some college         Republ~       1 Donald~
##  7 Male   Black Some college         Not su~      NA <NA>
##  8 Female White Some college         Indepe~       1 Donald~
##  9 Female White High school graduate Republ~       1 Donald~
## 10 Female White 4-year               Democr~       1 Joe Bi~
## # ... with 51,541 more rows, and abbreviated variable names
## #   1: turnout_self, 2: pres_vote
```

Categorical variables in the CCES data

```r
turnout_pred <- lm(turnout_self ~ pid3, data = cces_2020)
turnout_pred
```

```
##
## Call:
## lm(formula = turnout_self ~ pid3, data = cces_2020)
##
## Coefficients:
##     (Intercept)   pid3Republican  pid3Independent
##          0.9635          -0.0103          -0.0394
##       pid3Other    pid3Not sure
##         -0.0066          -0.3331
```

What R does internally with factor variables in `lm`

```r
cces_2020 |> drop_na(turnout_self, pid3) |> select(pid3) |> pull() |>
  head()
```

```
## [1] Republican  Independent Democrat    Independent
## [5] Republican  Independent
## 7 Levels: Democrat Republican Independent ... not asked
```

```r
model.matrix(turnout_pred) |>
  head()
```

```
##   (Intercept) pid3Republican pid3Independent pid3Other
## 1           1              1               0         0
## 3           1              0               1         0
## 4           1              0               0         0
## 5           1              0               1         0
## 6           1              1               0         0
## 8           1              0               1         0
##   pid3Not sure
## 1            0
## 3            0
## 4            0
## 5            0
## 6            0
```

Gov 50: 16. Sampling

Matthew Blackwell

Harvard University

Roadmap

1. Sampling exercise
2. Sampling framework
3. Polls

1/ Sampling exercise

Data on class years enrolled in Gov 50

```r
library(gov50data)
class_years
```

```
## # A tibble: 122 x 1
##    year
##    <chr>
##  1 Senior
##  2 Junior
##  3 Sophomore
##  4 Junior
##  5 Graduate Year 2
##  6 Sophomore
##  7 Professional Year 2
##  8 First-Year
##  9 Sophomore
## 10 Junior
## # ... with 112 more rows
```

What proportion of the class is first years?

```r
class_years |>
  count(year) |>
  mutate(prop = n / nrow(class_years))
```

```
## # A tibble: 9 x 3
##   year                      n    prop
##   <chr>                 <int>   <dbl>
## 1 First-Year               25 0.205
## 2 Graduate Year 1           2 0.0164
## 3 Graduate Year 2           1 0.00820
## 4 Junior                   31 0.254
## 5 Not Set                   3 0.0246
## 6 Professional Year 2       2 0.0164
## 7 Senior                   14 0.115
## 8 Sophomore                43 0.352
## 9 Year 1, Semester 1        1 0.00820
```

Let’s take some samples!

We can use the `slice_sample()` function to take a random sample of rows of a tibble:

```
class_years |>
  slice_sample(n = 5)
```

```
## # A tibble: 5 x 1
##   year
##   <chr>
## 1 Sophomore
## 2 Junior
## 3 Junior
## 4 Sophomore
## 5 Sophomore
```

Another sample

```r
class_years |>
  slice_sample(n = 5)
```

```
## # A tibble: 5 x 1
##   year
##   <chr>
## 1 Junior
## 2 Not Set
## 3 First-Year
## 4 First-Year
## 5 Sophomore
```

Sample proportion of first-years

```r
class_years |>
  slice_sample(n = 20) |>
  summarize(fy_prop = mean(year == "First-Year"))
```

```
## # A tibble: 1 x 1
##   fy_prop
##     <dbl>
## 1    0.15
```

Repeated sampling

We sometimes want to draw multiple samples from a tibble. For this we can use `rep_slice_sample()` from the `infer` package:

```r
library(infer)
class_years |>
  rep_slice_sample(n = 5, reps = 2)
```

```
## # A tibble: 10 x 2
## # Groups:   replicate [2]
##    replicate year
##        <int> <chr>
##  1         1 First-Year
##  2         1 Sophomore
##  3         1 First-Year
##  4         1 Sophomore
##  5         1 First-Year
##  6         2 Junior
##  7         2 First-Year
##  8         2 Sophomore
##  9         2 First-Year
## 10         2 Sophomore
```

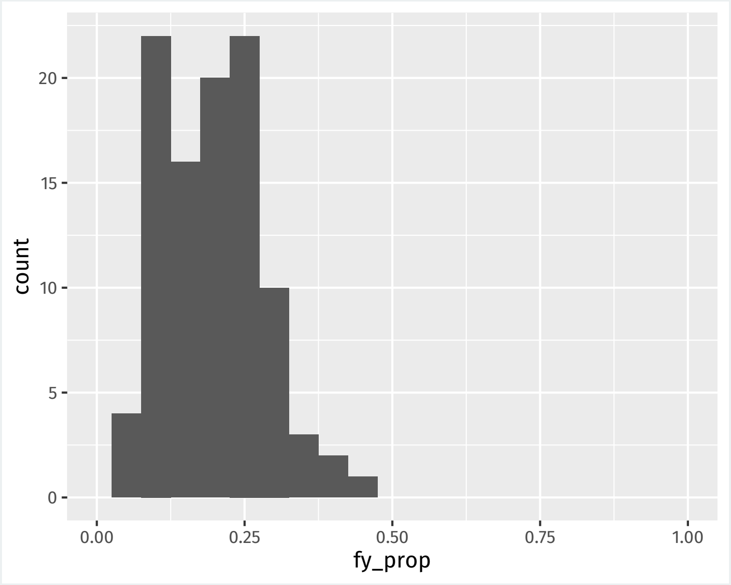
Simulate many separate studies being done

```r
samples_n20 <- class_years |>
  rep_slice_sample(n = 20, reps = 100) |>
  group_by(replicate) |>
  summarize(fy_prop = mean(year == "First-Year"))
samples_n20
```

```
## # A tibble: 100 x 2
##    replicate fy_prop
##        <int>   <dbl>
##  1         1    0.25
##  2         2    0.4
##  3         3    0.3
##  4         4    0.4
##  5         5    0.2
##  6         6    0.25
##  7         7    0.1
##  8         8    0.25
##  9         9    0.35
## 10        10    0.1
## # ... with 90 more rows
```

Distribution of these proportions

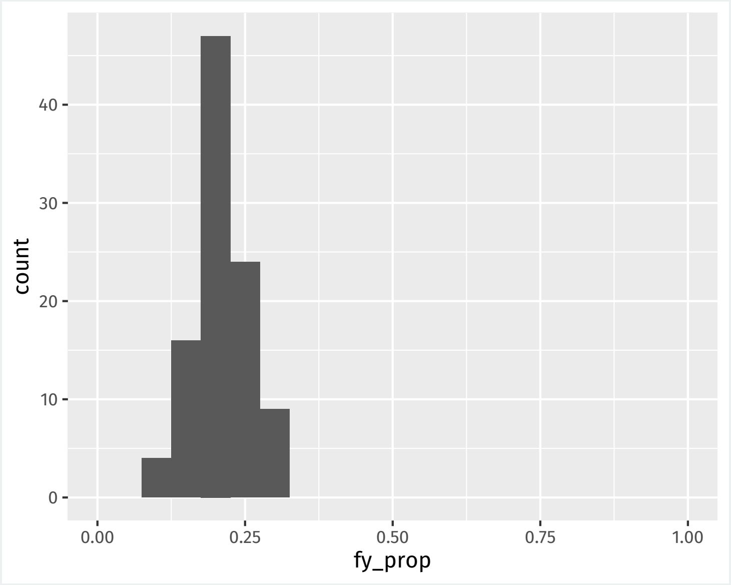
```r
samples_n20 |>
  ggplot(mapping = aes(x = fy_prop)) +
  geom_histogram(binwidth=0.05) +
  lims(x = c(0, 1))
```



count
20
15
10
5
0
0.00
0.25
0.50
0.75
1.00
fy_prop

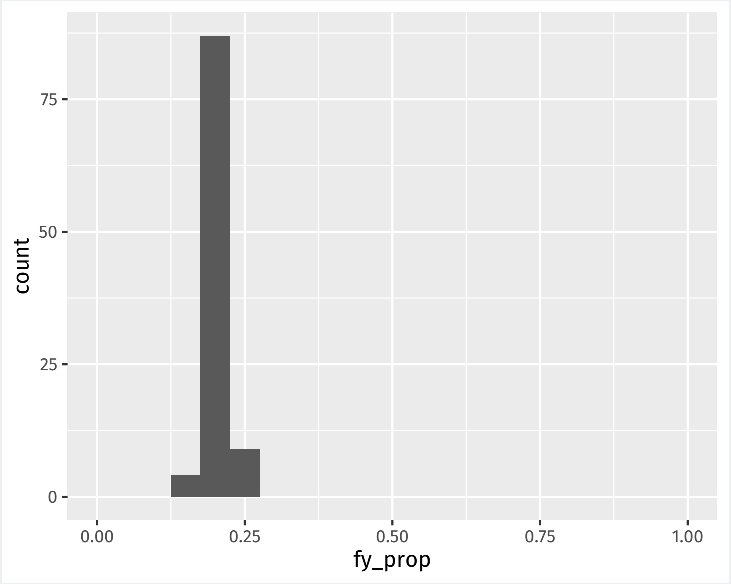
What if the sample sizes are bigger?

```r
samples_n50 <- class_years |>
  rep_slice_sample(n = 50, reps = 100) |>
  group_by(replicate) |>
  summarize(fy_prop = mean(year == "First-Year"))

samples_n50 |>
  ggplot(mapping = aes(x = fy_prop)) +
  geom_histogram(binwidth=0.05)  +
  lims(x = c(0, 1))
```


count
fy_prop
0
10
20
30
40
0.00
0.25
0.50
0.75
1.00

What if the sample sizes are bigger?

```
samples_n100 <- class_years |>
  rep_slice_sample(n = 100, reps = 100) |>
  group_by(replicate) |>
  summarize(fy_prop = mean(year == "First-Year"))

samples_n100 |>
  ggplot(mapping = aes(x = fy_prop)) +
  geom_histogram(binwidth=0.05)  +
  lims(x = c(0, 1))
```


count
75
50
25
0
0.00
0.25
0.50
0.75
1.00
fy_prop

Sample size and variability across samples

```r
samples_n20 |>
  summarize(sd(fy_prop)) |> pull()
```

```
## [1] 0.0849
```

```r
samples_n50 |>
  summarize(prop_sd = sd(fy_prop)) |> pull()
```

```
## [1] 0.0427
```

```r
samples_n100 |>
  summarize(prop_sd = sd(fy_prop)) |> pull()
```

```
## [1] 0.0147
```

2/ Sampling framework

Populations

Population: group of units/people we want to learn about.

Population parameter: some numerical summary of the population we would like to know. - population mean/proportion, population standard deviation.

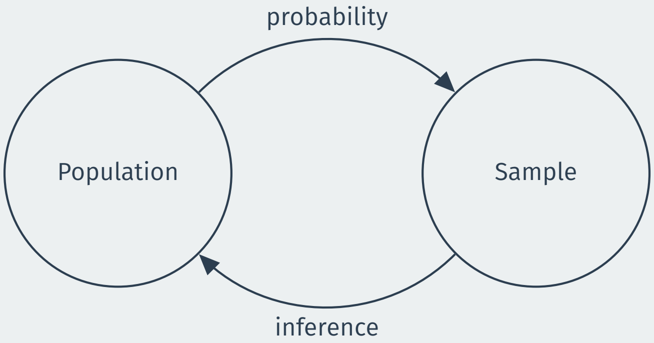
Census: complete recording of data on the entire population.

Samples

Sample: subset of the population taken in some way (hopefully randomly).

Estimator or sample statistic: numerical summary of the sample that is our “best guess” for the unknown population parameter.

Sampling framework



Sampling at random

Random sample: units selected into sample from population with a non-zero probability.

Simple random sample: all units have the same probability of being selected into the sample.

Our sampling exercise

- **Population**: all students enrolled in Gov 50.
- **Population parameter**: population proportion of first-years enrolled in Gov 50
    - Population proportions often denoted p
- **Sample**: simple random sample of different sizes.
- **Sample statistic/estimator**: sample proportion of first-years
    - Estimators often denoted with a hat: $\hat{p}$
    - We saw the $\hat{p}$ varies with the random sample taken.

Expected value

The **expected value** of a sample statistic, $\mathbb{E}[\hat{p}]$, is the average value of the statistic across repeated samples.

```r
samples_n100 |>
  summarize(mean(fy_prop)) |> pull()
```

```
## [1] 0.205
```

The **expected value** of a sample proportion from a simple random sample is equal to the population proportion, $\mathbb{E}[\hat{p}] = p$

Standard error

The **standard error** is the standard deviation of the sample statistic across repeated samples.

```
samples_n100 |>
  summarize(sd(fy_prop)) |> pull()
```

```
## [1] 0.0147
```

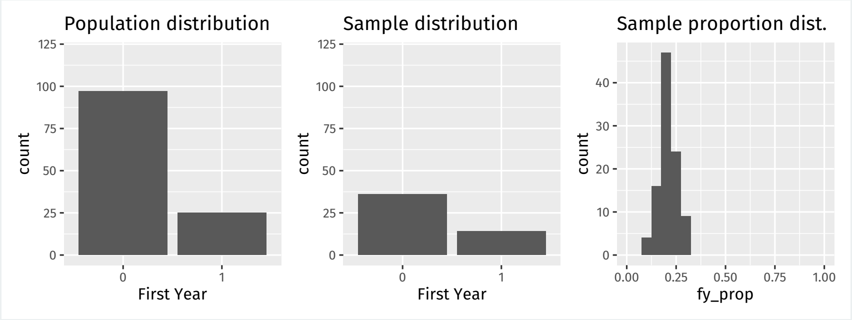
Tells us how far away, on average, the sample proportion will be from the population proportion.

Standard error vs population standard deviation

The **standard error** is the SD of the statistic across repeated samples.

Should not be confused with the population standard deviation or sample standard deviation, both of which measure how far **units** are away from a mean.

The three distributions



3/ Polls

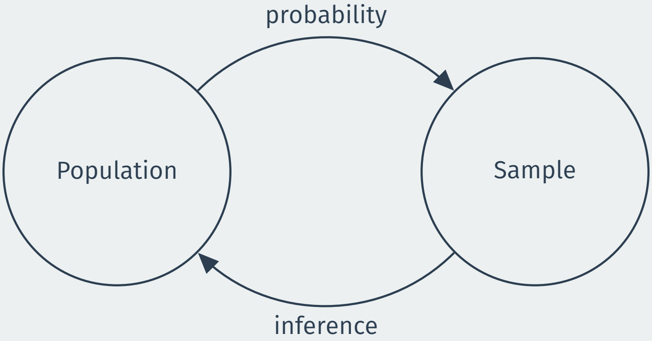
How popular is Joe Biden?

- What proportion of the public approves of Biden’s job as president?
- Latest Gallup poll:
    - Sept 1st-16th
    - 812 adult Americans
    - Telephone interviews
    - Approve (42%), Disapprove (56%)

Poll in our framework

- **Population**: adults 18+ living in 50 US states and DC.
- **Population parameter**: population proportion of all US adults that approve of Biden.
    - Census: not possible.
- **Sample:** random digit dialing phone numbers (cell and landline).
- **Point estimate**: sample proportion that approve of Biden

Where are we going?



We only get 1 sample. Can we learn about the population from that sample?

Gov 50: 17. Sampling Distributions

Matthew Blackwell

Harvard University

Roadmap

1. Poll example
2. Random variables and probability distributions
3. Sampling distribution
4. Normal variables and the Central Limit Theorem

1/ Poll example

How popular is Joe Biden?

- What proportion of the public approves of Biden's job as president?
- Latest Gallup poll:
  - Sept 1st-16th
  - 812 adult Americans
  - Telephone interviews
  - Approve (42%), Disapprove (56%)

Poll in our framework

- **Population**: adults 18+ living in 50 US states and DC.
- **Population parameter**: population proportion of all US adults that approve of Biden.
    - Census: not possible.
- **Sample:** random digit dialing phone numbers (cell and landline).
- **Point estimate**: sample proportion that approve of Biden

2/ Random variables and probability distributions

Random variables

Random variables are numerical summaries of chance processes:

$$X_i = \begin{cases} 1 & \text{if respondent } i \text{ supports Biden,} \\ 0 & \text{otherwise} \end{cases}$$

With a simple random sample, chance of $X_i = 1$ is equal to the population proportion of people that support Biden.

Types of random variables

- **Discrete**: X can take a finite (or countably infinite) number of values.
    - Number of heads in 5 coin flips
    - Sampled senator is a woman ($X = 1$) or not ($X = 0$)
    - Number of battle deaths in a civil war
- **Continuous**: X can take any real value (usually within an interval).
    - GDP per capita (average income) in a country.
    - Share of population that approves of Biden.
    - Amount of time spent on a website.

Probability distributions

Probability distributions tell us the chances of different values of a r.v. occurring

Discrete variables: like a frequency barplot for the population distribution.

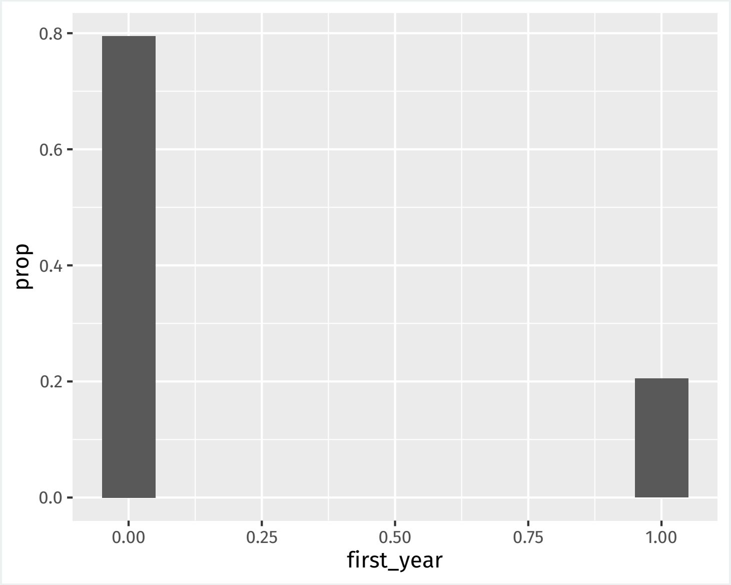
Continuous variables: like a continuous version of population histogram.

Discrete probability distribution

We can use the `y = ..prop..` aesthetic to get a barplot with proportions instead of count to show us the chance/probability of selecting a first-year student:

```
library(gov50data)
class_years |>
  mutate(first_year = as.numeric(year == "First-Year")) |>
  ggplot(aes(x = first_year)) +
  geom_bar(mapping = aes(y = ..prop..), width = 0.1)
```

Discrete probability distribution



Midwest data

```r
library(ggplot2)
midwest
```

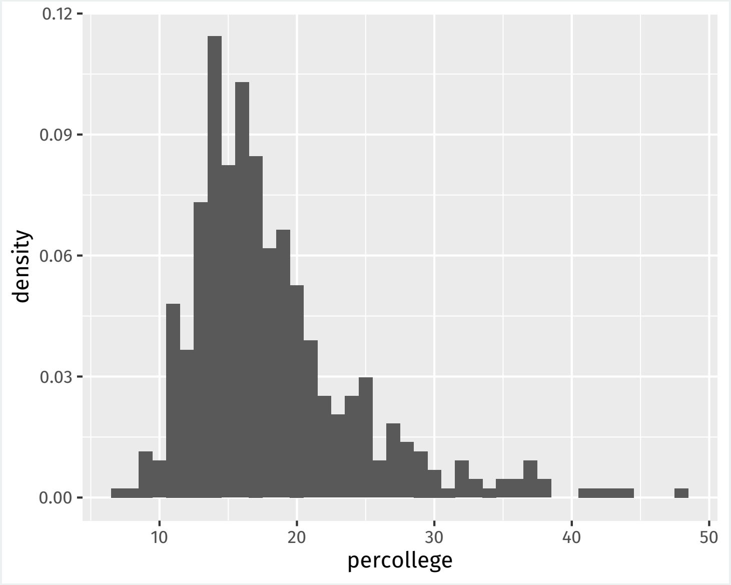
```
## # A tibble: 437 x 28
##       PID county  state  area popto~1 popde~2 popwh~3 popbl~4
##     <int> <chr>   <chr> <dbl>   <int>   <dbl>   <int>   <int>
##  1    561 ADAMS   IL    0.052   66090   1271.   63917    1702
##  2    562 ALEXAN~ IL    0.014   10626    759     7054    3496
##  3    563 BOND    IL    0.022   14991    681.   14477     429
##  4    564 BOONE   IL    0.017   30806   1812.   29344     127
##  5    565 BROWN   IL    0.018    5836    324.    5264     547
##  6    566 BUREAU  IL    0.05    35688    714.   35157      50
##  7    567 CALHOUN IL    0.017    5322    313.    5298       1
##  8    568 CARROLL IL    0.027   16805    622.   16519     111
##  9    569 CASS    IL    0.024   13437    560.   13384      16
## 10    570 CHAMPA~ IL    0.058  173025   2983.  146506   16559
## # ... with 427 more rows, 20 more variables:
## #   popamerindian <int>, popasian <int>, popother <int>,
## #   percwhite <dbl>, percblack <dbl>, percamerindan <dbl>,
## #   percasian <dbl>, percother <dbl>, popadults <int>,
## #   perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>,
```

Continuous probability distribution

We can use the `y = ..density..` to create a **density histogram** instead of a count histogram so that the area of the histogram boxes are equal to the chance of randomly selecting a unit in that bin:

```r
midwest |>
  ggplot(aes(x = percollege)) +
  geom_histogram(aes(y = ..density..), binwidth = 1)
```

Continuous probability distribution



Why density?

Histograms with **density** on the y-axis are drawn so that the area of each box is equal to the proportion of units in the sample in that horizontal bin.

Easier to compare distributions across sample sizes.

Sum up all the area = 1 (but heights can go above 1)

3/ Sampling distribution

Key properties of sums and means

Suppose $X_1, X_2, \ldots, X_n$ is a simple random sample from a population distribution with mean μ ("mu") and variance σ^2 ("sigma squared")

Sample mean: $\overline{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i$

$$\overline{X}_n = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

. . .

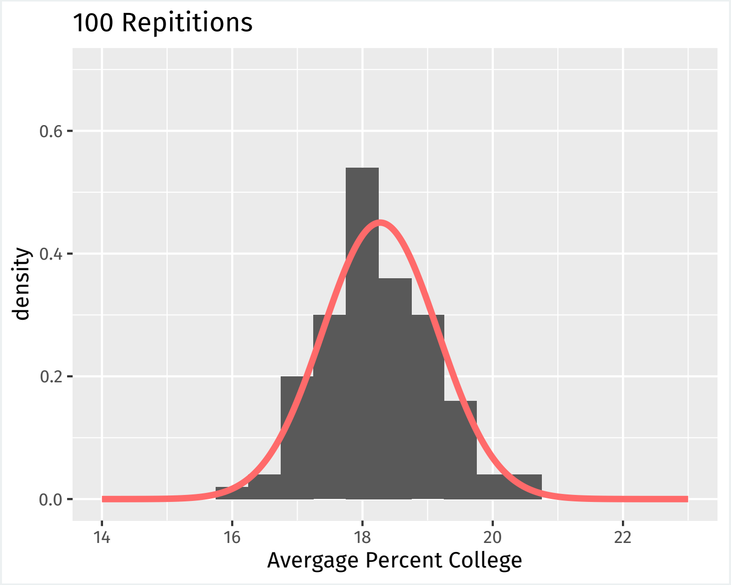
$\overline{X}_n$ is a random variable with a distribution!!

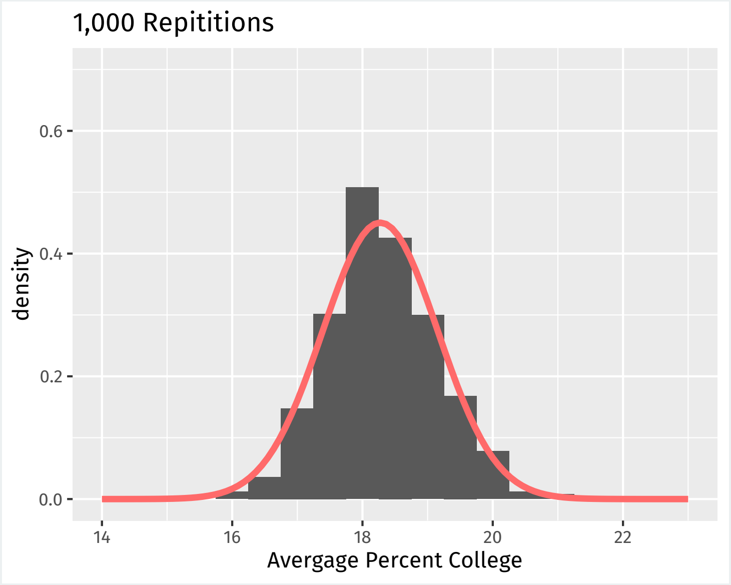
Sample means/proportions distribution

Sampling distributions are the probability distributions of an estimator like $\overline{X}_n$

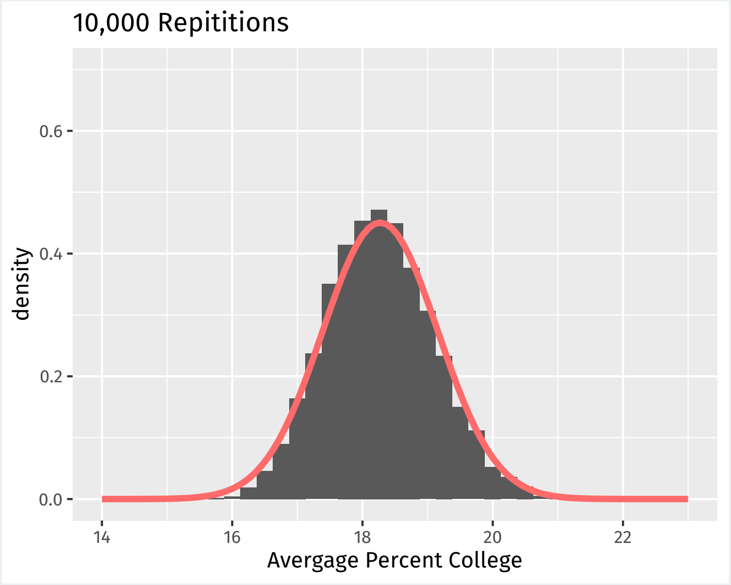
When we have access to the full population, we can approximate the sampling distribution with repeated sampling.

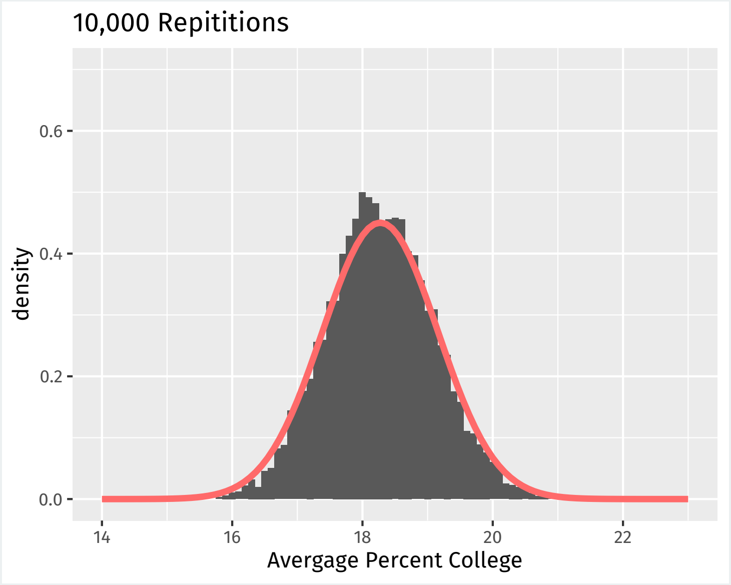
```
library(infer)
midwest |>
  rep_slice_sample(n = 50, reps = 100) |>
  group_by(replicate) |>
  summarize(`Avergage Percent College` = mean(percollege)) |>
  ggplot(aes(x = `Avergage Percent College`)) +
  geom_histogram(mapping = aes(y = ..density..), binwidth = 0.5) +
  coord_cartesian(xlim = c(14, 23), ylim = c(0, 0.7)) +
  labs(title = "100 Repititions") +
  stat_function(fun = dnorm, args = c(mean(midwest$percollege), sd(midwest$
                color = "indianred1", size = 1.5, xlim = c(14, 23))
```


100 Repititions
density
0.6
0.4
0.2
0.0
14
16
18
20
22
Avergage Percent College



1,000 Repititions
density
0.6
0.4
0.2
0.0
14
16
18
20
22
Avergage Percent College


10,000 Repititions
density
0.6
0.4
0.2
0.0
14
16
18
20
22
Avergage Percent College


10,000 Repititions
density
0.6
0.4
0.2
0.0
14
16
18
20
22
Avergage Percent College

Sampling distribution of the sample mean

Suppose $X_1, X_2, \dots, X_n$ is a simple random sample from a population distribution with mean μ and variance σ^2.

Expected value of the distribution of $\overline{X}_n$ is the population mean, μ.

Standard error of the distribution of $\overline{X}_n$ is approximately $\sigma/\sqrt{n}$:

$$SE \approx \frac{\text{population standard deviation}}{\sqrt{\text{sample size}}}$$

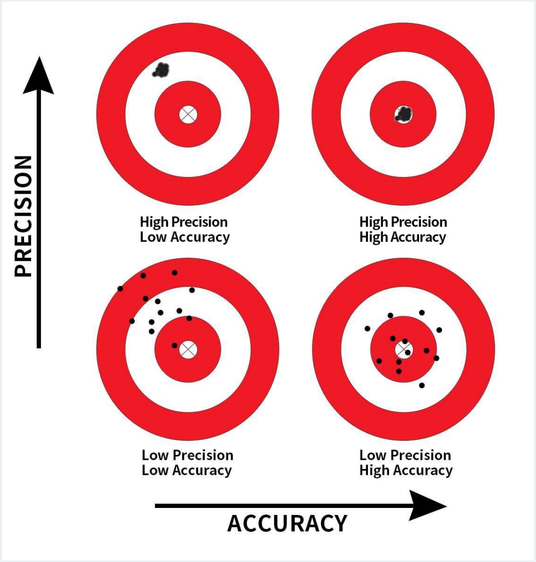
Unbiasedness

An estimator is **unbiased** when its expected value across repeated samples equals the population parameter of interest.

Sample mean of a simple random sample is **unbiased** for the population mean, $\mathbb{E}[\overline{X}_n] = \mu$

An estimator that isn't unbiased is called **biased**.

Precision vs accuracy



Law of large numbers

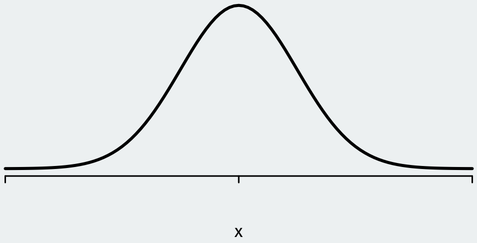
Law of large numbers

Let $X_1, \dots, X_n$ be a simple random sample from a population with mean μ and finite variance σ^2. Then, $\overline{X}_n$ converges to μ as n gets large.

- Probability of $\overline{X}_n$ being “far away” from μ goes to 0 as n gets big.
- The distribution of sample mean “collapses” to population mean.
- Can see this from the SE of $\overline{X}_n$: $SE = \sigma/\sqrt{n}$.
- Not necessarily true with a biased sample!

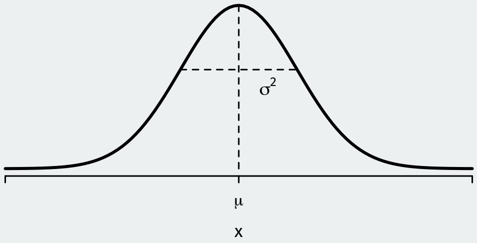
4/ Normal variables and the Central Limit Theorem

Normal random variable



- A **normal distribution** has a PDF that is the classic “bell-shaped” curve.
    - Extremely ubiquitous in statistics.
    - An r.v. is more likely to be in the center, rather than the tails.
- Three key properties of this PDF:
    - **Unimodal**: one peak at the mean.
    - **Symmetric** around the mean.
    - **Everywhere positive**: any real value can possibly occur.

Normal distribution



- A normal distribution can be affect by two values:
    - **mean/expected value** usually written as μ
    - **variance** written as σ^2 (standard deviation is σ)
    - Written $X \sim N(\mu, \sigma^2)$.
- **Standard normal distribution**: mean 0 and standard deviation 1.

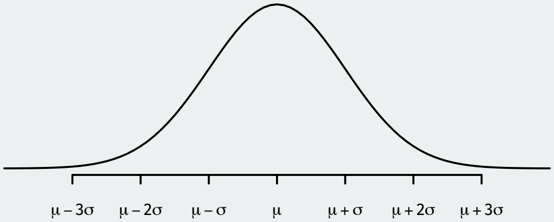
Central limit theorem

Central limit theorem

Let $X_1, \dots, X_n$ be a simple random sample from a population with mean μ and finite variance σ^2. Then, $\overline{X}_n$ will be approximately distributed $N(\mu, \sigma^2/n)$ in large samples.

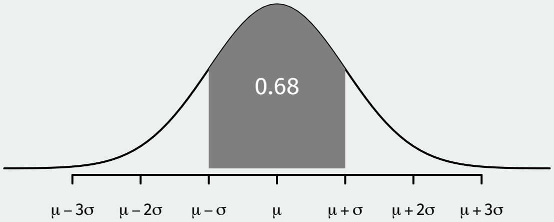
- "Sample means tend to be normally distributed as samples get large."
- $\rightsquigarrow$ we know (an approx. of) the entire probability distribution of $\overline{X}_n$
  - Approximation is better as n goes up.
  - Does not depend on the distribution of X_i!

Empirical Rule for the Normal Distribution



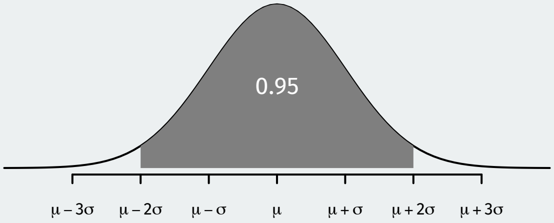
- If $X \sim N(\mu, \sigma^2)$, then:

Empirical Rule for the Normal Distribution



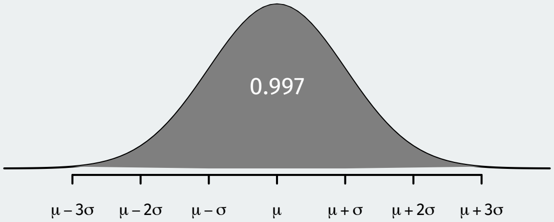
- If $X \sim N(\mu, \sigma^2)$, then:
  - $\approx$ 68% of the distribution of X is within 1 SD of the mean.

Empirical Rule for the Normal Distribution



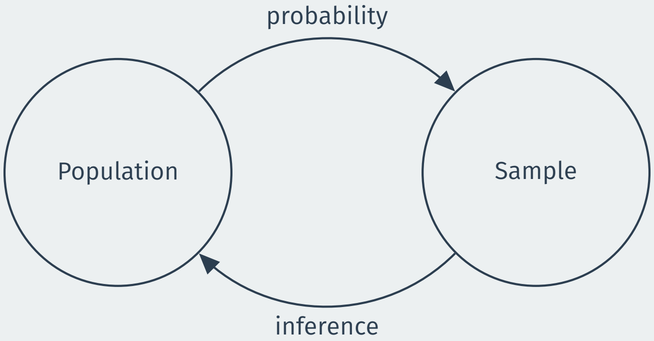
- If $X \sim N(\mu, \sigma^2)$, then:
    - $\approx$ 68% of the distribution of X is within 1 SD of the mean.
    - $\approx$ 95% of the distribution of X is within 2 SDs of the mean.

Empirical Rule for the Normal Distribution



- If $X \sim N(\mu, \sigma^2)$, then:
    - $\approx$ 68% of the distribution of X is within 1 SD of the mean.
    - $\approx$ 95% of the distribution of X is within 2 SDs of the mean.
    - $\approx$ 99.7% of the distribution of X is within 3 SDs of the mean.
- CLT + empirical rule: we'll know the rough distribution of estimation errors we should expect.

Where are we going?



We only get 1 sample. Can we learn about the population from that sample?

Gov 50: 18. The Bootstrap

Matthew Blackwell

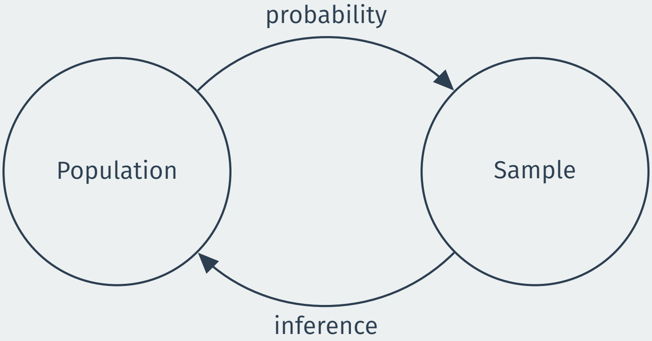
Harvard University

Roadmap

1. Resampling from our sample
2. Confidence intervals
3. Calculating confidence intervals

1/ Resampling from our sample

Where are we?



Can we approximate the **sampling distribution** with our single sample?

American National Election Survey data

Name	Description
`state`	State of respondent
`district`	Congressional district of respondent
`pid7`	Party ID (1=Strong D, 7=Strong R)
`pres_vote`	Self reported vote in 2020
`sci_therm`	0-100 therm score for scientists
`rural_therm`	0-100 therm score for rurual Americans
`favor_voter_id`	1 if respondent thinks voter ID should be required
`envir_doing_more`	1 if respondent thinks gov't should be doing more about climate change

ANES data

```r
library(gov50data)
anes
```

```
## # A tibble: 5,162 x 8
##    state district  pid7 pres_vote sci_therm rural_~1 favor~2
##    <chr>    <dbl> <dbl> <chr>         <dbl>    <dbl>   <dbl>
##  1 ID           2     4 Other            70       60       1
##  2 VA           2     3 Biden           100       75       0
##  3 CO           4     4 Trump            60       90       1
##  4 TX           5     3 Biden            85       85       1
##  5 WI           6     6 Trump            85       70       1
##  6 CA          40     2 Biden            50       50       1
##  7 WI           5     2 Biden           100       70       1
##  8 OR           4     7 Trump            70       50       0
##  9 MA           5     3 Biden            80       70       0
## 10 NV           3     1 Biden            85       40       0
## # ... with 5,152 more rows, 1 more variable:
## #   envir_doing_more <dbl>, and abbreviated variable names
## #   1: rural_therm, 2: favor_voter_id
```

Sample statistic

What is the average thermemeter score for scientists?

```r
anes |>
  summarize(mean(sci_therm))
```

```
## # A tibble: 1 x 1
##   `mean(sci_therm)`
##               <dbl>
## 1              80.6
```

What is the sampling distribution of this average? We only have this 1 draw!

Notation review

Population: all US adults.

Population parameter: average feeling thermometer score for scientists among all US adults.

Sample: (complicated) random sample of all US adults.

Sample statistic/point estimate: sample average of thermometer scores.

Roughly how far our point estimate is likely to be from the truth?

The bootstrap

Mimic sampling from the population by **resampling** many times from the sample itself.

Bootstrap resampling done **with replacement** (same row can appear more than once)

One bootstrap resample

```r
boot_1 <- anes |>
  slice_sample(prop = 1, replace = TRUE)
boot_1
```

```
## # A tibble: 5,162 x 8
##    state district  pid7 pres_vote sci_therm rural_~1 favor~2
##    <chr>    <dbl> <dbl> <chr>         <dbl>    <dbl>   <dbl>
##  1 CO           6     1 Biden            85       70       0
##  2 NY           8     1 Biden            85       70       1
##  3 SC           7     1 Biden           100      100       0
##  4 CO           3     4 Trump            85       85       1
##  5 CA          39     2 Biden           100       60       0
##  6 CA          37     3 Biden            90       65       0
##  7 AR           2     1 Biden            85       70       0
##  8 CO           6     1 Biden            90       70       0
##  9 WA           5     3 Biden            70       85       0
## 10 MI           7     3 Other            60       70       0
## # ... with 5,152 more rows, 1 more variable:
## #   envir_doing_more <dbl>, and abbreviated variable names
## #   1: rural_therm, 2: favor_voter_id
```

Sample mean in the bootstrap sample

```r
boot_1 |>
  summarize(mean(sci_therm))
```

```
## # A tibble: 1 x 1
##   `mean(sci_therm)`
##               <dbl>
## 1              81.0
```

Many bootstrap samples

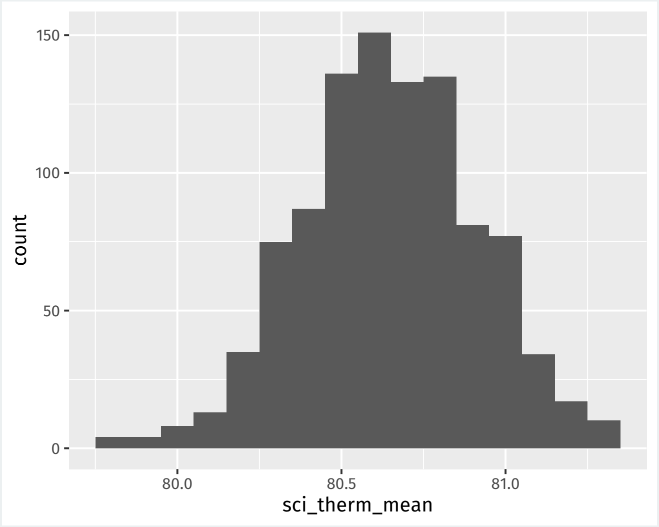
```r
library(infer)
bootstrap_dist <- anes |>
  rep_slice_sample(prop = 1, reps = 1000, replace = TRUE) |>
  group_by(replicate) |>
  summarize(sci_therm_mean = mean(sci_therm))
bootstrap_dist
```

Many bootstrap samples

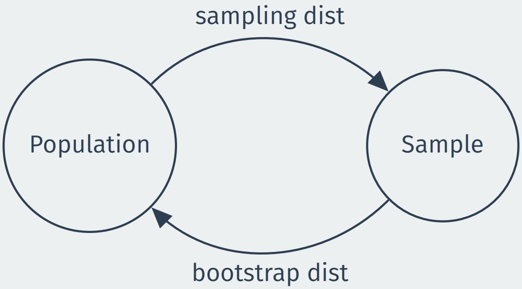
```
## # A tibble: 1,000 x 2
##    replicate sci_therm_mean
##        <int>          <dbl>
##  1         1           80.6
##  2         2           80.2
##  3         3           80.1
##  4         4           80.8
##  5         5           80.7
##  6         6           80.3
##  7         7           80.5
##  8         8           81.1
##  9         9           80.9
## 10        10           80.8
## # ... with 990 more rows
```

Visualizing the bootstrap distribution

```r
bootstrap_dist |>
  ggplot(aes(x = sci_therm_mean)) + geom_histogram(binwidth = 0.1)
```



Bootstrap distribution



Bootstrap distribution **approximates** the sampling distribution of the estimator.

Both should have a **similar shape and spread** if sampling from the distribution ≈ bootstrap resampling.

Approximation gets better as sample gets bigger.

Comparing to the point estimate

Given the sampling, not surprising that bootstrap distribution is centered on the point estimate:

```
bootstrap_dist |>
  summarize(mean(sci_therm_mean))
```

```
## # A tibble: 1 x 1
##   `mean(sci_therm_mean)`
##                    <dbl>
## 1                   80.6
```

```
anes |>
  summarize(mean(sci_therm))
```

```
## # A tibble: 1 x 1
##   `mean(sci_therm)`
##               <dbl>
## 1              80.6
```

2/ Confidence intervals

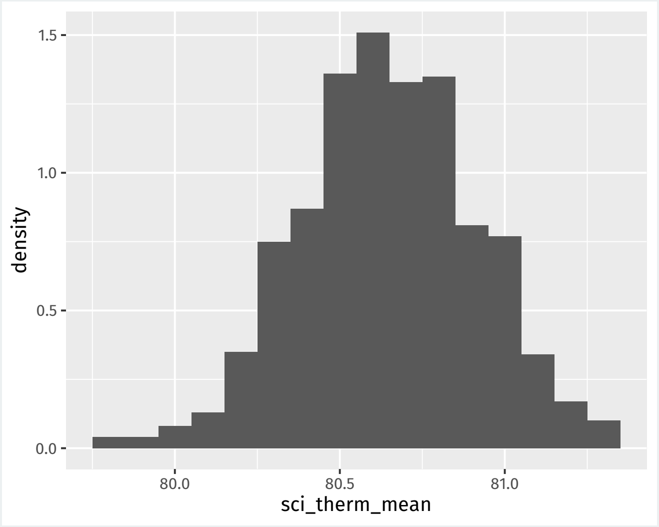
What is a confidence interval?

Point estimate: best single guess about the population parameter. Unlikely to be exactly correct.

Confidence interval: a range of plausible values of the population parameter.

Where is most of the bootstrap distribution?

```r
bootstrap_dist |>
  ggplot(aes(x = sci_therm_mean)) +
  geom_histogram(aes(y= ..density..), binwidth = 0.1)
```



Confidence intervals

- Each sample gives a different CI or toss of the ring.
- Some samples the ring will contain the target (the CI will contain the truth) other times it won't.
  - We don't know if the CI for our sample contains the truth!
- **Confidence level:** percent of the time our CI will contain the population parameter.
  - Number of ring tosses that will hit the target.
  - We get to choose, but typical values are 90%, 95%, and 99%

Confidence intervals as occasional liars

The **confidence level** of a CI determine how often the CI will be wrong.

A 95% confidence interval will:

- Tell you the truth in 95% of repeated samples (contain the population parameter 95% of the time)
- Lie to you in 5% of repeated sample (not contain the population parameter 5% of the time)

Can you tell if your particular confidence interval is telling the truth? No!

Percentile method

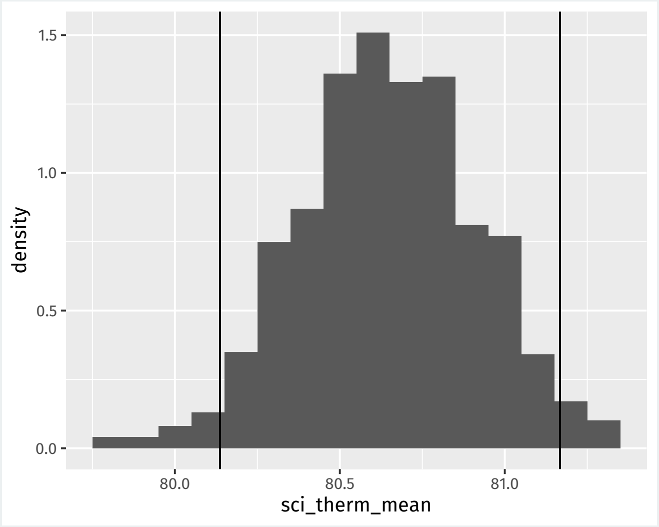
Percentile method: find the middle 95% of the bootstrap distribution.

We can do this by finding the points that the 2.5th percentile and the 97.5th percentile.

```r
perc_ci95 <- quantile(bootstrap_dist$sci_therm_mean,
                      probs = c(0.025, 0.975))
perc_ci95
```

```
##  2.5% 97.5%
##  80.1  81.2
```

Visualizing the CI



Width of the interval

What happens if we want the CI to be right more often? Will the width of a 99% confidence interval be wider or narrower?

99% confidence interval

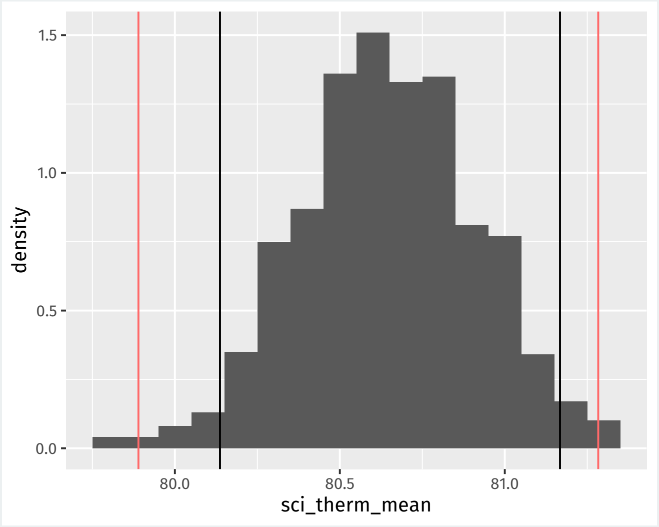
For 99% CI we need to find the middle 99% of the bootstrap distribution.

We can do this by finding the points that the 0.5th percentile and the 99.5th percentile.

```
perc_ci99 <- quantile(bootstrap_dist$sci_therm_mean,
                      probs = c(0.005, 0.995))
perc_ci99
```

```
##  0.5% 99.5%
##  79.9  81.3
```

Visualizing the CIs



3/ Calculating confidence intervals

`infer` package

Possible to use `quantile` to calculate CIs, but `infer` package is a more unified framework for CIs and hypothesis tests.

We'll use a `dplyr`-like approach of chained calls.

Step 1: define an outcome of interest

Start with defining the variable of interest:

```
anes |>
  specify(response = sci_therm)
```

```
## Response: sci_therm (numeric)
## # A tibble: 5,162 x 1
##    sci_therm
##        <dbl>
##  1        70
##  2       100
##  3        60
##  4        85
##  5        85
##  6        50
##  7       100
##  8        70
##  9        80
## 10        85
## # ... with 5,152 more rows
```

Step 2: generate bootstraps

Next `infer` can generate bootstraps with the `generate()` function (similar to `rep_slice_sample()`):

```
anes |>
  specify(response = sci_therm) |>
  generate(reps = 1000, type = "bootstrap")
```

```
## Response: sci_therm (numeric)
## # A tibble: 5,162,000 x 2
## # Groups:   replicate [1,000]
##    replicate sci_therm
##        <int>     <dbl>
##  1         1        85
##  2         1        85
##  3         1        60
##  4         1        70
##  5         1        70
##  6         1        85
##  7         1        90
##  8         1       100
##  9         1        50
## 10         1       100
## # ... with 5,161,990 more rows
```

Step 3: calculate sample statistics

Use `calculate()` to do the `group_by(replicate)` and `summarize` commands in one:

```
boot_dist_infer <- anes |>
  specify(response = sci_therm) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean")
```

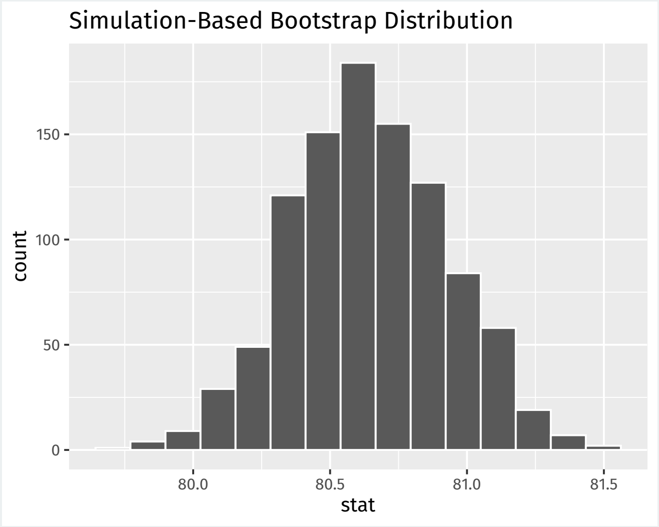
```r
boot_dist_infer
```

```
## Response: sci_therm (numeric)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  80.7
##  2         2  80.8
##  3         3  80.5
##  4         4  80.9
##  5         5  80.4
##  6         6  81.2
##  7         7  81.0
##  8         8  80.7
##  9         9  80.5
## 10        10  80.4
## # ... with 990 more rows
```

Step 3(b): visualize the boostrap distribution

`infer` also has a shortcut for plotting called `visualize()`:

```r
visualize(boot_dist_infer)
```



Step 4: calculate CIs

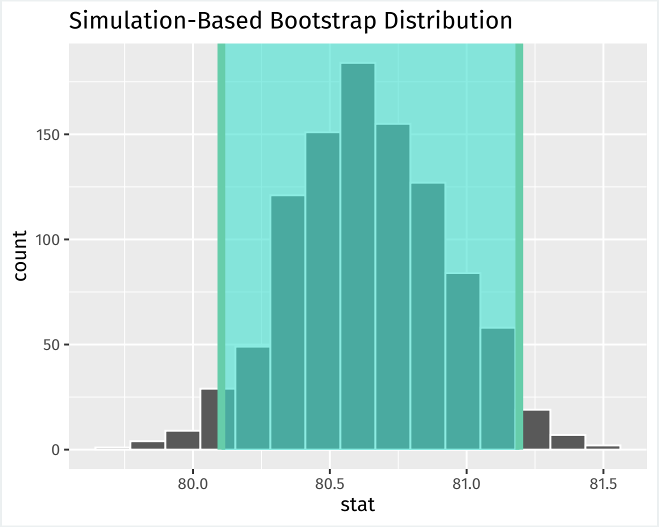
Finally we can calculate the CI using the percentile method with `get_confidence_interval()`:

```
perc_ci_95 <- boot_dist_infer |>
  get_confidence_interval(level = 0.95, type = "percentile")
perc_ci_95
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1     80.1     81.2
```

Step 4(b): visualize CIs

```r
visualize(boot_dist_infer) +
  shade_confidence_interval(endpoints = perc_ci_95)
```



Gov 50: 19. More Confidence Intervals

Matthew Blackwell

Harvard University

Roadmap

1. Bootstrap CIs for a difference in means
2. Bootstrap CIs for a difference in ATEs
3. Interpreting confidence intervals

1/ Bootstrap CIs for a difference in means

Comparison between groups

- Last time: confidence intervals for means.
- More interesting to compare across groups.
    - Differences in public opinion across groups
    - Difference between treatment and control groups.
- Bedrock of causal inference!

Trains experiment

- Back to the Boston trains example.
    - Boston commuter rail platform setting.
- Treatment group: presence of native Spanish-speaking confederates.
- Control group: no confederates.
- Outcome: X_i change in views on immigration.
    - Sample average in the treated group, $\overline{X}_T$
    - Sample average in the control group, $\overline{X}_C$
- Estimated **average treatment effect**

$$\widehat{\text{ATE}} = \overline{X}_T - \overline{X}_C$$

Inference for the difference

- Parameter: **population ATE** $\mu_T - \mu_C$
    - μ_T: Average outcome in the population if everyone received treatment.
    - μ_C: Average outcome in the population if everyone received control.
- Difference-in-means estimator: $\widehat{\text{ATE}} = \overline{X}_T - \overline{X}_C$
- $\overline{X}_T$ is a r.v. with mean $\mathbb{E}[\overline{X}_T] = \mu_T$
- $\overline{X}_C$ is a r.v. with mean $\mathbb{E}[\overline{X}_C] = \mu_C$
- $\rightsquigarrow \overline{X}_T - \overline{X}_C$ is a r.v. with mean $\mu_T - \mu_C$
    - Sample difference in means is on average equal to the population difference in means.

Trains data

```r
library(gov50data)
trains
```

```
## # A tibble: 115 x 14
##      age  male income white college usborn treatment ideol~1
##    <dbl> <dbl>  <dbl> <dbl>   <dbl>  <dbl>     <dbl>   <dbl>
##  1    31     0 135000     1       1      1         1       3
##  2    34     0 105000     1       1      0         1       4
##  3    63     1 135000     1       1      1         1       2
##  4    45     1 300000     1       1      1         1       4
##  5    55     1 135000     1       1      1         0       2
##  6    37     0  87500     1       1      1         1       5
##  7    53     0  87500     1       0      1         0       5
##  8    36     1 135000     1       1      1         1       4
##  9    54     0 105000     1       0      1         0       3
## 10    42     1 135000     1       1      1         1       4
## # ... with 105 more rows, 6 more variables:
## #   numberim.pre <dbl>, numberim.post <dbl>,
## #   remain.pre <dbl>, remain.post <dbl>, english.pre <dbl>,
## #   english.post <dbl>, and abbreviated variable name
## #   1: ideology
```

Estimating the difference in means

```r
diff_in_means <- trains |>
  group_by(treatment) |>
  summarize(post_mean = mean(numberim.post)) |>
  pivot_wider(names_from = treatment, values_from = post_mean) |>
  mutate(ATE = `1` - `0`)
diff_in_means
```

```
## # A tibble: 1 x 3
##     `0`   `1`   ATE
##   <dbl> <dbl> <dbl>
## 1  2.73  3.12 0.383
```

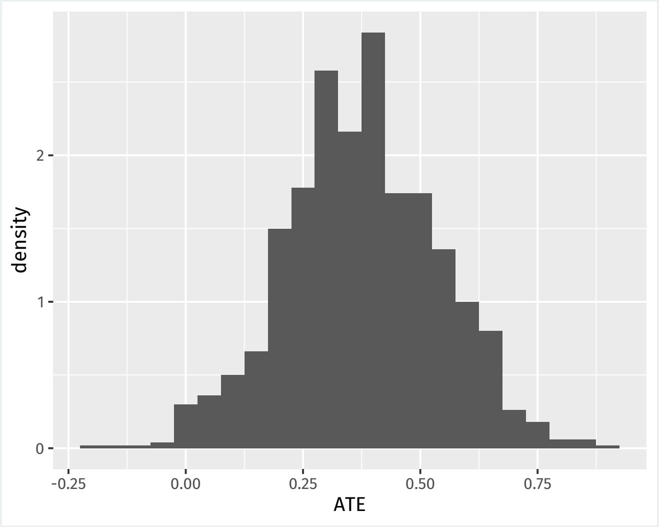
Bootstrap for the difference in means

```r
library(infer)
dim_boots <- trains |>
  rep_slice_sample(prop = 1, replace = TRUE, reps = 1000) |>
  group_by(replicate, treatment) |>
  summarize(post_mean = mean(numberim.post)) |>
  pivot_wider(names_from = treatment, values_from = post_mean) |>
  mutate(ATE = `1` - `0`)
dim_boots
```

```
## # A tibble: 1,000 x 4
## # Groups:   replicate [1,000]
##    replicate   `0`   `1`    ATE
##        <int> <dbl> <dbl>  <dbl>
##  1         1  2.83  3.02 0.194
##  2         2  2.67  3.07 0.406
##  3         3  2.74  3.09 0.346
##  4         4  2.79  3.19 0.398
##  5         5  2.76  3.13 0.376
##  6         6  2.62  3.14 0.520
##  7         7  2.87  3.27 0.395
##  8         8  2.71  3.07 0.360
##  9         9  3.03  3.26 0.229
```

Visualizing the bootstraps

```r
dim_boots |>
  ggplot(aes(x = ATE)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.05)
```



Calculating the percentile CI

You can use `get_confidence_interval()` with your “hand-rolled” bootstraps, but you have to make sure you only pass it the variable of interest using `select`:

```
dim_ci_95 <- dim_boots |>
  select(replicate, ATE) |>
  get_confidence_interval(level = 0.95, type = "percentile")

dim_ci_95
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0514    0.685
```

What about change in views as the outcome?

```r
change_ci_95 <- trains |>
  rep_slice_sample(prop = 1, replace = TRUE, reps = 1000) |>
  group_by(replicate, treatment) |>
  summarize(change_mean = mean(numberim.post - numberim.pre)) |>
  pivot_wider(names_from = treatment, values_from = change_mean) |>
  mutate(ATE = `1` - `0`) |>
  select(replicate, ATE) |>
  get_confidence_interval(level = 0.95, type = "percentile")
change_ci_95
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0157    0.613
```

What's different?

Let's look at the width of the two confidence intervals:

```
## Post outcome width
dim_ci_95[2]-dim_ci_95[1]
```

```
##   upper_ci
## 1    0.634
```

```
## Change outcome width
change_ci_95[2] - change_ci_95[1]
```

```
##   upper_ci
## 1    0.597
```

Width of CI depends on outcome variability

Change CI is narrower! Why? Because the change is less variable than the post outcome:

```
trains |> summarize(sd_post = sd(numberim.post),
                    sd_change = sd(numberim.post - numberim.pre))
```

```
## # A tibble: 1 x 2
##   sd_post sd_change
##     <dbl>     <dbl>
## 1   0.917     0.826
```

infer workflow

For infer, we have to do a bit of massaging. It wants the treatment variable to be a vector and we have to tell it what order we take the difference:

```
dim_boots_infer <- trains |>
  mutate(treatment = if_else(treatment == 1, "Treated", "Control")) |>
  specify(numberim.post ~ treatment) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "diff in means", order = c("Treated", "Control"))
dim_boots_infer |>
  get_confidence_interval(level = 0.95, type = "percentile")
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0569    0.720
```

2/ Bootstrap CIs for a difference in ATEs

Interactions

We have also estimated conditional ATEs:

$$ATE_{\text{college}} = \overline{X}_{T,\text{college}} - \overline{X}_{C,\text{college}}$$
$$ATE_{\text{noncollege}} = \overline{X}_{T,\text{noncollege}} - \overline{X}_{C,\text{noncollege}}$$

An **interaction** between treatment and college is the difference between these two effects:

$$ATE_{\text{college}} - ATE_{\text{noncollege}}$$

This is a random variable and has a **sampling distribution**.

Estimating the interaction

To estimate the interaction, we need to pivot both treatment and college to the columns.

```
trains |>
  mutate(
    treatment = if_else(treatment == 1, "Treated", "Control"),
    college = if_else(college == 1, "College", "Noncollege")
  ) |>
  group_by(treatment, college) |>
  summarize(post_mean = mean(numberim.post)) |>
  pivot_wider(
    names_from = c(treatment, college),
    values_from = post_mean
  )
```

```
## # A tibble: 1 x 4
##   Control_College Control_Noncollege Treated_College Treat~1
##             <dbl>              <dbl>           <dbl>   <dbl>
## 1            2.63               3.57            3.11    3.14
## # ... with abbreviated variable name 1: Treated_Noncollege
```

Estimating the interaction

```r
trains |>
  mutate(
    treatment = if_else(treatment == 1, "Treated", "Control"),
    college = if_else(college == 1, "College", "Noncollege")
  ) |>
  group_by(treatment, college) |>
  summarize(post_mean = mean(numberim.post)) |>
  pivot_wider(
    names_from = c(treatment, college),
    values_from = post_mean
  ) |>
  mutate(
    ATE_c = Treated_College - Control_College,
    ATE_nc = Treated_Noncollege - Control_Noncollege,
    interaction = ATE_c - ATE_nc
  ) |>
  select(ATE_c, ATE_nc, interaction)
```

```
## # A tibble: 1 x 3
##   ATE_c ATE_nc interaction
##   <dbl>  <dbl>       <dbl>
## 1 0.482 -0.429       0.911
```

Bootstrapping the interaction

```r
int_boots <- trains |>
  mutate(
    treatment = if_else(treatment == 1, "Treated", "Control"),
    college = if_else(college == 1, "College", "Noncollege")
  ) |>
  rep_slice_sample(prop = 1, replace = TRUE, reps = 1000) |>
  group_by(replicate, treatment, college) |>
  summarize(post_mean = mean(numberim.post)) |>
  pivot_wider(
    names_from = c(treatment, college),
    values_from = post_mean
  ) |>
  mutate(
    ATE_c = Treated_College - Control_College,
    ATE_nc = Treated_Noncollege - Control_Noncollege,
    interaction = ATE_c - ATE_nc
  ) |>
  select(replicate, ATE_c, ATE_nc, interaction)
```

```
int_boots
```

```
## # A tibble: 1,000 x 4
## # Groups:   replicate [1,000]
##    replicate ATE_c ATE_nc interaction
##        <int> <dbl>  <dbl>       <dbl>
##  1         1 0.580 -0.175       0.755
##  2         2 0.515 -0.458       0.973
##  3         3 0.753 -0.812       1.57
##  4         4 0.339  0.125       0.214
##  5         5 0.355  0           0.355
##  6         6 0.465 -0.568       1.03
##  7         7 0.492 -0.75        1.24
##  8         8 0.382 -0.5         0.882
##  9         9 0.277  0.125       0.152
## 10        10 0.449 -0.633       1.08
## # ... with 990 more rows
```

Getting the confidence interval

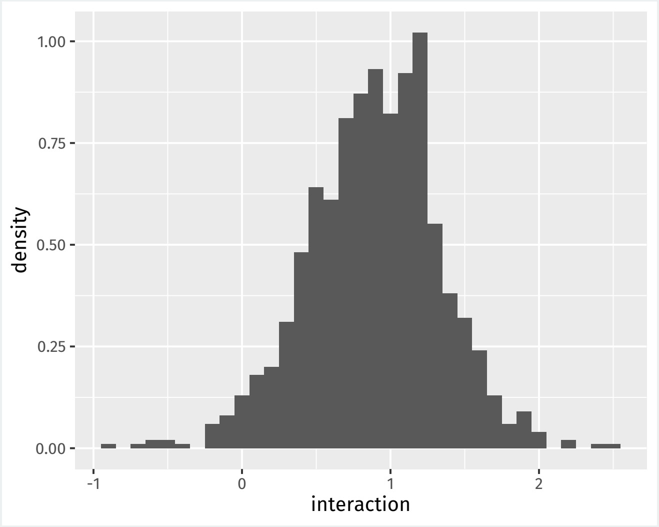
We have to drop NA values because sometimes the bootstrap gets a draw of all college or all noncollege and we can't calculate the interaction:

```
int_boots |>
  select(replicate, interaction) |>
  drop_na() |>
  get_confidence_interval(level = 0.95)
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1 -0.00805     1.72
```

Visualizing the bootstrap

```r
int_boots |>
  ggplot(aes(x = interaction)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1)
```

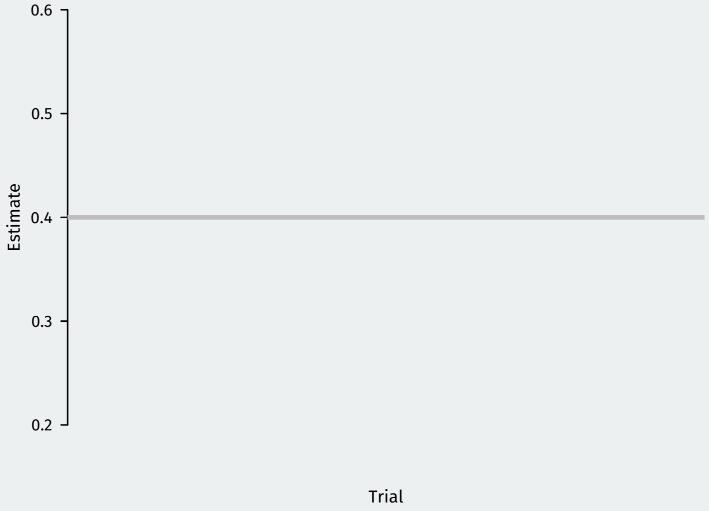


3/ Interpreting confidence intervals

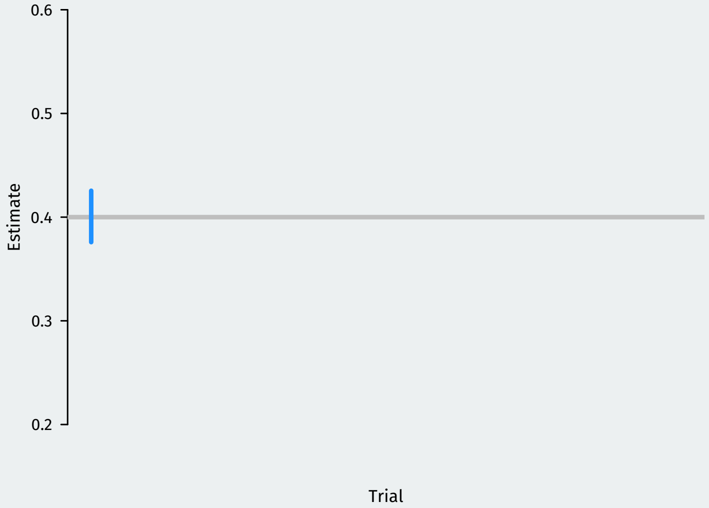
Interpretation and simulation

- Be careful about interpretation:
  - A 95% confidence interval will contain the true value in 95% of repeated samples.
  - For a particular calculated confidence interval, truth is either in it or not.
- A simulation can help our understanding:
  - Draw samples of size 1500 assuming population approval for Trump of $p = 0.4$.
  - Calculate 95% confidence intervals in each sample.
  - See how many overlap with the true population approval.

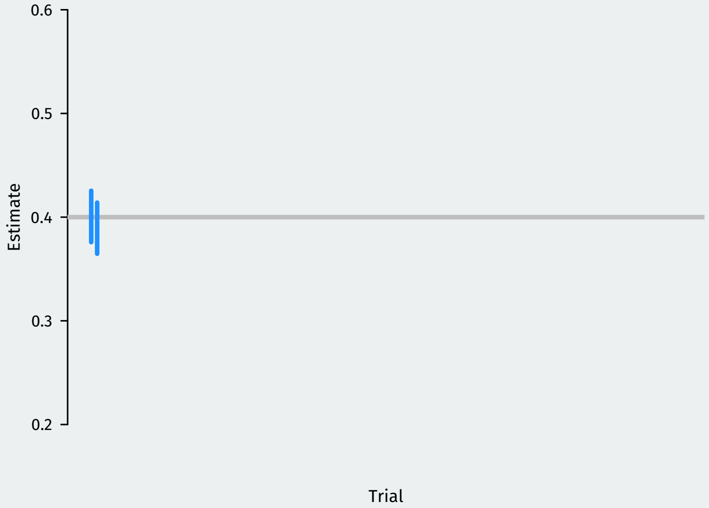
Plotting the CIs



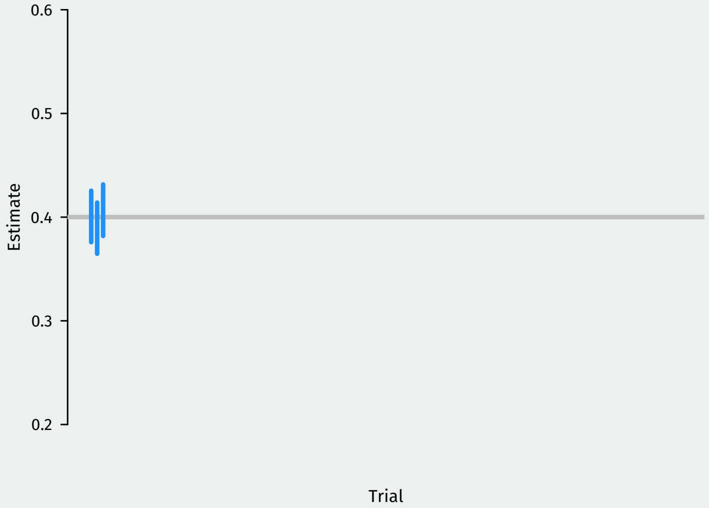
Plotting the CIs



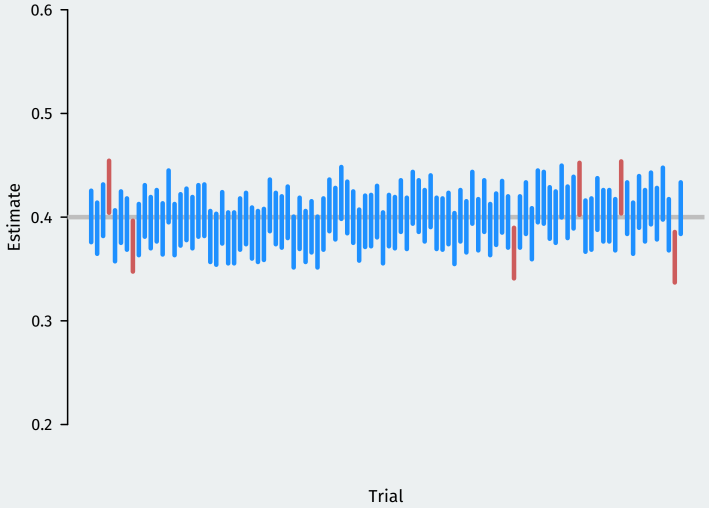
Plotting the CIs



Plotting the CIs



Plotting the CIs



Gov 50: 20. Hypothesis testing

Matthew Blackwell

Harvard University

Roadmap

1. The lady tasting tea
2. Hypothesis tests
3. Hypothesis testing using infer

1/ The lady tasting tea

The lady tasting tea

Your friend asks you to grab a tea with milk for her before meeting up and she says that she prefers tea poured before the milk. You stop by a local tea shop and ask for a tea with milk. When you bring it to her, she complains that it was prepared milk-first.

- You're skeptical that she can tell the difference, so you devise a test:
    - Prepare 8 cups of tea, 4 milk-first, 4 tea-first
    - Present cups to friend in a **random** order
    - Ask friend to pick which 4 of the 8 were milk-first.

Lady Tasting Tea data

Friend picks out all 4 milk-first cups correctly!

```r
library(gov50data)
tea
```

```
## # A tibble: 8 x 2
##   truth      guess
##   <chr>      <chr>
## 1 tea-first  tea-first
## 2 milk-first milk-first
## 3 milk-first milk-first
## 4 tea-first  tea-first
## 5 tea-first  tea-first
## 6 milk-first milk-first
## 7 tea-first  tea-first
## 8 milk-first milk-first
```

Thought experiment

Could she have been guessing at random? What would guessing look like?

```r
set.seed(02138)
one_guess <- tea |>
  mutate(random_guess = sample(guess))
one_guess
```

```
## # A tibble: 8 x 3
##   truth      guess      random_guess
##   <chr>      <chr>      <chr>
## 1 tea-first  tea-first  milk-first
## 2 milk-first milk-first tea-first
## 3 milk-first milk-first tea-first
## 4 tea-first  tea-first  milk-first
## 5 tea-first  tea-first  tea-first
## 6 milk-first milk-first milk-first
## 7 tea-first  tea-first  tea-first
## 8 milk-first milk-first milk-first
```

4 correct in this random guess!

Another guess

```r
another_guess <- tea |>
  mutate(random_guess = sample(guess))
another_guess
```

```
## # A tibble: 8 x 3
##   truth      guess      random_guess
##   <chr>      <chr>      <chr>       
## 1 tea-first  tea-first  tea-first   
## 2 milk-first milk-first tea-first   
## 3 milk-first milk-first milk-first  
## 4 tea-first  tea-first  tea-first   
## 5 tea-first  tea-first  milk-first  
## 6 milk-first milk-first milk-first  
## 7 tea-first  tea-first  tea-first   
## 8 milk-first milk-first milk-first
```

6 correct in this random guess!

All possible guesses

We could enumerate all possible guesses. “Guessing” would mean choosing one of these at random:

```
##   Cup 1 Cup 2 Cup 3 Cup 4 Cup 5 Cup 6 Cup 7 Cup 8
## 1  milk  milk  milk  milk   tea   tea   tea   tea
## 2  milk  milk  milk   tea  milk   tea   tea   tea
## 3  milk  milk   tea  milk  milk   tea   tea   tea
## 4  milk   tea  milk  milk  milk   tea   tea   tea
## 5   tea  milk  milk  milk  milk   tea   tea   tea
## 6  milk  milk  milk   tea   tea  milk   tea   tea
```

[snip]

```
##    Cup 1 Cup 2 Cup 3 Cup 4 Cup 5 Cup 6 Cup 7 Cup 8
## 65   tea   tea   tea  milk  milk   tea  milk  milk
## 66  milk   tea   tea   tea   tea  milk  milk  milk
## 67   tea  milk   tea   tea   tea  milk  milk  milk
## 68   tea   tea  milk   tea   tea  milk  milk  milk
## 69   tea   tea   tea  milk   tea  milk  milk  milk
## 70   tea   tea   tea   tea  milk  milk  milk  milk
```

Statistical thought experiment

- Statistical thought experiment: how often would she get all 4 correct **if she were guessing randomly**?
    - Only one way to choose all 4 correct cups.
    - But 70 ways of choosing 4 cups among 8.
    - Choosing at random: picking each of these 70 with equal probability.
- Chances of guessing all 4 correct is $\frac{1}{70} \approx 0.014$ or 1.4%.
- → the guessing hypothesis might be implausible.
    - Impossible? No, because of random chance!

2/ Hypothesis tests

Statistical hypothesis testing

- Statistical hypothesis testing is a **thought experiment**.
    - Could our results just be due to random chance?
- What would the world look like **if we knew the truth**?
- Example 1:
    - An analyst claims that 20% of Boston households are in poverty.
    - You take a sample of 900 households and find that 23% of the sample is under the poverty line.
    - Should you conclude that the analyst is wrong?
- Example 2:
    - Trump won 47.5% of the vote in the 2020 election.
    - Last YouGov poll of 1,363 likely voters said 44% planned to vote for Trump.
    - Could the difference between the poll and the outcome be just due to random chance?

Null and alternative hypothesis

- **Null hypothesis**: Some statement about the population parameters.
  - “Devil’s advocate” position $\rightsquigarrow$ assumes what you seek to prove wrong.
  - Usually that an observed difference is due to chance.
  - Ex: poll drawn from the same population as all voters.
  - Denoted H_0
- **Alternative hypothesis**: The statement we hope or suspect is true instead of H_0.
  - It is the opposite of the null hypothesis.
  - An observed difference is real, not just due to chance.
  - Ex: polling for Trump is systematically wrong.
  - Denoted H_1 or H_a
- **Probabilistic** proof by contradiction: try to “disprove” the null.

Hypothesis testing example

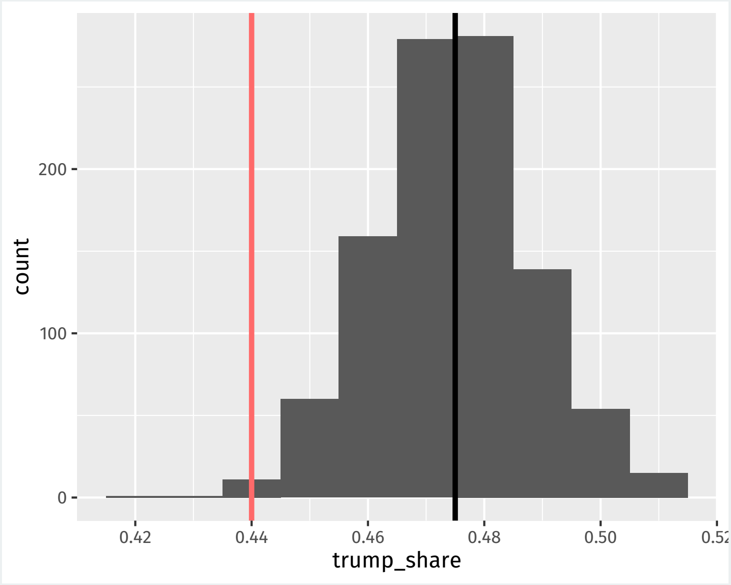
- Are we polling the same population as the actual voters?
    - If so, how likely are we to see polling error this big by chance?
- What is the parameter we want to learn about?
    - True population mean of the surveys, p.
    - Null hypothesis: $H_0 : p = 0.475$ (surveys drawing from same population)
    - Alternative hypothesis: $H_1 : p \neq 0.475$
- Data: poll has $\overline{X} = 0.44$ with $n = 1363$.

Statistical thought experiment

- If the null were true, what should the distribution of the data be?
  - X_i is 1 if respondent i will vote for Trump.
  - Under null, X_i is a coin flip with probability $p = 0.475$ of landing on "Trump".
  - $\sum_{i=1}^{n} X_i$ is the number in sample that will vote for Trump.
- We can simulate sums of coin flips using a function called `rbinom()`
- Compare the distribution of proportions under the null to the observed proportion.

```r
null_dist <- tibble(
  trump_share = rbinom(n = 1000, size = 1363, prob = 0.475) / 1363
)
ggplot(null_dist, aes(x = trump_share)) +
  geom_histogram(binwidth = 0.01) +
  geom_vline(xintercept = 0.44, color = "indianred1", size = 1.25) +
  geom_vline(xintercept = 0.475, size = 1.25)
```

Simulations of the reference distribution



p-value

p-value

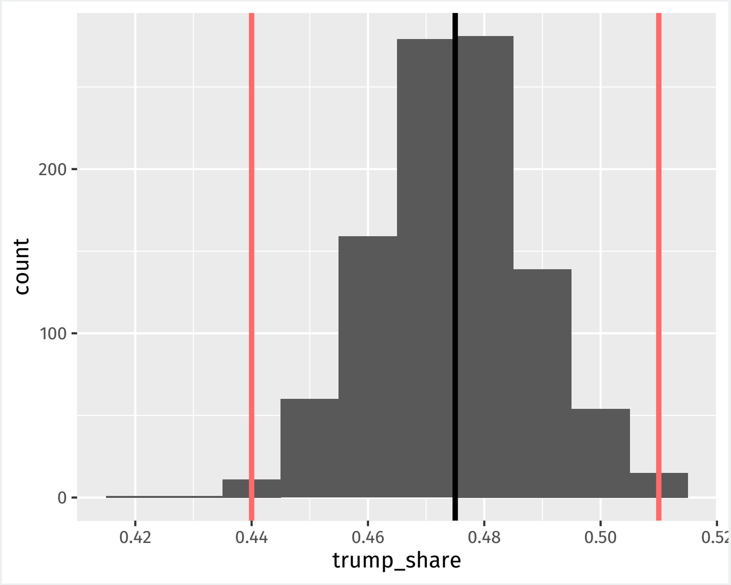
The **p-value** is the probability of observing data as or more extreme as our data under the null.

- If the null is true, how often would we expect polling errors this big?
  - Smaller p-value $\rightsquigarrow$ stronger evidence against the null
  - **NOT** the probability that the null is true!
- p-values are usually **two-sided**:
  - Observed error of 0.44 - 0.475 = -0.035 under the null.
  - p-value is probability of sample proportions being less than 0.44 **plus**
  - Probability of sample proportions being greater than 0.475 + 0.035 = 0.51.

```r
mean(null_dist$trump_share < 0.44) + mean(null_dist$trump_share > 0.51)
```

```
## [1] 0.01
```

Two-sided p-value



One-sided tests

- Sometimes our hypothesis is directional.
    - We only consider evidence against the null from one direction.
- Null: our polls are from the same population as actual voters
    - $H_0 : p = 0.475$
- **One-sided alternative**: polls underestimate Trump support.
    - $H_1 : p < 0.475$
- Makes the p-value one-sided:
    - What's the probability of a random sample underestimating Trump support by as much as we see in the sample?
    - Always smaller than a two-sided p-value.

```r
mean(null_dist$trump_share < 0.44)
```

```
## [1] 0.005
```

Rejecting the null

- Tests usually end with a decision to reject the null or not.
- Choose a threshold below which you'll reject the null.
    - **Test level** α: the threshold for a test.
    - Decision rule: "reject the null if the p-value is below α"
    - Otherwise "fail to reject" or "retain", not "accept the null"
- Common (arbitrary) thresholds:
    - $p \geq 0.1$ "not statistically significant"
    - $p < 0.05$ "statistically significant"
    - $p < 0.01$ "highly significant"

Testing errors

- A p-value of 0.05 says that data this extreme would only happen in 5% of repeated samples if the null were true.
  - $\rightsquigarrow$ 5% of the time we'll reject the null when it is actually true.
- Test errors:

	H_0 True	H_0 False
Retain H_0	Awesome!	Type II error
Reject H_0	Type I error	Good stuff!

- Type I error because it's the worst
  - "Convicting" an innocent null hypothesis
- Type II error less serious
  - Missed out on an awesome finding

3/ Hypothesis testing using infer

GSS data from `infer`

```r
library(infer)
gss
```

```
## # A tibble: 500 x 11
##     year   age sex    college   partyid hompop hours income
##    <dbl> <dbl> <fct>  <fct>     <fct>    <dbl> <dbl> <ord>
##  1  2014    36 male   degree    ind          3    50 $25000~
##  2  1994    34 female no degree rep          4    31 $20000~
##  3  1998    24 male   degree    ind          1    40 $25000~
##  4  1996    42 male   no degree ind          4    40 $25000~
##  5  1994    31 male   degree    rep          2    40 $25000~
##  6  1996    32 female no degree rep          4    53 $25000~
##  7  1990    48 female no degree dem          2    32 $25000~
##  8  2016    36 female degree    ind          1    20 $25000~
##  9  2000    30 female degree    rep          5    40 $25000~
## 10  1998    33 female no degree dem          2    40 $15000~
## # ... with 490 more rows, and 3 more variables:
## #   class <fct>, finrela <fct>, weight <dbl>
```

What is the average hours worked?

`dplyr` way:

```
gss |>
  summarize(mean(hours))
```

```
## # A tibble: 1 x 1
##   `mean(hours)`
##           <dbl>
## 1          41.4
```

`infer` way:

```
observed_mean <- gss |>
  specify(response = hours) |>
  calculate(stat = "mean")
observed_mean
```

```
## Response: hours (numeric)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  41.4
```

Hypothesis test

Could we get a mean this different from 40 hours if that was the true population average of hours worked?

Null and alternative:

$$
\begin{aligned}
H_0 &: \mu_{\texttt{hours}} = 40 \\
H_1 &: \mu_{\texttt{hours}} \neq 40
\end{aligned}
$$

How do we perform this test using infer? The **bootstrap!**

Specifying the hypotheses

```
gss |>
  specify(response = hours) |>
  hypothesize(null = "point", mu = 40)
```

```
## Response: hours (numeric)
## Null Hypothesis: point
## # A tibble: 500 x 1
##    hours
##    <dbl>
##  1    50
##  2    31
##  3    40
##  4    40
##  5    40
##  6    53
##  7    32
##  8    20
##  9    40
## 10    40
## # ... with 490 more rows
```

Generating the null distribution

We can use the bootstrap to determine how much variation there will be around 40 in the null distribution.

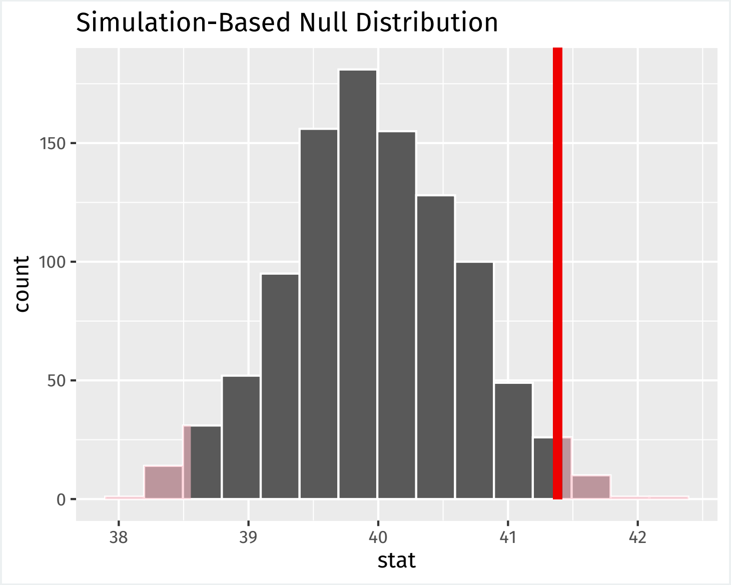
```r
null_dist <- gss |>
  specify(response = hours) |>
  hypothesize(null = "point", mu = 40) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean")
null_dist
```

```
## Response: hours (numeric)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  40.3
##  2         2  39.8
##  3         3  40.0
##  4         4  39.2
##  5         5  40.3
##  6         6  40.2
##  7         7  40.4
```

Visualizing the p-value

We can visualize our bootstrapped null distribution and the p-value as a shaded region:

```
null_dist |>
  visualize() +
  shade_p_value(observed_mean,
                direction = "two-sided")
```


Simulation-Based Null Distribution
count
150
100
50
0
38
39
40
41
42
stat

Gov 50: 21. More Hypothesis testing

Matthew Blackwell

Harvard University

Roadmap

1. Hypothesis testing using infer
2. Two-sample tests
3. Two-sample permutation tests with infer

1/ Hypothesis testing using infer

Statistical hypothesis testing

- Statistical hypothesis testing is a **thought experiment**.
- What would the world look like **if we knew the truth**?
- Conducted with several steps:

  1. Specify your **null** and **alternative hypotheses**
  2. Choose an appropriate **test statistic** and level of test α
  3. Derive the **reference distribution** of the test statistic under the null.
  4. Use this distribution to calculate the **p-value**.
  5. Use p-value to decide whether to reject the null hypothesis or not

GSS data from `infer`

```r
library(infer)
gss
```

```
## # A tibble: 500 x 11
##     year   age sex    college   partyid hompop hours income
##    <dbl> <dbl> <fct>  <fct>     <fct>    <dbl> <dbl> <ord>
##  1  2014    36 male   degree    ind          3    50 $25000~
##  2  1994    34 female no degree rep          4    31 $20000~
##  3  1998    24 male   degree    ind          1    40 $25000~
##  4  1996    42 male   no degree ind          4    40 $25000~
##  5  1994    31 male   degree    rep          2    40 $25000~
##  6  1996    32 female no degree rep          4    53 $25000~
##  7  1990    48 female no degree dem          2    32 $25000~
##  8  2016    36 female degree    ind          1    20 $25000~
##  9  2000    30 female degree    rep          5    40 $25000~
## 10  1998    33 female no degree dem          2    40 $15000~
## # ... with 490 more rows, and 3 more variables:
## #   class <fct>, finrela <fct>, weight <dbl>
```

What is the average hours worked?

`dplyr` way:

```
gss |>
  summarize(mean(hours))
```

```
## # A tibble: 1 x 1
##   `mean(hours)`
##           <dbl>
## 1          41.4
```

`infer` way:

```
observed_mean <- gss |>
  specify(response = hours) |>
  calculate(stat = "mean")
observed_mean
```

```
## Response: hours (numeric)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  41.4
```

Hypothesis test

Could we get a mean this different from 40 hours if that was the true population average of hours worked?

Null and alternative:

$$H_0 : \mu_{\texttt{hours}} = 40$$
$$H_1 : \mu_{\texttt{hours}} \neq 40$$

How do we perform this test using infer? The **bootstrap!**

Specifying the hypotheses

```r
gss |>
  specify(response = hours) |>
  hypothesize(null = "point", mu = 40)
```

```
## Response: hours (numeric)
## Null Hypothesis: point
## # A tibble: 500 x 1
##    hours
##    <dbl>
##  1    50
##  2    31
##  3    40
##  4    40
##  5    40
##  6    53
##  7    32
##  8    20
##  9    40
## 10    40
## # ... with 490 more rows
```

Generating the null distribution

We can use the bootstrap to determine how much variation there will be around 40 in the null distribution.

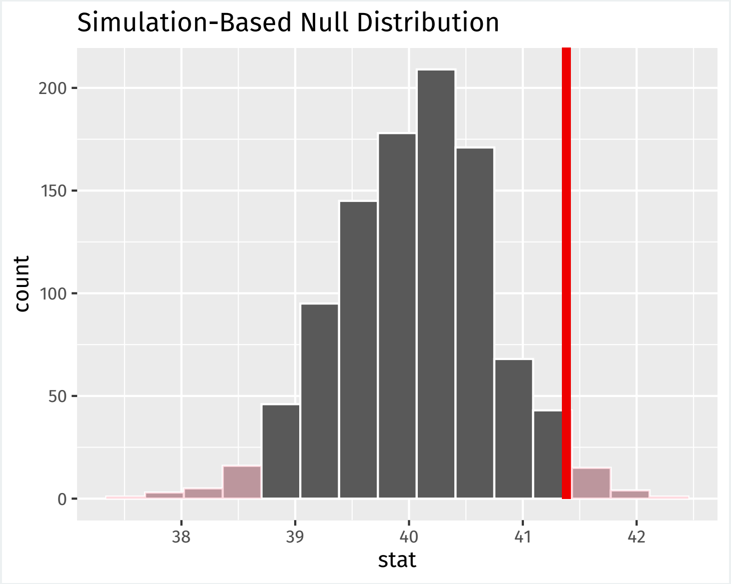
```r
null_dist <- gss |>
  specify(response = hours) |>
  hypothesize(null = "point", mu = 40) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean")
null_dist
```

```
## Response: hours (numeric)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  40.3
##  2         2  39.6
##  3         3  40.8
##  4         4  39.6
##  5         5  39.8
##  6         6  39.8
##  7         7  40.6
```

Visualizing the p-value

We can visualize our bootstrapped null distribution and the p-value as a shaded region:

```r
null_dist |>
  visualize() +
  shade_p_value(observed_mean,
                direction = "two-sided")
```

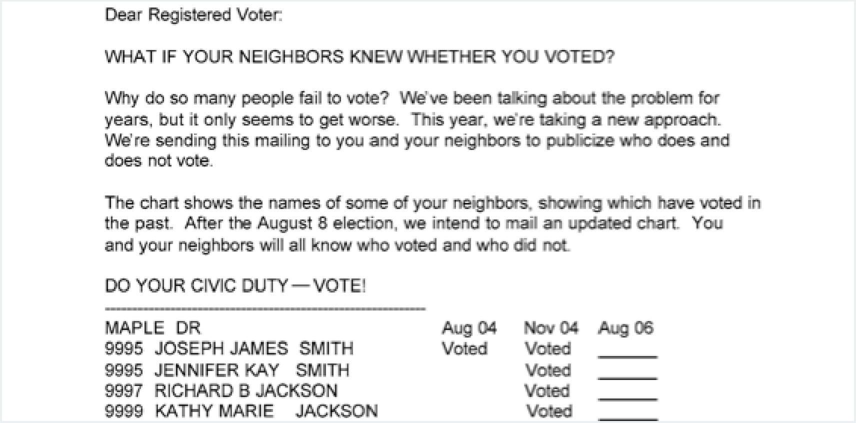

Simulation-Based Null Distribution
count
200
150
100
50
0
38
39
40
41
42
stat

2/ Two-sample tests

Social pressure experiment

- Experimental study where each household for 2006 MI primary was randomly assigned to one of 4 conditions:
    - Control: no mailer
    - Civic Duty: mailer saying voting is your civic duty.
    - Hawthorne: a “we’re watching you” message.
    - Neighbors: naming-and-shaming social pressure mailer.
- Outcome: whether household members voted or not.
- We’ll focus on Neighbors vs Control
- Randomized implies samples are **independent**

Neighbors mailer



Dear Registered Voter:

WHAT IF YOUR NEIGHBORS KNEW WHETHER YOU VOTED?

Why do so many people fail to vote? We've been talking about the problem for years, but it only seems to get worse. This year, we're taking a new approach. We're sending this mailing to you and your neighbors to publicize who does and does not vote.

The chart shows the names of some of your neighbors, showing which have voted in the past. After the August 8 election, we intend to mail an updated chart. You and your neighbors will all know who voted and who did not.

DO YOUR CIVIC DUTY — VOTE!

MAPLE DR	Aug 04	Nov 04	Aug 06
9995 JOSEPH JAMES SMITH	Voted	Voted	______
9995 JENNIFER KAY SMITH		Voted	______
9997 RICHARD B JACKSON		Voted	______
9999 KATHY MARIE JACKSON		Voted	______

Social pressure data

```r
data(social, package = "qss")
social <- as_tibble(social)
social
```

```
## # A tibble: 305,866 x 6
##    sex    yearofbirth primary2004 messages   primar~1 hhsize
##    <chr>        <int>       <int> <chr>         <int>  <int>
##  1 male          1941           0 Civic Duty        0      2
##  2 female        1947           0 Civic Duty        0      2
##  3 male          1951           0 Hawthorne         1      3
##  4 female        1950           0 Hawthorne         1      3
##  5 female        1982           0 Hawthorne         1      3
##  6 male          1981           0 Control           0      3
##  7 female        1959           0 Control           1      3
##  8 male          1956           0 Control           1      3
##  9 female        1968           0 Control           0      2
## 10 male          1967           0 Control           0      2
## # ... with 305,856 more rows, and abbreviated variable name
## #   1: primary2006
```

Two-sample hypotheses

- Parameter: **population ATE** $\mu_T - \mu_C$
    - μ_T: Turnout rate in the population if everyone received treatment.
    - μ_C: Turnout rate in the population if everyone received control.
- Goal: learn about the population difference in means
- Usual null hypothesis: no difference in population means (ATE = 0)
    - Null: $H_0 : \mu_T - \mu_C = 0$
    - Two-sided alternative: $H_1 : \mu_T - \mu_C \neq 0$
- In words: are the differences in sample means just due to chance?

Permutation test

How do we generate draws of the difference in means under the null?
$H_0 : \mu_T - \mu_C = 0$

If the voting distribution is the same in the treatment and control groups, we could randomly swap who is labelled as treated and who is labelled as control and it shouldn’t matter.

Permutation test: generate the null distribution by permuting the group labels and see the resulting distribution of differences in proportions

Permuting the labels

```r
social <- social |>
  filter(messages %in% c("Neighbors", "Control"))

social |>
  mutate(messages_permute = sample(messages)) |>
  select(primary2006, messages, messages_permute)
```

```
## # A tibble: 229,444 x 3
##    primary2006 messages messages_permute
##          <int> <chr>    <chr>
##  1           0 Control  Control
##  2           1 Control  Control
##  3           1 Control  Neighbors
##  4           0 Control  Control
##  5           0 Control  Control
##  6           1 Control  Neighbors
##  7           0 Control  Control
##  8           1 Control  Control
##  9           1 Control  Control
## 10           1 Control  Control
## # ... with 229,434 more rows
```

3/ Two-sample permutation tests with infer

Calculating the difference in proportion

`infer` functions with binary outcomes work best with factor variables:

```r
social <- social |>
  mutate(turnout = if_else(primary2006 == 1, "Voted", "Didn't Vote"))

est_ate <- social |>
  specify(turnout ~ messages, success = "Voted") |>
  calculate(stat = "diff in props", order = c("Neighbors", "Control"))
est_ate
```

```
## Response: turnout (factor)
## Explanatory: messages (factor)
## # A tibble: 1 x 1
##     stat
##    <dbl>
## 1 0.0813
```

Specifying the relationship of interest

infer functions with binary outcomes work best with factor variables:

```r
social |>
  specify(turnout ~ messages, success = "Voted")
```

```
## Response: turnout (factor)
## Explanatory: messages (factor)
## # A tibble: 229,444 x 2
##    turnout    messages
##    <fct>      <fct>
##  1 Didn't Vote Control
##  2 Voted       Control
##  3 Voted       Control
##  4 Didn't Vote Control
##  5 Didn't Vote Control
##  6 Voted       Control
##  7 Didn't Vote Control
##  8 Voted       Control
##  9 Voted       Control
## 10 Voted       Control
## # ... with 229,434 more rows
```

Setting the hypotheses

The null for these two-sample tests is called `"independence"` for the `infer` package because the assumption is that the two variables are statistically independent.

```
social |>
  specify(turnout ~ messages, success = "Voted") |>
  hypothesize(null = "independence")
```

```
## Response: turnout (factor)
## Explanatory: messages (factor)
## Null Hypothesis: independence
## # A tibble: 229,444 x 2
##    turnout     messages
##    <fct>       <fct>
##  1 Didn't Vote Control
##  2 Voted       Control
##  3 Voted       Control
##  4 Didn't Vote Control
##  5 Didn't Vote Control
##  6 Voted       Control
##  7 Didn't Vote Control
##  8 Voted       Control
```

Generating the permutations

We can tell `infer` to do our permutation test by using the argument `type = "permute"` to `generate()`:

```
social |>
  specify(turnout ~ messages, success = "Voted") |>
  hypothesize(null = "independence") |>
  generate(reps = 1000, type = "permute")
```

```
## Response: turnout (factor)
## Explanatory: messages (factor)
## Null Hypothesis: independence
## # A tibble: 229,444,000 x 3
## # Groups:   replicate [1,000]
##    turnout     messages replicate
##    <fct>       <fct>        <int>
##  1 Voted       Control          1
##  2 Didn't Vote Control          1
##  3 Voted       Control          1
##  4 Didn't Vote Control          1
##  5 Didn't Vote Control          1
##  6 Voted       Control          1
##  7 Voted       Control          1
```

Calculating the diff in proportions in each sample

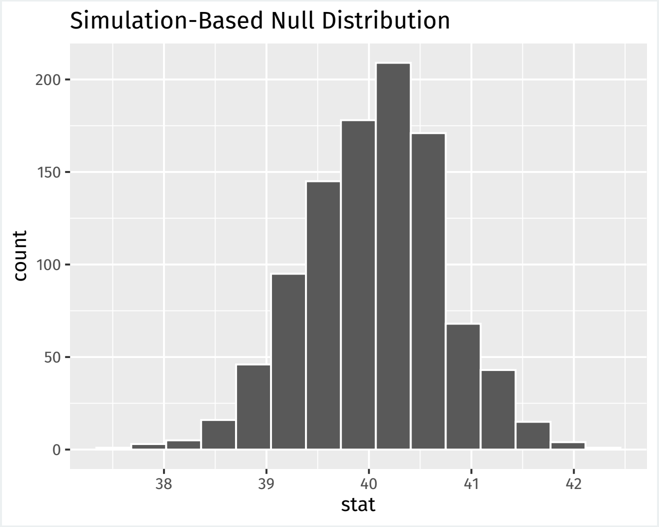
```r
null_dist <- social |>
  specify(turnout ~ messages, success = "Voted") |>
  hypothesize(null = "independence") |>
  generate(reps = 1000, type = "permute") |>
  calculate(stat = "diff in props", order = c("Neighbors", "Control"))
```

```r
null_dist
```

```
## Response: hours (numeric)
## Null Hypothesis: point
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  40.3
##  2         2  39.6
##  3         3  40.8
##  4         4  39.6
##  5         5  39.8
##  6         6  39.8
##  7         7  40.6
##  8         8  40.5
##  9         9  38.6
## 10        10  41.2
## # ... with 990 more rows
```

Visualizing

```r
null_dist |>
  visualize()
```



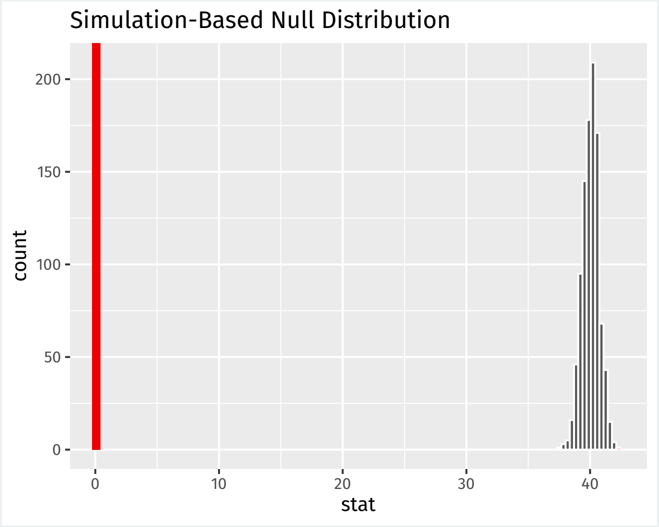
Calculating p-values

```r
ate_pval <- null_dist |>
  get_p_value(obs_stat = est_ate, direction = "both")
ate_pval
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1       0
```

Visualizing p-values

```
null_dist |>
  visualize() +
  shade_p_value(obs_stat = est_ate, direction = "both")
```



Gov 50: 22. More Hypothesis testing

Matthew Blackwell

Harvard University

Roadmap

1. Reviewing hypothesis testing
2. Issues with hypothesis testing
3. Power Analyses

1/ Reviewing hypothesis testing

Difference-in-means

```r
library(gov50data)
trains <- trains |>
  mutate(treated = if_else(treatment == 1, "Treated", "Untreated"))
trains
```

```
## # A tibble: 115 x 15
##      age  male income white college usborn treatment ideol~1
##    <dbl> <dbl>  <dbl> <dbl>   <dbl>  <dbl>     <dbl>   <dbl>
##  1    31     0 135000     1       1      1         1       3
##  2    34     0 105000     1       1      0         1       4
##  3    63     1 135000     1       1      1         1       2
##  4    45     1 300000     1       1      1         1       4
##  5    55     1 135000     1       1      1         0       2
##  6    37     0  87500     1       1      1         1       5
##  7    53     0  87500     1       0      1         0       5
##  8    36     1 135000     1       1      1         1       4
##  9    54     0 105000     1       0      1         0       3
## 10    42     1 135000     1       1      1         1       4
## # ... with 105 more rows, 7 more variables:
## #   numberim.pre <dbl>, numberim.post <dbl>,
## #   remain.pre <dbl>, remain.post <dbl>, english.pre <dbl>,
## #   english.post <dbl>, treated <chr>, and abbreviated
```

Calculating the ATE

```r
library(infer)
ate <- trains |>
  specify(numberim.post ~ treated) |>
  calculate(stat = "diff in means",
            order = c("Treated", "Untreated"))
ate
```

```
## Response: numberim.post (numeric)
## Explanatory: treated (factor)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1 0.383
```

Difference in means hypotheses

Hypotheses:

$$H_0 : \mu_T - \mu_C = 0$$
$$H_1 : \mu_T - \mu_c \neq 0$$

Observed difference in means:

$$\widehat{ATE} = \overline{Y}_T - \overline{Y}_C$$

How can we approximate the **null distribution**? **Permute** the outcome/treatment variables.

Permuting the treatment

Let’s do 2 permutations to see how things vary:

```
set.seed(02138)
perm <- trains |>
  specify(numberim.post ~ treated) |>
  hypothesize(null = "independence") |>
  generate(reps = 1000,
           type = "permute")
```

`generate(type = ”permute”)` shuffles to the outcomes, keeping treatment the same:

```r
perm |> filter(replicate == 1)
```

```
## # A tibble: 115 x 3
## # Groups:   replicate [1]
##   numberim.post treated   replicate
##           <dbl> <fct>         <int>
##  1            3 Treated           1
##  2            2 Treated           1
##  3            5 Treated           1
##  4            3 Treated           1
##  5            3 Untreated         1
##  6            3 Treated           1
##  7            2 Untreated         1
##  8            2 Treated           1
##  9            3 Untreated         1
## 10            3 Treated           1
## # ... with 105 more rows
```

```r
perm |> filter(replicate == 2)
```

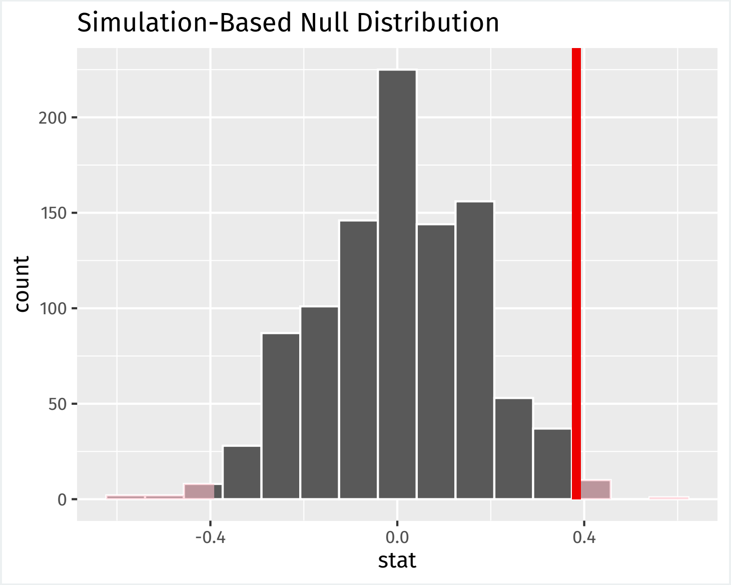
```
## # A tibble: 115 x 3
## # Groups:   replicate [1]
##   numberim.post treated   replicate
##           <dbl> <fct>         <int>
##  1            2 Treated           2
##  2            3 Treated           2
##  3            3 Treated           2
##  4            3 Treated           2
##  5            3 Untreated         2
##  6            4 Treated           2
##  7            2 Untreated         2
##  8            3 Treated           2
##  9            3 Untreated         2
## 10            2 Treated           2
## # ... with 105 more rows
```

Null distribution

The distribution of the differences-in-means under permutation will be mean 0 because shuffling the outcomes means that the outcomes in each permutation's treated and control group are coming from the same distribution.

```
null_dist <- trains |>
  specify(numberim.post ~ treated) |>
  hypothesize(null = "independence") |>
  generate(reps = 1000,
           type = "permute") |>
  calculate(stat = "diff in means", order = c("Treated", "Untreated"))
```

```r
null_dist |>
  visualize() +
  shade_p_value(obs_stat = ate, direction = "both")
```



Interpreting p-values

```r
get_p_value(null_dist, obs_stat = ate, direction = "both")
```

```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1   0.022
```

Hypotheses:

$$H_0 : \mu_T - \mu_C = 0$$
$$H_1 : \mu_T - \mu_c \neq 0$$

Observed difference in means:

$$\widehat{ATE} = \overline{Y}_T - \overline{Y}_C$$

p-value: probability of an estimated ATE as big as $|\widehat{ATE}|$ by random chance if there is no treatment effect.

Rejecting the null

Decision rule: “reject the null if the p-value is below the **test level** α”

Rejecting the null in two-sample tests: there is a true difference in means.

Test level α controls the amount of false positives:

	Null False (True difference)	Null True (No true difference)
Reject Null	True Positive	False Positive (Type I error)
Retain Null	False Negative (Type II error)	True Negative

Tests and confidence intervals

- There is a deep connection between confidence intervals and tests.
- Any value outside of a $100 \times (1 - \alpha)\%$ confidence interval would have a p-value less than α if we tested it as the null hypothesis.
    - 95% CI for social pressure experiment: $[0.016, 0.124]$
    - $\leadsto$ p-value for $H_0 : \mu_T - \mu_C = 0$ less than 0.05.
- Confidence intervals are all of the null hypotheses we **can't reject** with a test.

CI in the trains example

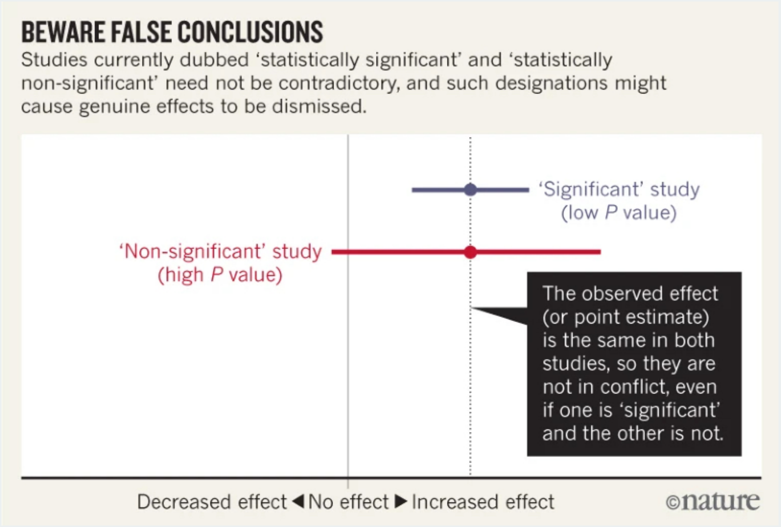
```r
trains |>
  specify(numberim.post ~ treated) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "diff in means",
            order = c("Treated", "Untreated")) |>
  get_ci(level = 0.95)
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1   0.0893    0.698
```

2/ Issues with hypothesis testing

Significant vs not significant

The difference between statistically significant and not statistically significant is itself not statistically significant:

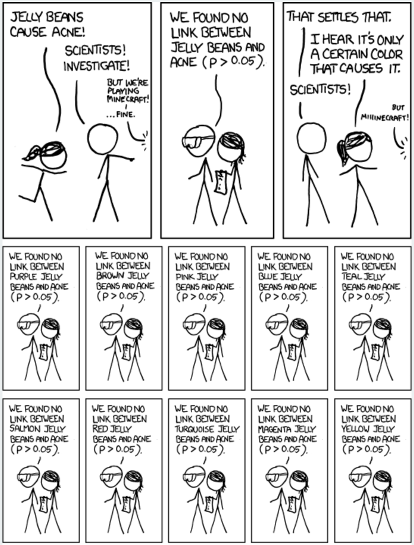


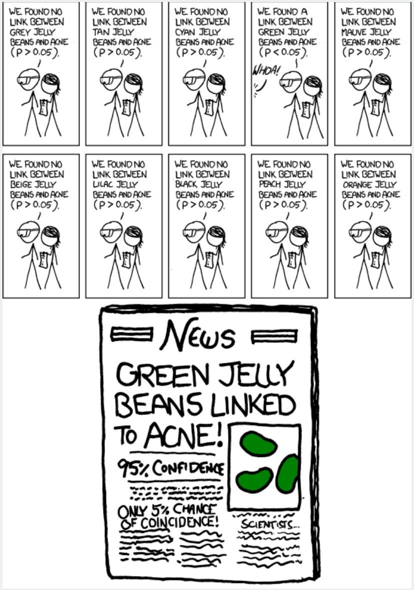
What kind of significance

There are different types of significance that don’t all have to be true together:

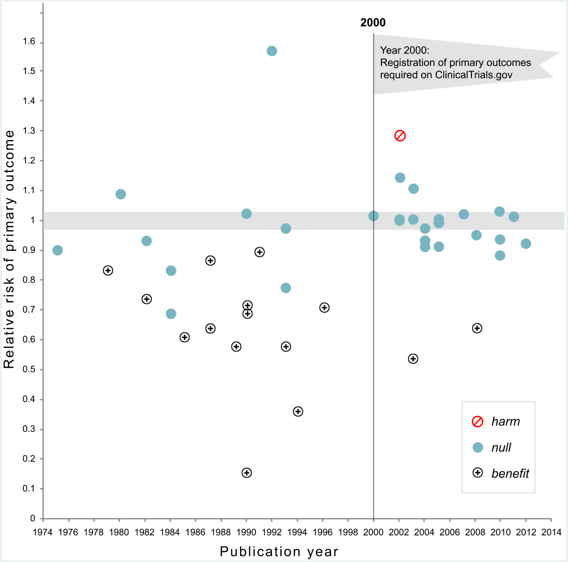
1. **Statistical significance:** we can reject the null of no effect.
2. **Causal significance**: we can interpret our estimated difference in means as a causal effect.
3. **Practical significance**: the estimated effect is meaningfully large.

p-hacking


JELLY BEANS CAUSE ACNE!
SCIENTISTS! INVESTIGATE!
BUT WE'RE PLAYING MINECRAFT!
... FINE.
WE FOUND NO LINK BETWEEN JELLY BEANS AND ACNE (P > 0.05).
THAT SETTLES THAT.
I HEAR IT'S ONLY A CERTAIN COLOR THAT CAUSES IT.
SCIENTISTS!
BUT MINECRAFT!
WE FOUND NO LINK BETWEEN PURPLE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BROWN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PINK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLUE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TEAL JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN SALMON JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN RED JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TURQUOISE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN MAGENTA JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN YELLOW JELLY BEANS AND ACNE (P > 0.05).


WE FOUND NO LINK BETWEEN GREY JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN TAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN CYAN JELLY BEANS AND ACNE (P > 0.05).
WE FOUND A LINK BETWEEN GREEN JELLY BEANS AND ACNE (P < 0.05).
WHOA!
WE FOUND NO LINK BETWEEN MAUVE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BEIGE JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN LILAC JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN BLACK JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN PEACH JELLY BEANS AND ACNE (P > 0.05).
WE FOUND NO LINK BETWEEN ORANGE JELLY BEANS AND ACNE (P > 0.05).
NEWS
GREEN JELLY BEANS LINKED TO ACNE!
95% CONFIDENCE
ONLY 5% CHANCE OF COINCIDENCE!
SCIENTISTS...

p-hacking



2000
Year 2000:
Registration of primary outcomes
required on ClinicalTrials.gov
Relative risk of primary outcome
Publication year
harm
null
benefit
1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6

3/ Power Analyses

Effect sizes

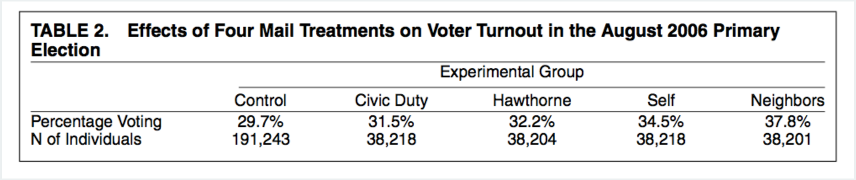


TABLE 2. Effects of Four Mail Treatments on Voter Turnout in the August 2006 Primary Election

	Experimental Group				
	Control	Civic Duty	Hawthorne	Self	Neighbors
Percentage Voting	29.7%	31.5%	32.2%	34.5%	37.8%
N of Individuals	191,243	38,218	38,204	38,218	38,201

- Why did Gerber, Green, and Larimer use sample sizes of 38,000 for each treatment condition?
- Choose the sample size to ensure that you can *detect* what you think might be the true treatment effect:
  - Small effect sizes (half percentage point) will require huge *n*
  - Large effect sizes (10 percentage points) will require smaller *n*
- **Detect** here means “reject the null of no effect”

Power of a test

- **Definition** The **power** of a test is the probability that a test rejects the null.
    - Probability that we reject given some specific value of the parameter
    - Power = $1 - \mathbb{P}(\text{Type II error})$
    - Better tests = higher power.
- If we fail to reject a null hypothesis, two possible states of the world:
    - Null is true (no treatment effect)
    - Null is false (there is a treatment effect), but test had low power.

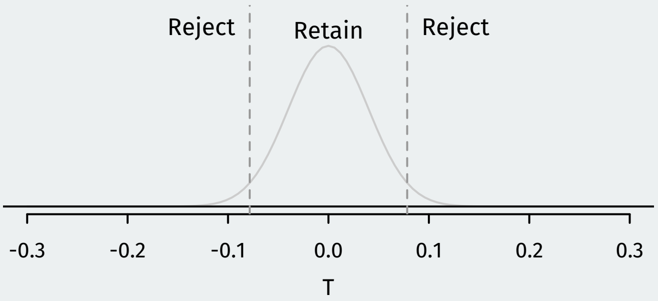
Why care about power?

- Imagine you are a company being sued for racial discrimination in hiring.
- Judge forces you to conduct hypothesis test:
  - Null hypothesis is that hiring rates for white and black people are equal, $H_0 : \mu_w - \mu_b = 0$
  - You sample 10 hiring records of each race, conduct hypothesis test and fail to reject null.
- Say to judge, “look we don’t have any racial discrimination”! What’s the problem?

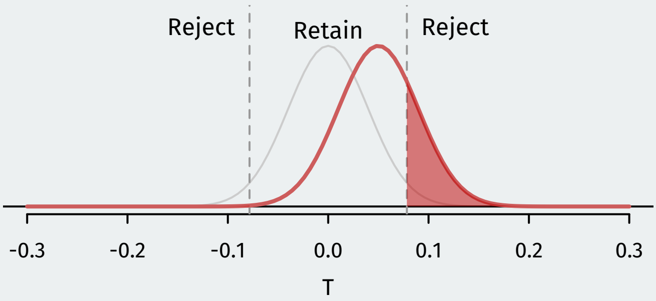
Power analysis procedure

- Power can help guide the choice of sample size through a **power analysis**.
    - Calculate how likely we are to reject different possible treatment effects at different sample sizes.
    - **Can be done before the experiment**: which effects will I be able to detect with high probability at my n?
- Steps to a power analysis:
    - Pick some hypothetical effect size, $\mu_T - \mu_C = 0.05$
    - Calculate the distribution of T under that effect size.
    - Calculate the probability of rejecting the null under that distribution.
    - Repeat for different effect sizes.

Power graph

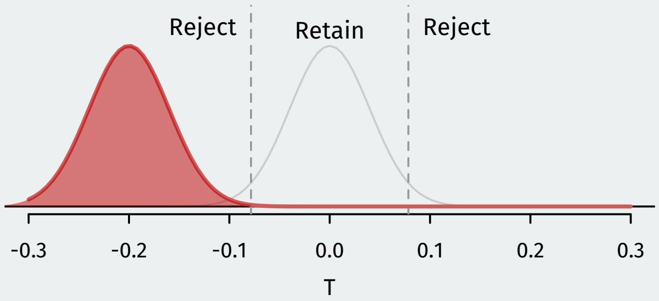


Power graph



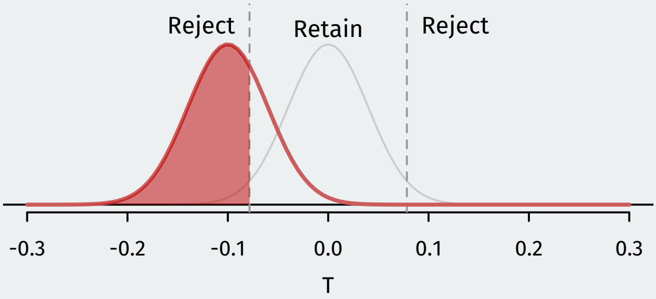
Assumed treatment effect $= 0.05$ and power $= 0.24$.

Power graph



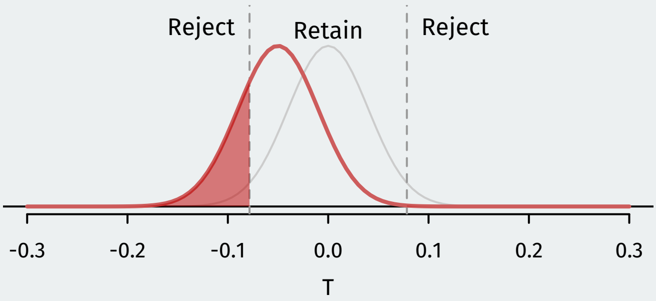
Assumed treatment effect = -0.2 and power = 0.999.

Power graph



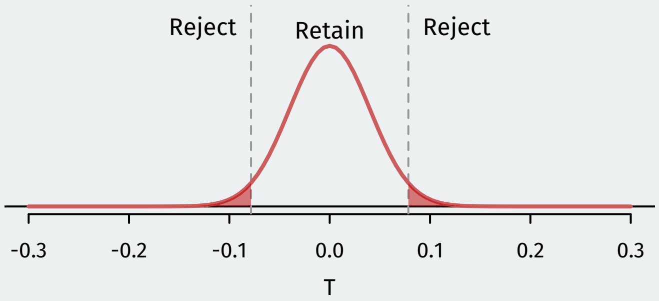
Assumed treatment effect = -0.1 and power = 0.705.

Power graph



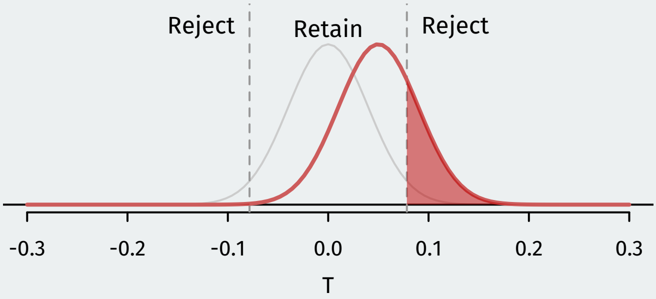
Assumed treatment effect = -0.05 and power = 0.24.

Power graph



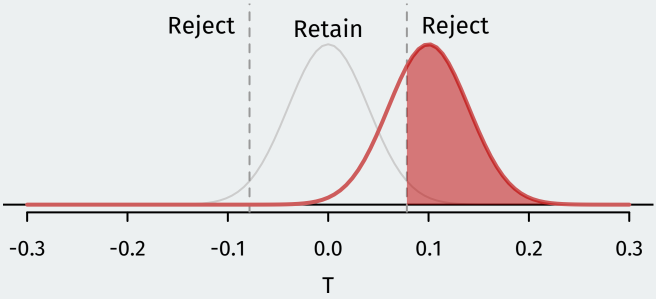
Assumed treatment effect $=$ 0 and power $=$ 0.05.

Power graph



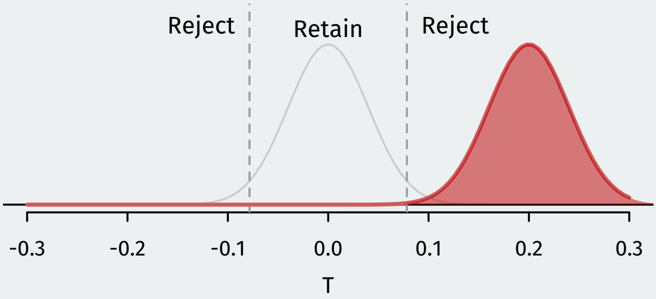
Assumed treatment effect $= 0.05$ and power $= 0.24$.

Power graph



Assumed treatment effect = 0.1 and power = 0.705.

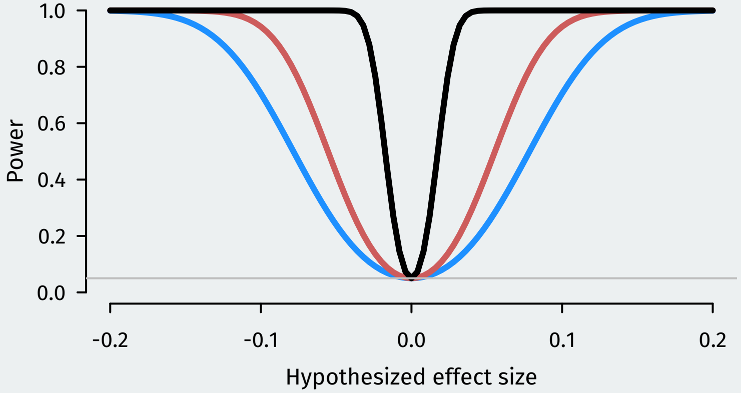
Power graph



Assumed treatment effect = 0.2 and power = 0.999.

A power analysis

- We can calculate the power for every possible effect size and plot the resulting **power curve**:
  - $n = 500$ (blue), 1000 (red), 10000 (black)



Gov 50: 23. Inference with Mathematical Models

Matthew Blackwell

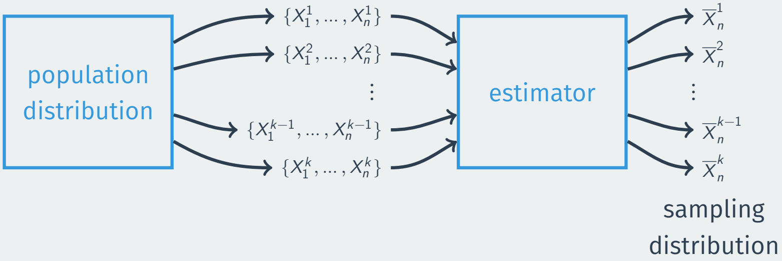
Harvard University

Roadmap

1. Central limit theorem
2. Normal distribution
3. Using the Normal for inference

1/ Central limit theorem

Sampling distribution, in pictures



Sampling distribution of the sample proportion

$$\text{sample mean} = \text{population mean} + \text{chance error}$$
$$\overline{X} = \mu + \text{chance error}$$

Then $\overline{X}$ centered at μ.

Spread: standard deviation of the sampling distribution is the **standard error**

Spread of the sample mean

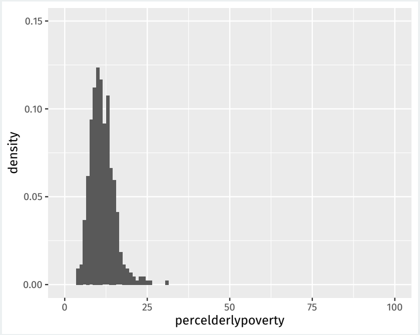
- **Standard error**: how big is the chance error on average?
    - This is the standard deviation of the estimator **across repeated samples**.
    - With random samples, we can get a formula for the SE for many estimators.
- Standard error for the sample mean:

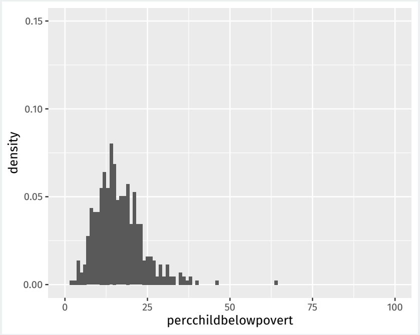
$$SE = \frac{\sigma}{\sqrt{n}} = \frac{\text{population standard deviation}}{\sqrt{\text{sample size}}}$$

- Two components:
    - Population SD: more spread of the variable in the population → more spread of sample means
    - Size of the sample: larger sample → smaller spread of the sample means

Midwest counties

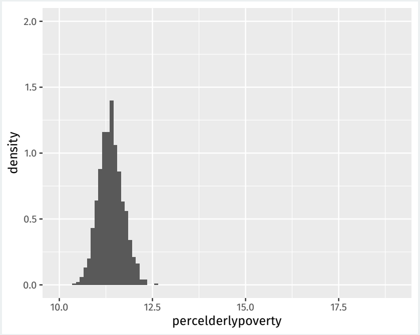
Population distributions:

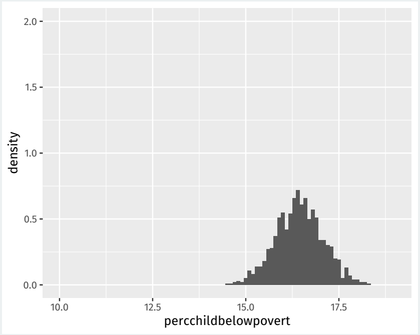




Midwest counties

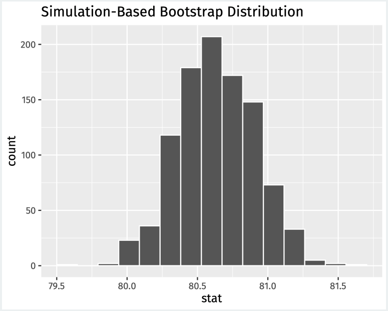
Sampling distributions with $n = 100$

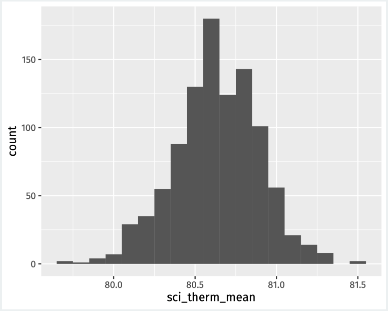


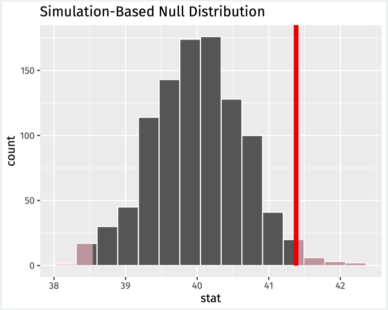


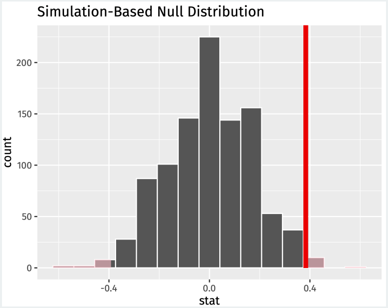
More population spread → higher SE

Similarity in the bootstrap/null distributions

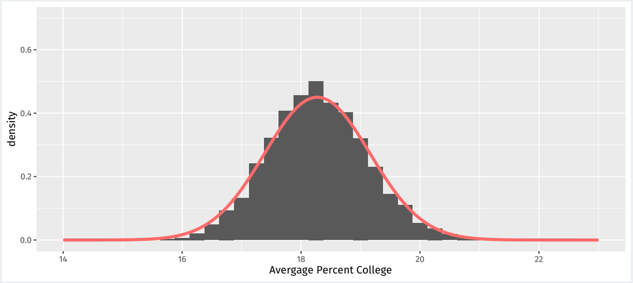








Conditions for the CLT

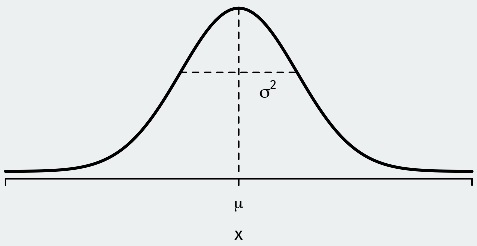


Central limit theorem: sums and means of **random samples** tend to be normally distributed as the **sample size grows**.

Many, many estimators will follow the CLT and have a normal distribution and will be easier to use this to do inference rather than doing increasingly complicated simulations.

2/ Normal distribution

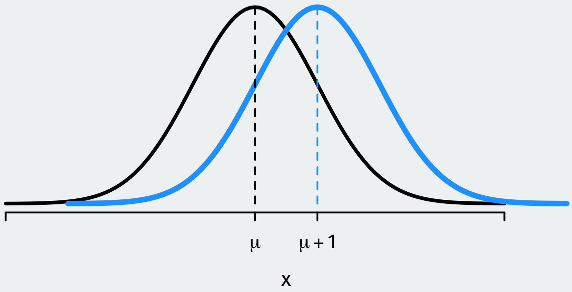
Normal distribution



- A normal distribution can be affect by two values:
    - **mean/expected value** usually written as μ
    - **variance** written as σ^2 (standard deviation is σ)
    - Written $X \sim N(\mu, \sigma^2)$.
- **Standard normal distribution**: mean 0 and standard deviation 1.

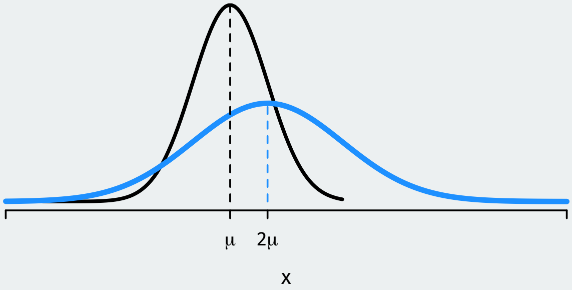
Reentering and scaling the normal

- How do transformations of a normal work?
- Let $X \sim N(\mu, \sigma^2)$ and c be a constant.
- If $Z = X + c$, then $Z \sim N(\mu + c, \sigma^2)$.
- Intuition: adding a constant to a normal shifts the distribution by that constant.



Recentering and scaling the normal

- Let $X \sim N(\mu, \sigma^2)$ and c be a constant.
- If $Z = cX$, then $Z \sim N(c\mu, (c\sigma)^2)$.
- Intuition: multiplying a normal by a constant scales the mean and the variance.



Z-scores of normals

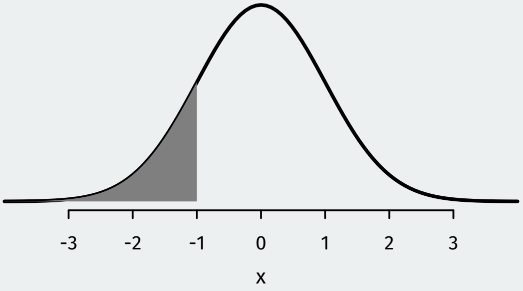
- These facts imply the **z-score** of a normal variable is a standard normal:

$$z = \frac{X - \mu}{\sigma} \sim N(0, 1)$$

- Subtract the mean and divide by the SD $\rightsquigarrow$ standard normal.
- z-score measures how many SDs away from the mean a value of X is.

Normal probability calculations

What's the probability of being below -1 for a standard normal?



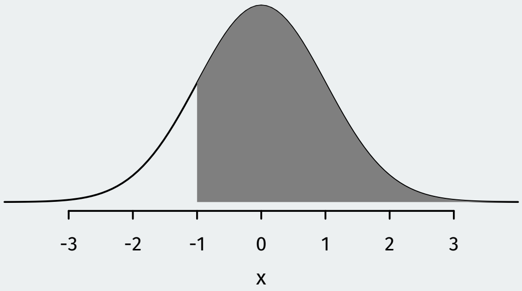
This is the area under the normal curve, which `pnorm()` function gives us this:

```
pnorm(-1, mean = 0, sd = 1)
```

```
## [1] 0.159
```

Normal probability calculations

What's the probability of being **above** -1 for a standard normal?



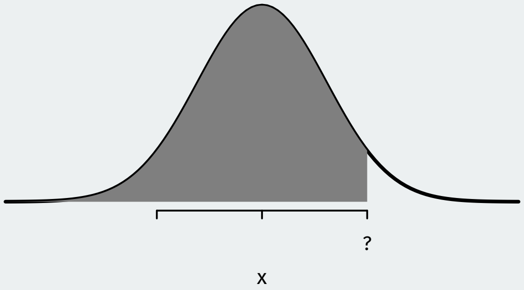
Total area under the curve (1) minus the area below -1:

```r
1 - pnorm(-1, mean = 0, sd = 1)
```

```
## [1] 0.841
```

Normal quantiles

What if we want to know the opposite? What value of the normal distribution puts 95% of the distribution below it?



This is a **quantile** and we can get it using qnorm( ):

```
qnorm(0.95, mean = 0, sd = 1)
```

```
## [1] 1.64
```

3/ Using the Normal for inference

How popular is Joe Biden?

- What proportion of the public approves of Biden's job as president?
- Latest Gallup poll:
    - Sept 1st-16th
    - 812 adult Americans
    - Telephone interviews
    - Approve (42%), Disapprove (56%)
- Define r.v. Y_i for Biden approval:
    - $Y_i = 1 \rightsquigarrow$ respondent i approves of Biden, 0 otherwise.
    - $p = \mathbb{P}(Y_i = 1)$ the population proportion of Biden approvers.
    - $\overline{Y} = 0.42$ is the sample proportion.

Standard errors for sample proportions

How variable will our sample proportion be? Depends on the **standard error**.

Special rule for SEs of sample proportion $\overline{Y}$:

$$SE \text{ for } \overline{Y} = \sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{(\text{pop. proportion}) \times (1 - \text{pop. proportion})}{\text{sample size}}}$$

Because we don't know p, we replace it with our best guess, $\overline{Y}$:

$$\widehat{SE} = \sqrt{\frac{\overline{Y}(1-\overline{Y})}{n}}$$

CLT for confidence intervals

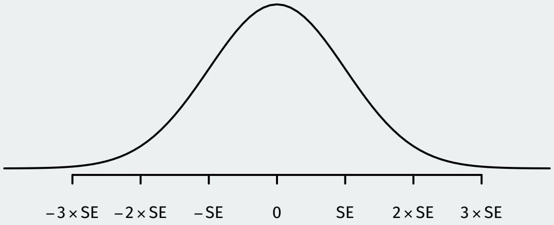
$$\overline{Y} - p = \text{chance error}$$

- How can we figure out a range of plausible chance errors?
  - Find a range of plausible chance errors and add them to $\overline{Y}$
  - With **bootstrap**, we used resampling to simulate chance error.
- Central limit theorem implies

$$\overline{Y} \approx N\left(p, \frac{p(1-p)}{n}\right)$$

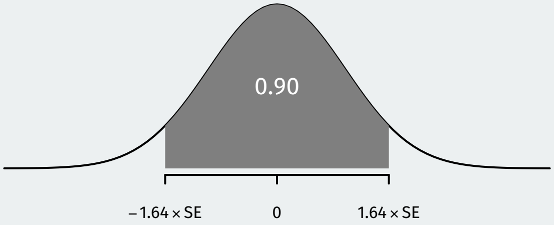
Chance error: $\overline{Y} - p$ is approximately normal with mean 0 and SE equal to $\sqrt{p(1-p)/n}$

Chance errors



If $\overline{Y} \sim N(p, SE^2)$, then chance errors are $\overline{Y} - p \sim N(0, SE^2)$ so:

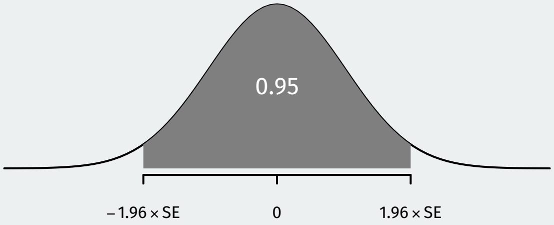
Chance errors



If $\overline{Y} \sim N(p, SE^2)$, then chance errors are $\overline{Y} - p \sim N(0, SE^2)$ so:

- ≈ 90% of chance errors $\overline{Y} - p$ are within 1.64 SEs of the mean.

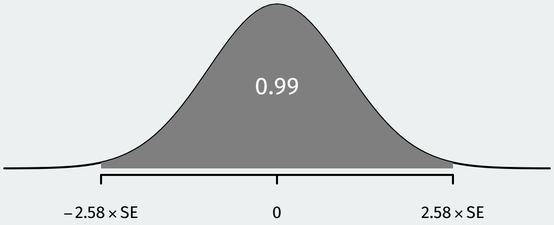
Chance errors



If $\overline{Y} \sim N(p, SE^2)$, then chance errors are $\overline{Y} - p \sim N(0, SE^2)$ so:

- ≈ 90% of chance errors $\overline{Y} - p$ are within 1.64 SEs of the mean.
- ≈ 95% of chance errors $\overline{Y} - p$ are within 1.96 SEs of the mean.

Chance errors



If $\overline{Y} \sim N(p, SE^2)$, then chance errors are $\overline{Y} - p \sim N(0, SE^2)$ so:

- ≈ 90% of chance errors $\overline{Y} - p$ are within 1.64 SEs of the mean.
- ≈ 95% of chance errors $\overline{Y} - p$ are within 1.96 SEs of the mean.
- ≈ 99% of chance errors $\overline{Y} - p$ are within 2.58 SEs of the mean.

This implies we can build a 95% confidence interval with $\overline{Y} \pm 1.96 \times SE$

How did we get those values?

- First, choose a **confidence level**.
    - What percent of chance errors do you want to count as “plausible”?
    - Convention is 95%.
- $100 \times (1 - \alpha)\%$ confidence interval:

$$CI = \overline{Y} \pm z_{\alpha/2} \times SE$$

- In polling, $\pm z_{\alpha/2} \times SE$ is called the **margin of error**
- $z_{\alpha/2}$ is the $N(0,1)$ z-score that would put $\alpha/2$ in the upper tail:
    - $\mathbb{P}(-z_{\alpha/2} < Z < z_{\alpha/2}) = \alpha$
    - 90% CI $\rightsquigarrow \alpha = 0.1 \rightsquigarrow z_{\alpha/2} = 1.64$
    - 95% CI $\rightsquigarrow \alpha = 0.05 \rightsquigarrow z_{\alpha/2} = 1.96$
    - 99% CI $\rightsquigarrow \alpha = 0.01 \rightsquigarrow z_{\alpha/2} = 2.58$

Standard normal z-scores in R

`qnorm(x, lower.tail = FALSE)` will find the quantile of $N(0, 1)$ that puts x in the upper tail:

```r
qnorm(0.05, lower.tail = FALSE)
```

```
## [1] 1.64
```

```r
qnorm(0.025, lower.tail = FALSE)
```

```
## [1] 1.96
```

```r
qnorm(0.005, lower.tail = FALSE)
```

```
## [1] 2.58
```

Gov 50: 24. More Inference with Mathematical Models

Matthew Blackwell

Harvard University

Roadmap

1. Confidence intervals for experiments
2. Hypothesis testing with the CLT
3. Two-sample tests

1/ Confidence intervals for experiments

Comparison between groups

- More interesting to compare across groups.
    - Differences in public opinion across groups
    - Difference between treatment and control groups.
- Bedrock of causal inference!

Social pressure experiment

- Back to the Social Pressure Mailer GOTV example.
  - Primary election in MI 2006
- Treatment group: postcards showing their own and their neighbors' voting records.
  - Sample size of treated group, $n_T = 360$ (artificially reducing sample size to highlight the math)
- Control group: received nothing.
  - Sample size of the control group, $n_C = 1890$

Outcomes

- Outcome: $Y_i = 1$ if i voted, 0 otherwise.
- Turnout rate (sample mean) in treated group, $\overline{Y}_T = 0.37$
- Turnout rate (sample mean) in control group, $\overline{Y}_C = 0.30$
- Estimated **average treatment effect**

$$\widehat{\text{ATE}} = \overline{Y}_T - \overline{Y}_C = 0.07$$

Inference for the difference

- Parameter: **population ATE** $\mu_T - \mu_C$
  - μ_T: Turnout rate in the population if everyone received treatment.
  - μ_C: Turnout rate in the population if everyone received control.
- Estimator: $\widehat{\text{ATE}} = \overline{Y}_T - \overline{Y}_C$

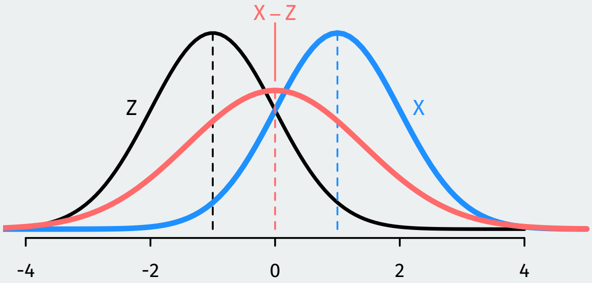
By the CLT in large samples, we know that:

- $\overline{Y}_T \approx N\left(\mu_T, \frac{\mu_T(1-\mu_T)}{n_C}\right)$
- $\overline{Y}_C \approx N\left(\mu_C, \frac{\mu_C(1-\mu_C)}{n_C}\right)$
- $\rightsquigarrow \overline{Y}_T - \overline{Y}_C \approx N(\mu_T - \mu_C, SE^2_{\text{diff}})$

But what is the SE_{diff} in this case?

Spread of a difference in normals

If we take the difference between two independent normal r.v.s, what happens to the spread?



The spread of the difference is **larger** than the spread of the two variables being differenced!

Standard error for the estimated ATE

- SE of a difference in means **adds** the SEs for each group

$$SE_{\text{diff}} = \sqrt{SE_T^2 + SE_C^2}$$

- Using what we know about SEs with binary outcomes:

$$SE_{\text{diff}} = \sqrt{\frac{\mu_T(1-\mu_T)}{n_t} + \frac{\mu_C(1-\mu_C)}{n_C}}$$

- Chance errors $\overline{Y}_T - \overline{Y}_C - (\mu_T - \mu_C) \approx N(0, SE_{\text{diff}})$
    - We can construct a 95% CI with $\widehat{\text{ATE}} \pm 1.96 \times SE_{\text{diff}}$

Confidence intervals

But we don't know μ_T or μ_C! Plug in our sample proportions to estimate the SE:

$$\begin{aligned}\widehat{\text{SE}}_{\text{diff}} &= \sqrt{\frac{\overline{Y}_T(1-\overline{Y}_T)}{n_t} + \frac{\overline{Y}_C(1-\overline{Y}_C)}{n_C}} \\ &= \sqrt{\frac{0.37 \times 0.63}{360} + \frac{0.3 \times 0.7}{1890}} = 0.028\end{aligned}$$

Now we can construct confidence intervals based on the CLT like last time:

$$\begin{aligned}CI_{95} =& \widehat{\text{ATE}} \pm 1.96 \times \widehat{\text{SE}}_{\text{diff}} \\ =& 0.07 \pm 1.96 \times 0.028 \\ =& 0.07 \pm 0.054 \\ =& [0.016, 0.124]\end{aligned}$$

Range of possibilities taking into account plausible chance errors.

2/ Hypothesis testing with the CLT

Statistical hypothesis testing

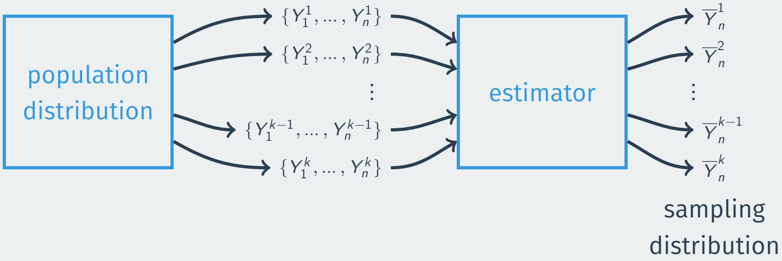
- Statistical hypothesis testing is a **thought experiment**.
- What would the world look like **if we knew the truth**?
- Conducted with several steps:

  1. Specify your **null** and **alternative hypotheses**
  2. Choose an appropriate **test statistic** and level of test α
  3. Derive the **reference distribution** of the test statistic under the null.
  4. Use this distribution to calculate the **p-value**.
  5. Use p-value to decide whether to reject the null hypothesis or not

How popular is Joe Biden?

- What proportion of the public approves of Biden's job as president?
- Latest Gallup poll: $\overline{Y} = 0.42$ with $n = 812$
- Could we reject the null that Biden's national support is 50%?
    - Null: $H_0 : p = 0.5$
    - Alternative: $H_1 : p \neq 0.5$

Sampling distribution, in pictures



CLT for hypothesis testing

Under the null, we know the distribution of $\overline{Y}$:

$$\overline{Y} \approx N\left(p, \frac{p(1-p)}{n}\right) = N\left(0.5, \frac{0.5 \times 0.5}{812}\right)$$

Using the rules of normal transformations if $X \sim N(\mu, \sigma^2)$:

$$\frac{X - \mu}{\sigma} \sim N(0, 1)$$

Then under the null, know the distribution of the following test statistic:

$$Z = \frac{\overline{Y} - 0.5}{0.5/\sqrt{812}} \approx N(0, 1)$$

p-values

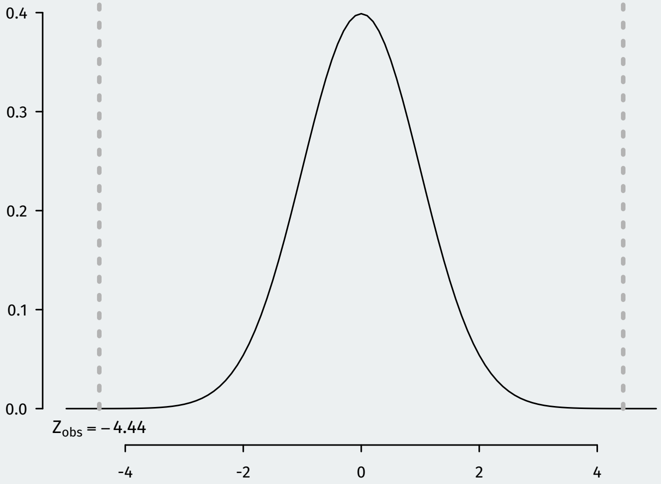
What we observe:

$$Z_{\text{obs}} = \frac{\overline{Y} - 0.5}{0.5/\sqrt{812}} = \frac{0.42 - 0.5}{0.5/\sqrt{812}}$$
$$= -\frac{0.08}{0.018} = -4.44$$

Our observed sample proportion is 4.44 SEs away from 0.5 under the null. What's the probability of being that far away? (**p-value**)

```
pnorm(-4.44, mean = 0, sd = 1) + ## prob being below -4.44
  (1 - pnorm(4.44, mean = 0, sd = 1)) ## prob being above 4.44
```

```
## [1] 0.000009
```


0.4
0.3
0.2
0.1
0.0
$Z_{obs} = -4.44$
-4
-2
0
2
4

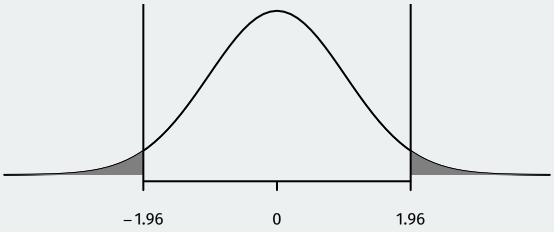
Generalizing hypothesis tests

- Hypothesis testing using the CLT pretty much takes this general form no matter what the estimator of interest is.
- Hypotheses: $H_0 : \mu = \mu_0$ (null guess), $H_1 : \mu \neq \mu_0$
- Test statistic:

$$Z = \frac{\text{observed value} - \text{null guess}}{\widehat{SE}} = \frac{\overline{Y} - \mu_0}{\widehat{SE}}$$

  - The exact estimator for the standard error $\widehat{SE}$ will depend on the estimator of interest.
- Null distribution: $Z \approx N(0, 1)$ by the CLT
- p-value: probability of a standard normal being bigger than $|Z_{\text{obs}}|$

Rejecting regions



- Reject if p-value is below α (usually 0.05).
    - We know 5% of the time Z will be bigger than 1.96.
    - If $Z_{\text{obs}} > 1.96$ or $Z_{\text{obs}} < -1.96$, then the p-value must be below 0.05
    - We can reject if $|Z_{\text{obs}}| > 1.96$

3/ Two-sample tests

Two-sample hypotheses

- Parameter: **population ATE** $\mu_T - \mu_C$
- Goal: learn about the population difference in means
- Usual null hypothesis: no difference in population means (ATE = 0)
    - Null: $H_0 : \mu_T - \mu_C = 0$
    - Two-sided alternative: $H_1 : \mu_T - \mu_C \neq 0$
- In words: are the differences in sample means just due to chance?

Difference-in-means review

- Sample turnout rates: $\overline{Y}_T = 0.37$, $\overline{Y}_C = 0.30$
- Sample sizes: $n_T = 360$, $n_C = 1890$
- Estimator is the **sample difference-in-means**:

$$\widehat{\text{ATE}} = \overline{Y}_T - \overline{Y}_C = 0.07$$

- Estimated SE for the difference in means:

$$\widehat{\text{SE}}_{\text{diff}} = \sqrt{\frac{\overline{Y}_T(1 - \overline{Y}_T)}{n_T} + \frac{\overline{Y}_C(1 - \overline{Y}_C)}{n_C}} = 0.028$$

CLT again and again

Earlier we saw that by the CLT we have:

$$\overline{Y}_T - \overline{Y}_C \approx N(\mu_T - \mu_C, \mathsf{SE}^2_{\text{diff}})$$

We can use Z-scores to get a test statistic:

$$Z = \frac{(\overline{Y}_T - \overline{Y}_C) - (\mu_T - \mu_C)}{\mathsf{SE}_{\text{diff}}} \sim N(0, 1)$$

Same general form of the test statistic as with one sample mean/proportion:

$$\frac{\text{observed - null guess}}{\text{SE}}$$

The usual null of no difference

- Null hypothesis: $H_0 : \mu_T - \mu_C = 0$
- Test statistic:

$$Z = \frac{(\overline{Y}_T - \overline{Y}_C) - (\mu_T - \mu_C)}{\text{SE}_{\text{diff}}} = \frac{(\overline{Y}_T - \overline{Y}_C) - 0}{\text{SE}_{\text{diff}}}$$

- In large samples, we can replace true SE with an estimate:

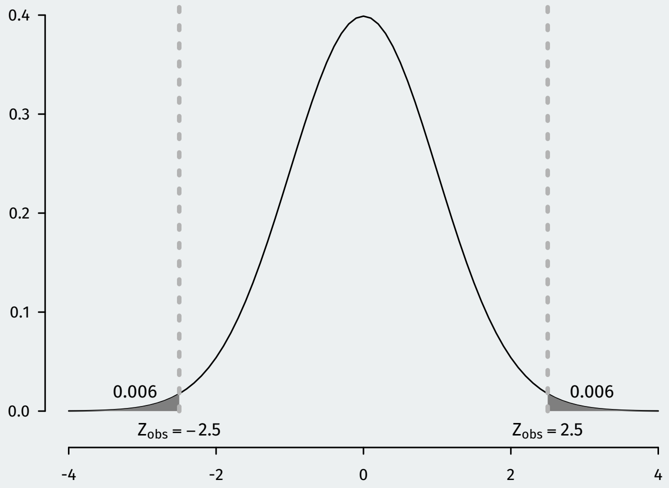
$$\widehat{\text{SE}}_{\text{diff}} = \sqrt{\widehat{\text{SE}}_T^2 + \widehat{\text{SE}}_C^2}$$

Calculating p-values

- Finally! Our test statistic in this sample:

$$Z = \frac{\overline{Y}_T - \overline{Y}_C}{\widehat{\text{SE}}_{\text{diff}}} = \frac{0.07}{0.028} = 2.5$$

- p-value based on a two-sided test: probability of getting a difference in means this big (or bigger) if the null hypothesis were true
    - Lower p-values $\rightsquigarrow$ stronger evidence against the null.



```r
2 * pnorm(2.5, lower.tail = FALSE)
```

```
## [1] 0.0124
```

Gov 50: 25. Inference for Linear Regression

Matthew Blackwell

Harvard University

Roadmap

1. Inference for linear regression
2. Presenting OLS regressions
3. Wrapping up the class

1/ Inference for linear regression

Data

- Do political institutions promote economic development?
  - Famous paper on this: Acemoglu, Johnson, and Robinson (2001)
  - Relationship between strength of property rights in a country and GDP.
- Data:

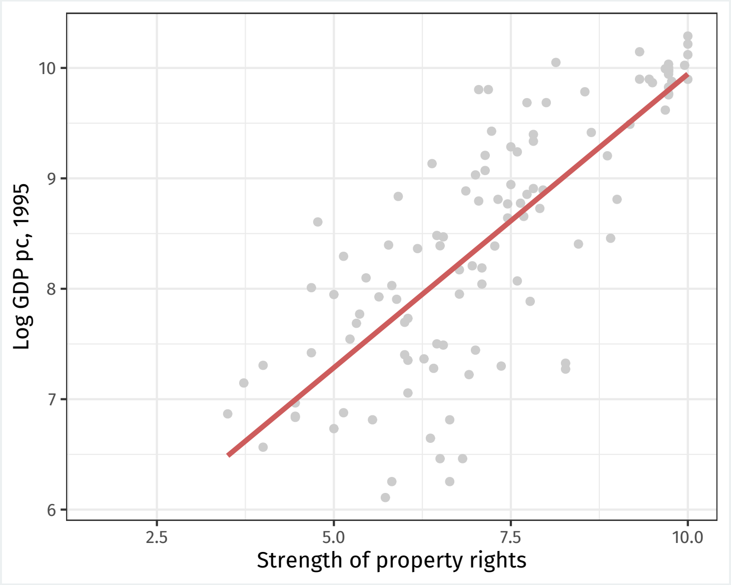
Name	Description
`shortnam`	three-letter country code
`africa`	indicator for if the country is in Africa
`asia`	indicator for if country is in Asia
`avexpr`	strength of property rights (protection against expropriation)
`logpgp95`	log GDP per capita

Loading the data

```r
library(gov50data)
head(ajr)
```

```
## # A tibble: 6 x 15
##   short~1 africa lat_a~2 malfa~3 avexpr logpg~4 logem4  asia
##   <chr>    <dbl>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl> <dbl>
## 1 AFG          0   0.367 0.00372 NA      NA       4.54     1
## 2 AGO          1   0.137 0.950     5.36    7.77   5.63     0
## 3 ARE          0   0.267 0.0123    7.18    9.80  NA        1
## 4 ARG          0   0.378 0         6.39    9.13   4.23     0
## 5 ARM          0   0.444 0        NA       7.68  NA        1
## 6 AUS          0   0.300 0         9.32    9.90   2.15     0
## # ... with 7 more variables: yellow <dbl>, baseco <dbl>,
## #   leb95 <dbl>, imr95 <dbl>, meantemp <dbl>,
## #   lt100km <dbl>, latabs <dbl>, and abbreviated variable
## #   names 1: shortnam, 2: lat_abst, 3: malfal94,
## #   4: logpgp95
```

AJR scatterplot



Simple linear regression model

- We are going to assume a linear model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

- Data:
  - Dependent variable: Y_i
  - Independent variable: X_i
- Population parameters:
  - Population intercept: β_0
  - Population slope: β_1
- Error/disturbance: ϵ_i
  - Represents all unobserved error factors influencing Y_i other than X_i.

Least squares

- How do we figure out the best line to draw?
    - Alt question: how do we figure out β_0 and β_1?
    - $(\hat{\beta}_0, \hat{\beta}_1)$: estimated coefficients.
    - $\widehat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$: predicted/fitted value.
    - $\hat{\epsilon}_i = Y_i - \widehat{Y}$: residual.
- Get these estimates by the **least squares method**.
- Minimize the **sum of the squared residuals** (SSR):

$$\text{SSR} = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \hat{\beta}_0 - \hat{\beta}_1 X_i)^2$$

Estimators

- Least squares is an **estimator**
  - it's a machine that we plug data into and we get out estimates.

$$\begin{tikzcd}
\text{Sample 1: } \{(Y_1, X_1), \dots, (Y_n, X_n)\} \arrow[ddr] & & (\hat{\beta}_0, \hat{\beta}_1)_1 \\
\text{Sample 2: } \{(Y_1, X_1), \dots, (Y_n, X_n)\} \arrow[dr] & & (\hat{\beta}_0, \hat{\beta}_1)_2 \\
\vdots & \boxed{\text{OLS}} \arrow[uur] \arrow[ur] \arrow[dr] \arrow[ddr] & \vdots \\
\text{Sample } k-1\text{: } \{(Y_1, X_1), \dots, (Y_n, X_n)\} \arrow[ur] & & (\hat{\beta}_0, \hat{\beta}_1)_{k-1} \\
\text{Sample } k\text{: } \{(Y_1, X_1), \dots, (Y_n, X_n)\} \arrow[uur] & & (\hat{\beta}_0, \hat{\beta}_1)_k
\end{tikzcd}$$

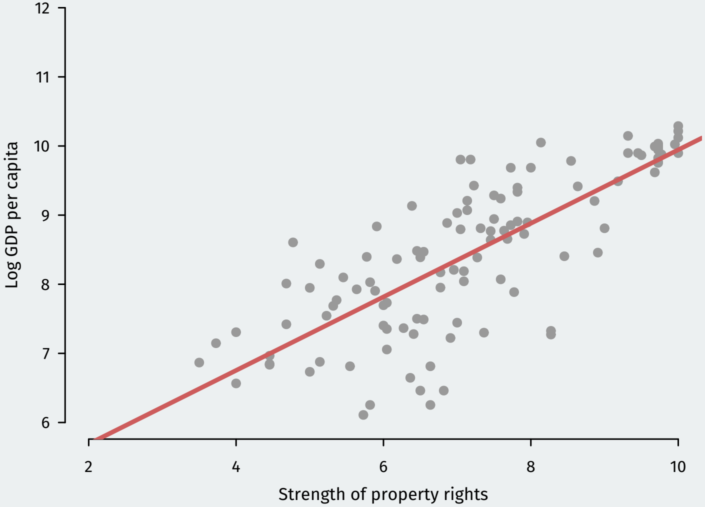
- Just like the sample mean or difference in sample means
- $\rightsquigarrow$ sampling distribution with a standard error, etc.

Simulation procedure

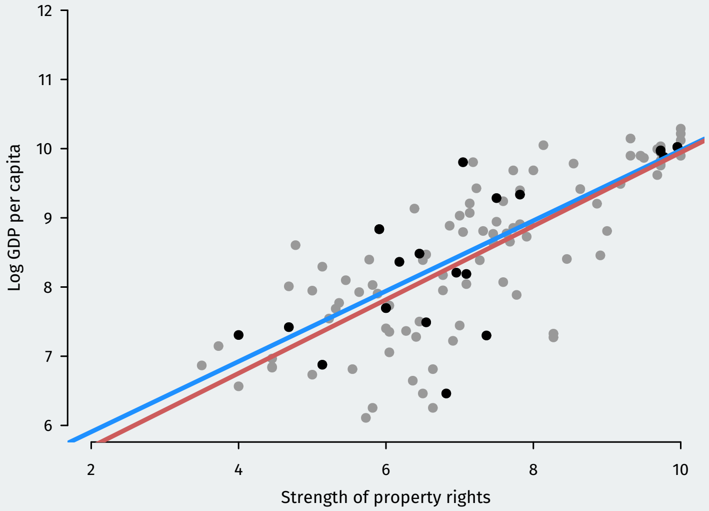
- Let's take a simulation approach to demonstrate:
    - Pretend that the AJR data represents the population of interest
    - See how the line varies from sample to sample

1. Randomly sample $n = 30$ countries w/ replacement using `sample()`
2. Use `lm()` to calculate the OLS estimates of the slope and intercept
3. Plot the estimated regression line

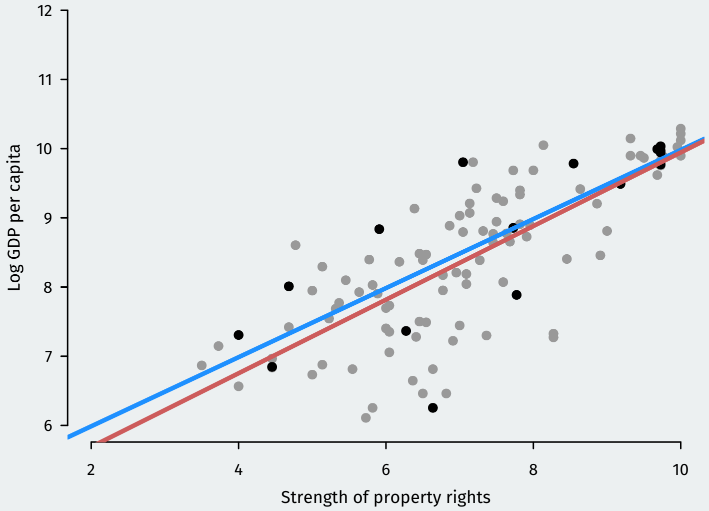
Population regression



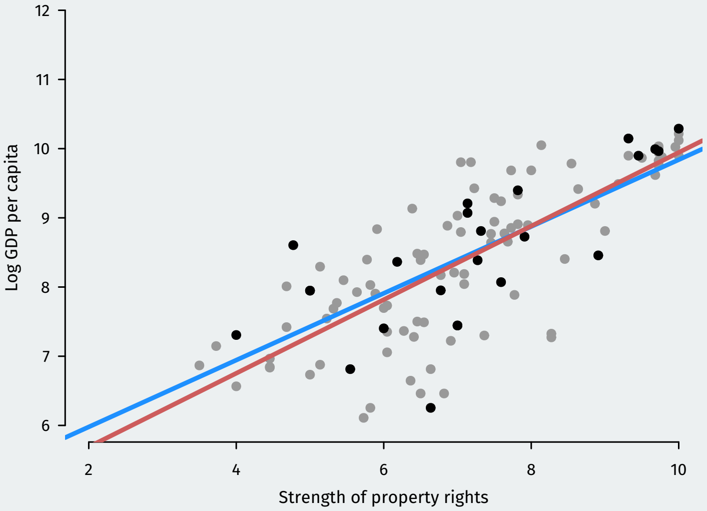
Randomly sample from AJR



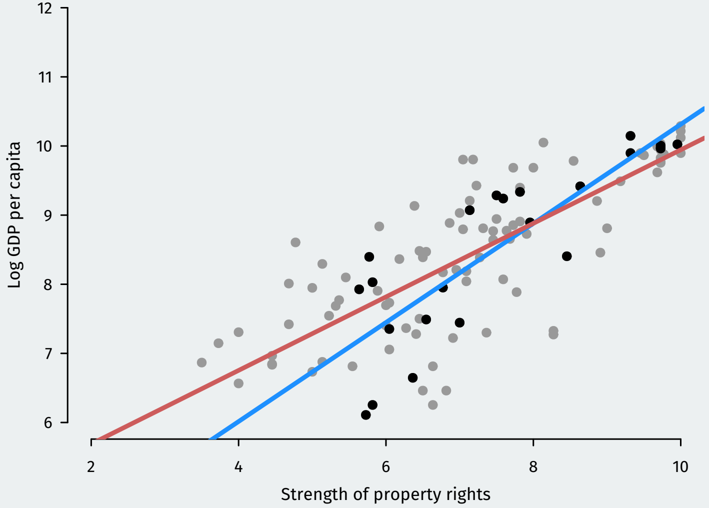
Randomly sample from AJR



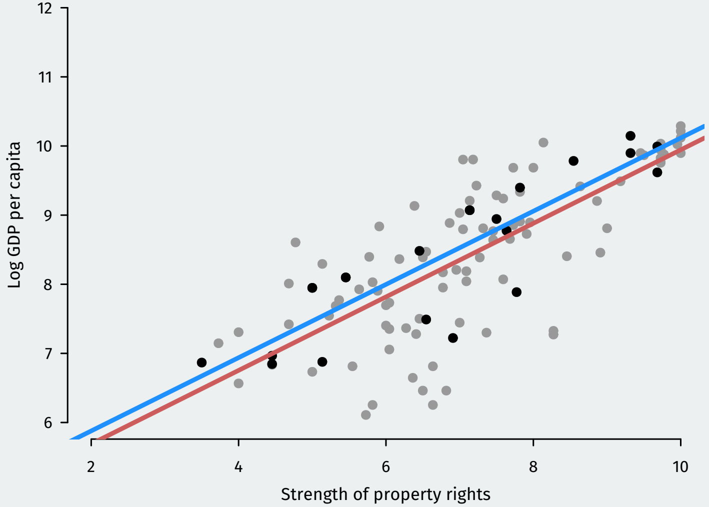
Randomly sample from AJR



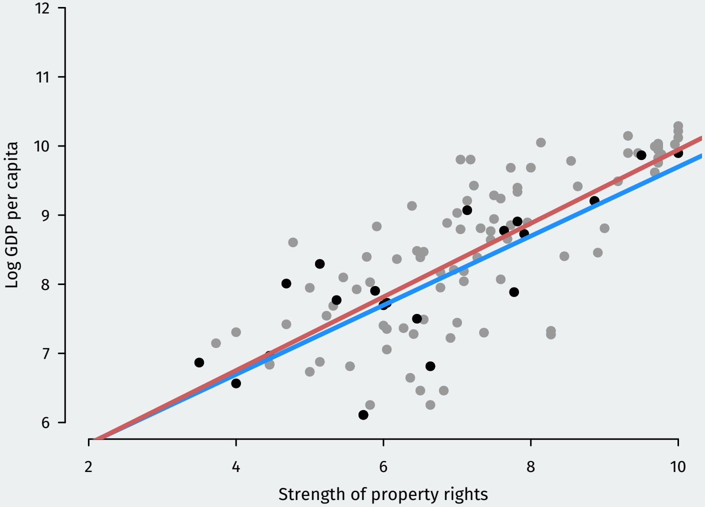
Randomly sample from AJR



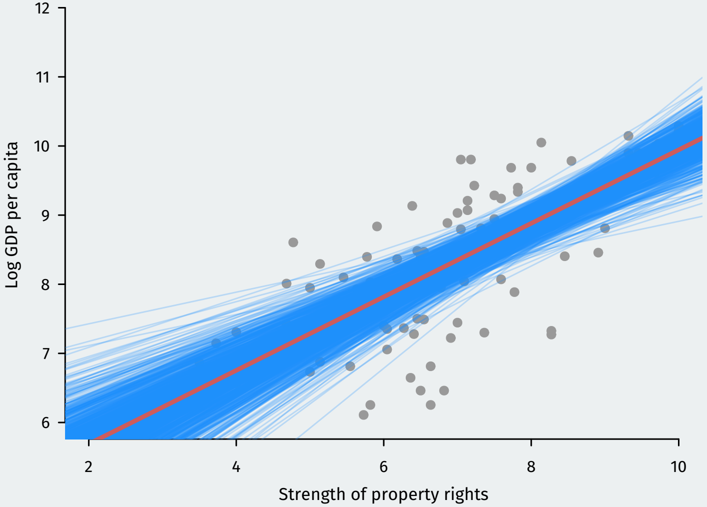
Randomly sample from AJR



Randomly sample from AJR

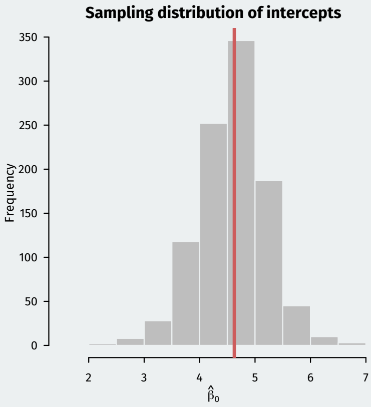


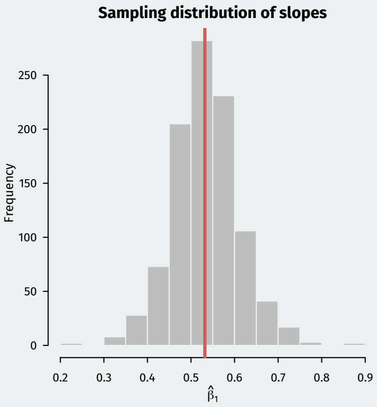
Randomly sample from AJR



Sampling distribution of OLS

- Estimated slope and intercept vary between samples, centered on truth.





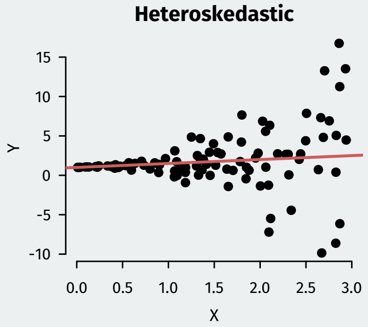
Properties of OLS

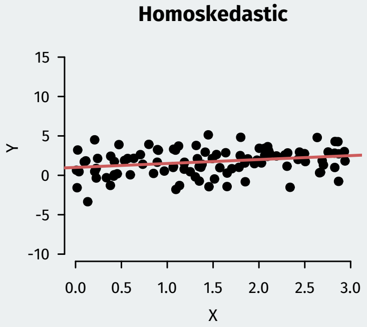
- $\hat{\beta}_0$ and $\hat{\beta}_1$ are random variables
  - Are they on average equal to the true values (bias)?
  - How spread out are they around their center (variance)?
- Under minimal conditions, $\hat{\beta}_0$ and $\hat{\beta}_1$ are unbiased for the population line of best fit, but...
  - This might be misleading if the true relationship is nonlinear.
  - May not represent a causal effect unless causal assumptions hold.

Standard errors of OLS

R will also calculate an estimate of the standard error: $\widehat{\mathrm{SE}}(\hat{\beta}_1)$

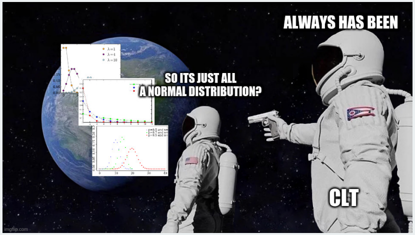
Default estimators for the SEs assume **homoskedasticity** or that the spread around the regression line is the same for all values of the independent variables.





Relatively easy fixes exist, but beyond the scope of this class.

Tests and CIs for regression



- $(\hat{\beta}_0, \hat{\beta}_1)$ can be written as weighted averages of the outcome...
  - Which means they follow the Central Limit Theorem!
- BAM! 95% confidence intervals: $\hat{\beta}_1 \pm 1.96 \times \widehat{\text{SE}}(\hat{\beta}_1)$
- BOOM! Hypothesis tests:
  - Null hypothesis: $H_0 : \beta_1 = \beta_1^*$
  - Test statistic: $\frac{\hat{\beta}_1 - \beta_1^*}{\widehat{\text{SE}}(\hat{\beta}_1)} \sim N(0, 1)$
  - Usual test is of $\beta_1 = 0$.
  - $\hat{\beta}_1$ is **statistically significant** if its p-value from this test is below some threshold (usually 0.05)

```r
ajr.reg <- lm(logpgp95 ~ avexpr, data = ajr)
summary(ajr.reg)
```

```
##
## Call:
## lm(formula = logpgp95 ~ avexpr, data = ajr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.902 -0.316  0.138  0.422  1.441
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.6261     0.3006    15.4   <2e-16 ***
## avexpr        0.5319     0.0406    13.1   <2e-16 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.718 on 109 degrees of freedom
##   (52 observations deleted due to missingness)
## Multiple R-squared:  0.611,  Adjusted R-squared:  0.608
## F-statistic:  171 on 1 and 109 DF,  p-value: <2e-16
```

Using broom with regression

```r
library(broom)
tidy(ajr.reg)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    4.63     0.301       15.4 4.28e-29
## 2 avexpr         0.532    0.0406      13.1 4.16e-24
```

Multiple regression

- Correlation doesn't imply causation
- Omitted variables $\rightsquigarrow$ violation of exogeneity
- You can adjust for multiple confounding variables:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \cdots + \beta_p X_{ip} + \epsilon_i$$

- Interpretation of β_j: an increase in the outcome associated with a one-unit increase in X_{ij} when other variables don't change their values
- Inference:
  - Confidence intervals constructed exactly the same for $\hat{\beta}_j$
  - Hypothesis tests done exactly the same for $\hat{\beta}_j$
  - $\rightsquigarrow$ interpret p-values the same as before.

Using `knitr::kable` to produce tables

```r
ajr.multreg <- lm(logpgp95 ~ avexpr + lat_abst + asia + africa, data = ajr)
tidy(ajr.multreg) |>
  knitr::kable(digits = 3)
```

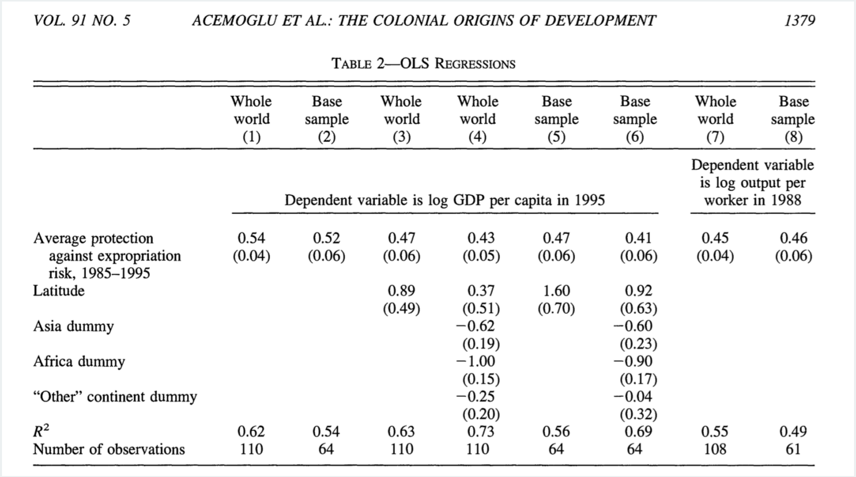
term	estimate	std.error	statistic	p.value
(Intercept)	5.840	0.339	17.239	0.000
avexpr	0.394	0.050	7.843	0.000
lat_abst	0.312	0.444	0.703	0.484
asia	-0.170	0.153	-1.108	0.270
africa	-0.930	0.165	-5.628	0.000

2/ Presenting OLS regressions

Regression tables

- In papers, you'll often find regression tables that have several models.
- Each column is a different regression:
    - Might differ by independent variables, dependent variables, sample, etc.
- Standard errors, p-values, sample size, and R^2 may be reported as well.

AJR regression table



VOL. 91 NO. 5 ACEMOGLU ET AL.: THE COLONIAL ORIGINS OF DEVELOPMENT 1379

TABLE 2—OLS REGRESSIONS

	Whole world (1)	Base sample (2)	Whole world (3)	Whole world (4)	Base sample (5)	Base sample (6)	Whole world (7)	Base sample (8)
	Dependent variable is log GDP per capita in 1995						Dependent variable is log output per worker in 1988	
Average protection against expropriation risk, 1985–1995	0.54	0.52	0.47	0.43	0.47	0.41	0.45	0.46
	(0.04)	(0.06)	(0.06)	(0.05)	(0.06)	(0.06)	(0.04)	(0.06)
Latitude			0.89	0.37	1.60	0.92		
			(0.49)	(0.51)	(0.70)	(0.63)		
Asia dummy				−0.62		−0.60		
				(0.19)		(0.23)		
Africa dummy				−1.00		−0.90		
				(0.15)		(0.17)		
"Other" continent dummy				−0.25		−0.04		
				(0.20)		(0.32)		
R^2	0.62	0.54	0.63	0.73	0.56	0.69	0.55	0.49
Number of observations	110	64	110	110	64	64	108	61

`modelsummary()` to produce tables

We can use `modelsummary()` to produce a table. It takes a list of outputs from `lm` and aligns them in the correct way.

```
modelsummary::modelsummary(list(ajr.reg, ajr.multreg))
```

Output

```r
modelsummary::modelsummary(list(ajr.reg, ajr.multreg))
```

	Model 1	Model 2
(Intercept)	4.626	5.840
	(0.301)	(0.339)
avexpr	0.532	0.394
	(0.041)	(0.050)
lat_abst		0.312
		(0.444)
asia		−0.170
		(0.153)
africa		−0.930
		(0.165)
Num.Obs.	111	111
R2	0.611	0.713
R2 Adj.	0.608	0.703
AIC	245.4	217.6
BIC	253.5	233.8
Log.Lik.	−119.709	−102.795
RMSE	0.71	0.61

Cleaning up the goodness of fit statistics

```r
modelsummary::modelsummary(
  list(ajr.reg, ajr.multreg),
  gof_map = c("nobs", "r.squared", "adj.r.squared"))
```

	Model 1	Model 2
(Intercept)	4.626	5.840
	(0.301)	(0.339)
avexpr	0.532	0.394
	(0.041)	(0.050)
lat_abst		0.312
		(0.444)
asia		−0.170
		(0.153)
africa		−0.930
		(0.165)
Num.Obs.	111	111
R2	0.611	0.713
R2 Adj.	0.608	0.703

Cleaning up the variable names

We can also map the variable names to more readable names using the `coef_map` argument. But first, we should do the mapping in a vector. Any term omitted from this vector will be omitted from the table

```
var_labels <- c(
  "avexpr" = "Avg. Expropriation Risk",
  "lat_abst" = "Abs. Value of Latitude",
  "asia" = "Asian country",
  "africa" = "African country"
)
var_labels
```

```
##                    avexpr                  lat_abst
## "Avg. Expropriation Risk"  "Abs. Value of Latitude"
##                      asia                    africa
##           "Asian country"         "African country"
```

Nice table

```r
modelsummary::modelsummary(
  list(ajr.reg, ajr.multreg),
  coef_map = var_labels,
  gof_map = c("nobs", "r.squared", "adj.r.squared"))
```

	Model 1	Model 2
Avg. Expropriation Risk	0.532	0.394
	(0.041)	(0.050)
Abs. Value of Latitude		0.312
		(0.444)
Asian country		−0.170
		(0.153)
African country		−0.930
		(0.165)
Num.Obs.	111	111
R2	0.611	0.713
R2 Adj.	0.608	0.703

3/ Wrapping up the class

Big takeaways

Important takeaways from the course:

1. Data wrangling and data visualizations are really important skills that you now have!
2. Causality is hugely important in the world but difficult to establish.
3. Really important to understand and assess statistical uncertainty when working with data.

I’m really proud of you!

You’ve come a long way! Hopefully the tools you learned in this course will help you throughout your life and career!

What next?

If you end your training now, if you choose the quick and easy path, as Vader did, you will become an agent of evil.

- Gov 51 with Naijia Liu:
    - A more in-depth review of some ideas from Gov 50 including causality and regression plus new models (maybe some machine learning).
    - Really helpful for students looking to write senior theses.
- Only need 3 more classes to finish the data science track in Gov!
- More theoretical stats side: Stat 110/111
- More CS approach to data science: CS109 (Data Science 1)

Thanks!

Fill out your evaluations!