

Gov 50: 1. Introduction

Matthew Blackwell

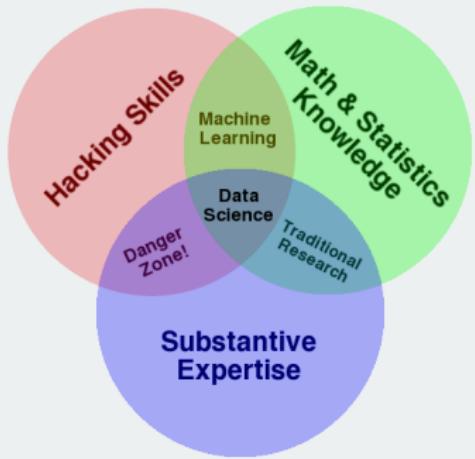
Harvard University

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.

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

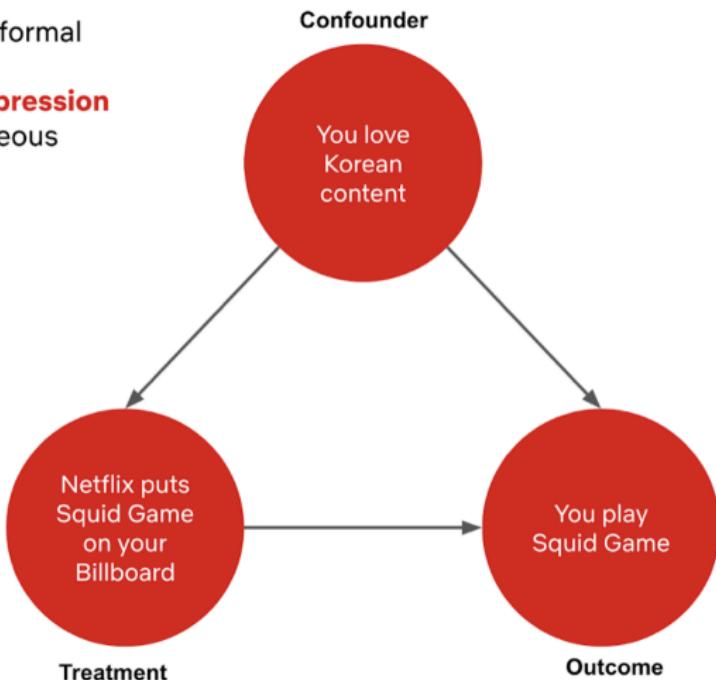


A COMMUTER BOARDS A BAY AREA RAPID TRANSIT (BART) TRAIN IN THE NEW MONTGOMERY STATION IN SAN FRANCISCO, CALIFORNIA, AS SEEN IN MARCH 2022. (PHOTOGRAPHER: DAVID PAUL MORRIS—BLOOMBERG/GETTY IMAGES)

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**)



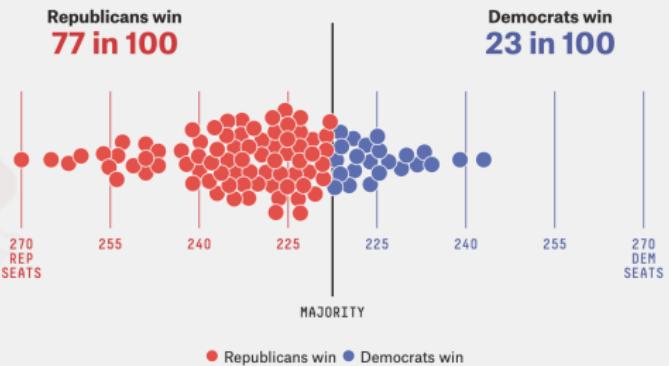
Compared to machine learning, causal inference allows us to build a robust framework that controls for confounders in order to estimate the true incremental impact to members

Prediction

UPDATED 17 MINUTES AGO

Republicans are *favored* to win the House

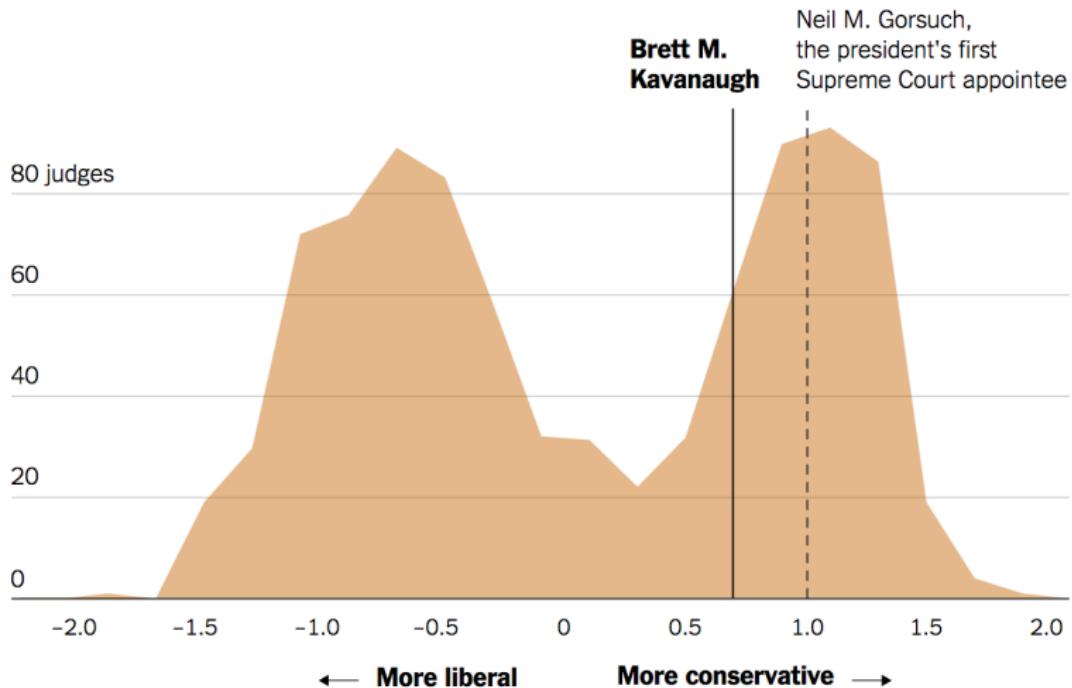
The Deluxe version of our model simulates the election 40,000 times to see which party wins the House most often. This sample of 100 outcomes gives you an idea of the range of scenarios the model considers possible.



We use numbers to express uncertainty. Upset wins are surprising but not impossible.

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

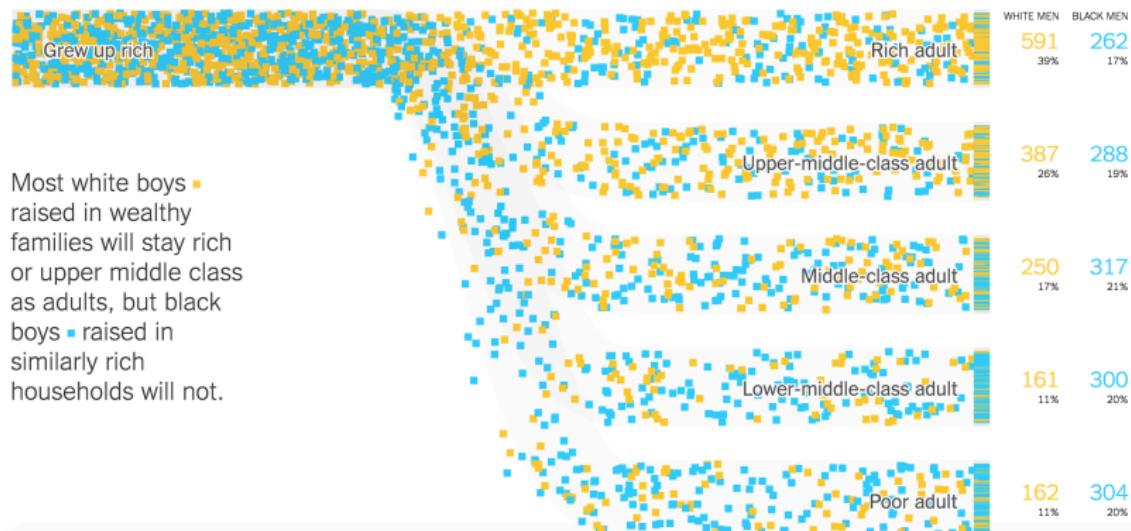
SHARE

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:



Adult outcomes reflect household incomes in 2014 and 2015.

Making government work better

The screenshot shows the CityScore website. At the top, there's a navigation bar with links for 'CITY OF BOSTON' (Mayor Martin J. Walsh), 'PUBLIC NOTICES', 'PAY AND APPLY', 'FEEDBACK', and 'TRANSLATE'. Below the navigation is a large graphic with the word 'CITY SCORE' in large letters, with a small registered trademark symbol. The 'O' in 'SCORE' is replaced by a circular seal featuring a building and text. Below this graphic, the word 'CITYSCORE' is written in a bold, sans-serif font. A sub-headline below it reads: '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.' Underneath this, there's a section titled 'THE SCORE' with a horizontal scale from 0.0 to 1.0. The current score is 1.02, indicated by a central dot on the scale. Arrows on either side of the scale point left and right, labeled with '<' and '>'. Below the scale, explanatory text states: 'Scores below 1.0 indicate that performance is below the target.' and 'Scores above 1.0 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
MISSSED 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

Who Gets Miscounted In The Census ?

Black

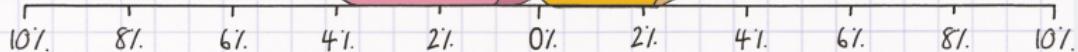


Hispanic

Native
American

Asian

White

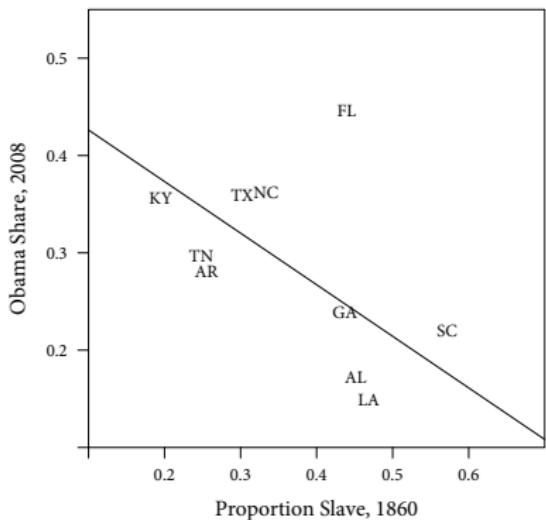


%. NOT COUNTED

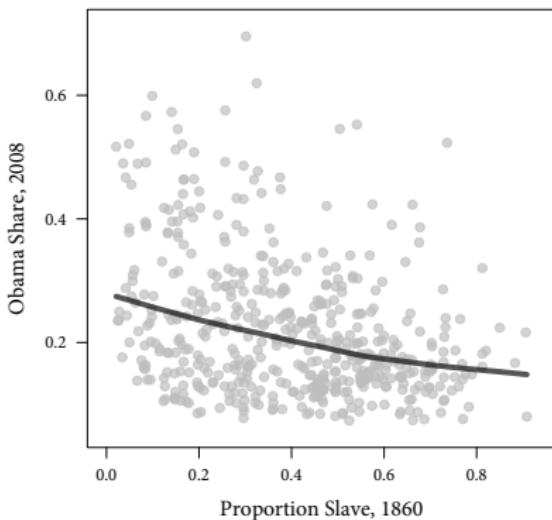
%. DOUBLE COUNTED OR FOUND

Understanding how the past matters

States

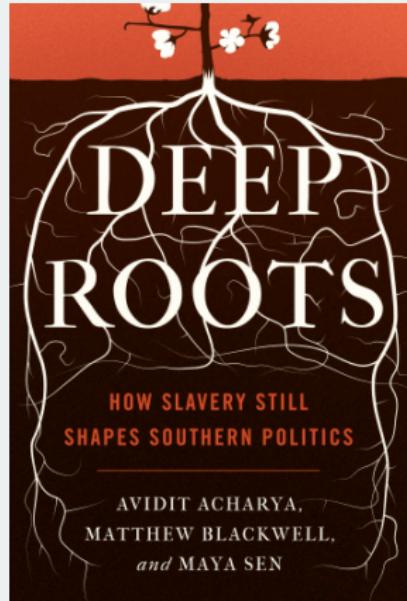
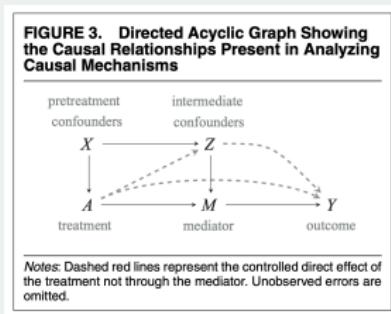


Counties



2/ Course Details

About me



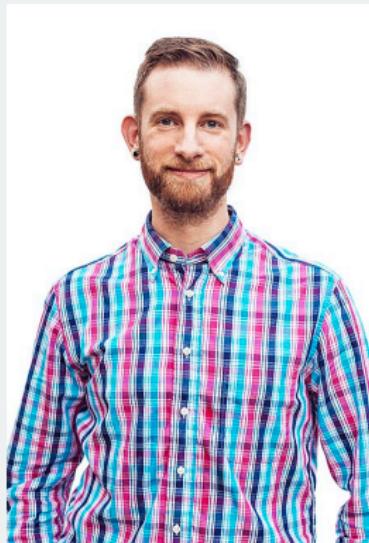
What will you learn in this class?

- 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!

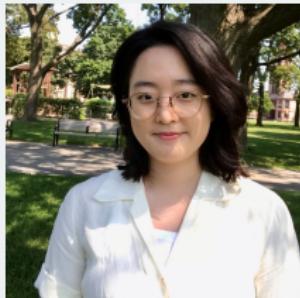
Teaching fellows



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.

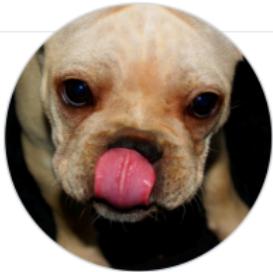
COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE 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 manageable.
- 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



Mara Averick
batpigandme

[Follow](#)

At 742 followers · 136 following

[@tidyverse](#)
[Missoula, MT](#)
[@dataandme](#)

Achievements



Beta [Send feedback](#)

Overview Repositories 227 Projects Packages Stars 808

batpigandme / README.ind

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 @sdraa's Night Owl VS Code theme...

1 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

The screenshot shows the 'ed' application interface. On the left, there's a sidebar with a 'New Thread' button, a 'COURSES' section containing 'GOV 50' (selected), 'Gov JLR', 'Drafts' (1), and 'Scheduled', and a 'CATEGORIES' section with 'General', 'Lectures', 'Sections', 'Problem Sets', 'Assignments', and 'Social'. The main area has a search bar and a 'Welcome!' thread from 'General' by 'Matt Blackwell' with 2w ago. The right side shows a 'New Post' screen with tabs for 'Question', 'Post' (selected), and 'Announcement'. It includes fields for 'Title', 'Category' (General, Lectures, Sections, Problem Sets, Assignments, Social), and a rich text editor toolbar. Below the toolbar, it says 'I can use markdown and insert code easily:' with a code block example. At the bottom, there are checkboxes for 'Pinned', 'Private', 'Anonymous', 'Anonymous Comments', and 'Megathread', along with a 'Draft saved' message and a 'Post' button.

ed GOV 50 – Ed Discussion

New Thread

COURSES

GOV 50

Gov JLR

Drafts 1

Scheduled

CATEGORIES

- General
- Lectures
- Sections
- Problem Sets
- Assignments
- Social

Search

Welcome!

General Matt Blackwell 2w

Cancel

New Post

Question Post Announcement

Title

Category General Lectures Sections Problem Sets Assignments Social

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

```
mean(some_data)
```

Pinned Keep at top of thread list

Private Visible to you and staff only

Anonymous Hide your name from students

Anonymous Comments Allow anonymous comments

Megathread Resolvable comments

Draft saved

Post

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

The Plain Person's Guide

~/>_

to Plain Text Social Science

Kieran Healy

- 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 intrinsically
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 outputs 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 programmatically 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 'licence()' 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.  
? |
```



RStudio interface showing a project named "cars-project".

Project View:

- File: cars-project.Rmd
- Knit on Save
- Knit
- Run
- Environment
- History
- Connections
- Tutorial
- Import Dataset (158 MiB)
- Global Environment

The Global Environment pane shows "Environment is empty".

File Explorer:

- New Folder
- New Blank File
- Delete
- Rename
- More

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		

Console:

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

cars-project - RStudio

cars-project

cars-project.Rmd x Knit on Save Knit Run Outline

Source Visual

Environment History Connections Tutorial Import Dataset 158 MiB List Global Environment

Environment is empty

Write notes, paper in Rmarkdown

```
1 - ---
2 #title: "Car Project"
3 author: "Matthew Blackwell"
4 date: "2022-09-06"
5 output: pdf_document
6 -
7
8 ````{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 -
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
17
18 ````{r cars}
19 summary(cars)
20 -
21 Car Project
```

R Markdown

Files Plots Packages Help Viewer Presentation

New Folder New Blank File Delete Rename More

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		

Console Background jobs

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
✓ tidyverse 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()
```

cars-project - RStudio

cars-project.Rmd x Knit on Save Knit Run Addins Environment History Connections Tutorial Import Dataset 158 MiB Global Environment List Environment is empty

```
1 - ---
2 title: "Car Project"
3 author: "Matthew Blackwell"
4 date: "2022-09-06"
5 output: pdf_document
6 - ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 -
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
17
18 ```{r cars}
19 summary(cars)
20 ```
```

R Markdown

Console Background jobs

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

Files Plots Packages Help Viewer Presentation

New Folder New Blank File Delete Rename More

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		

cars-project - RStudio

cars-project.Rmd x Knit on Save Knit Run Addins Outline Source Visual

```
1 - ---
2 title: "Car Project"
3 author: "Matthew Blackwell"
4 date: "2022-09-06"
5 output: pdf_document
6 - ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ---
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:
17
18 ```{r cars}
19 summary(cars)
20 ---
21 Car Project R Markdown
```

Environment History Connections Tutorial Import Dataset 158 MB Global Environment

Environment is empty

Files Plots Packages Help Viewer Presentation

New Folder New Blank File Delete Rename More

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		

Console Background jobs

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()
```

Project files, plots, and help

cars-project - RStudio

cars-project.Rmd x Knit on Save Knit Run Addins Outline Source Visual

```
1 - ---  
2 title: "Car Project"  
3 author: "Matthew Blackwell"  
4 date: "2022-09-06"  
5 output: pdf_document  
6 - ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr:::opts_chunk$set(echo = TRUE)  
10 - ---  
11  
12 ## R Markdown  
13  
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.  
15  
16 When you click the Knit button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:  
17  
18 ```{r cars}  
19 summary(cars)  
20 ```  
2:1 □ Car Project R Markdown
```

Console Background Jobs

```
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  
✓ tidyv 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()  
>  
>  
>  
>  
>
```

Environment History Connections Tutorial Import Dataset 158 MiB Global Environment List

Environment is empty

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

Files Plots Packages Help Viewer Presentation New Folder New Blank File Delete Rename More 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

3/ Using Rmarkdown

The acts of coding

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

Figure: 1. Writing code

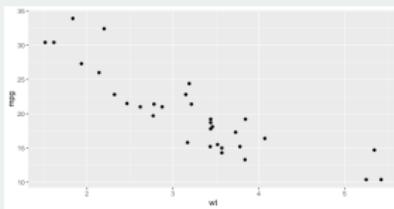


Figure: 2. Looking at output

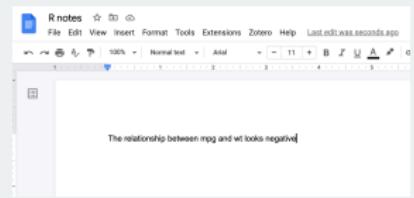


Figure: 3. Taking notes

How to do all of these efficiently?

Rmarkdown files to the rescue

```
notes.Rmd x
Source Visual
1 # ...
2 title: "Car Project"
3 output: pdf_document
4 date: "2022-09-04"
5 ...
6
7 ```{r setup, include=FALSE}
8 knitr::opts_chunk$set(echo = TRUE)
9 library(ggplot2)
10 ...
11
12
13
14 Now I will produce a scatterplot of car weight against
15 mileage per gallon:
16 ...
17 ```{r}
18 ggplot(mtcars, aes(x = wt, y = mpg)) +
19   geom_point()
20 ...
21 As we can see, this relationship is negative.
```

Figure: Rmarkdown file

Keep code and notes
together in plain text



Figure: Knit in R

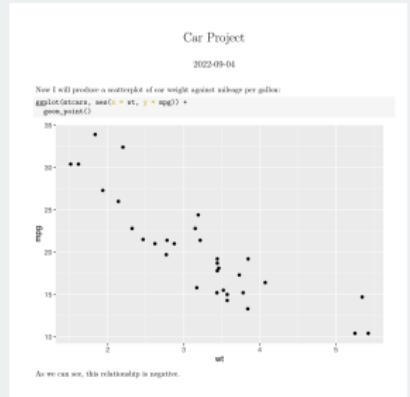


Figure: PDF output

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_
**bolds** 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: 

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**
superscript²
~~strikethrough~~
[link](#)

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi * r^2$
image: ![]



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

General

Code

Console

Appearance

Pane Layout

Packages

R Markdown

Python

Sweave

Spelling

Git/SVN

Publishing

Terminal

Accessibility

Basic Graphics Advanced

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:

History

Always save history (even when not saving .RData)

Remove duplicate entries in history

Other

Wrap around when navigating to previous/next tab

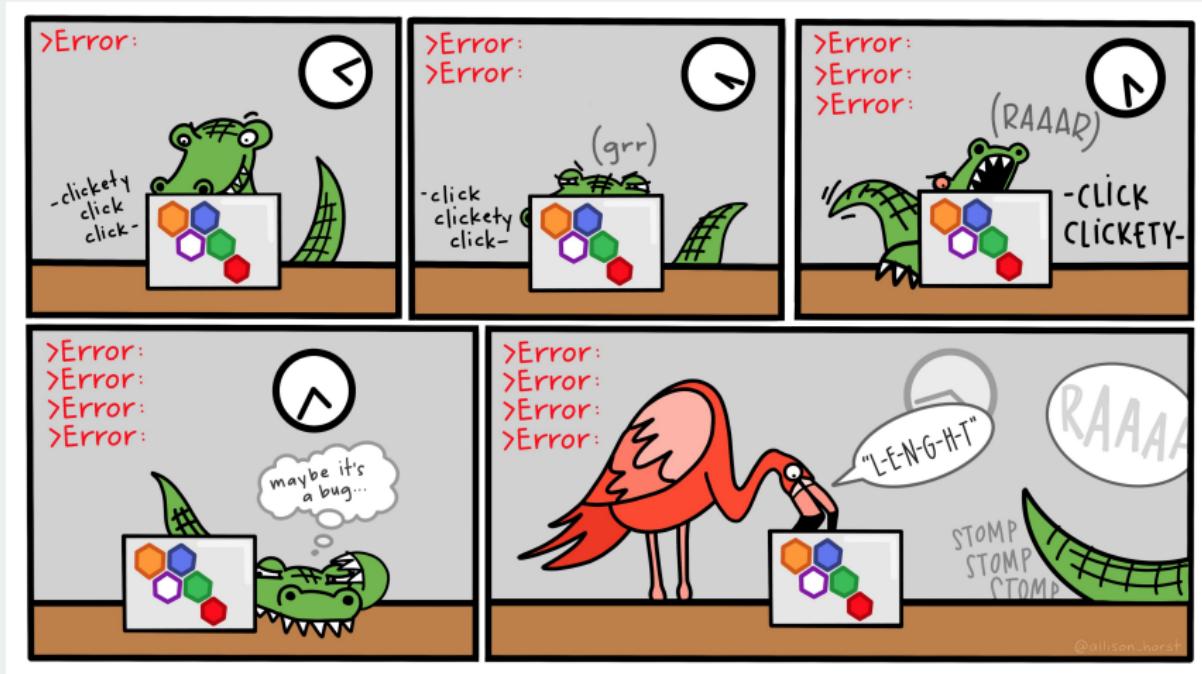
Automatically notify me of updates to RStudio

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

What R looks like

Code that you can type and run:

```
## 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:

```
my_numbers
```

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

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

```
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:

```
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:

```
mean(my_numbers)
```

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

Getting help with R

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

```
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:

```
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

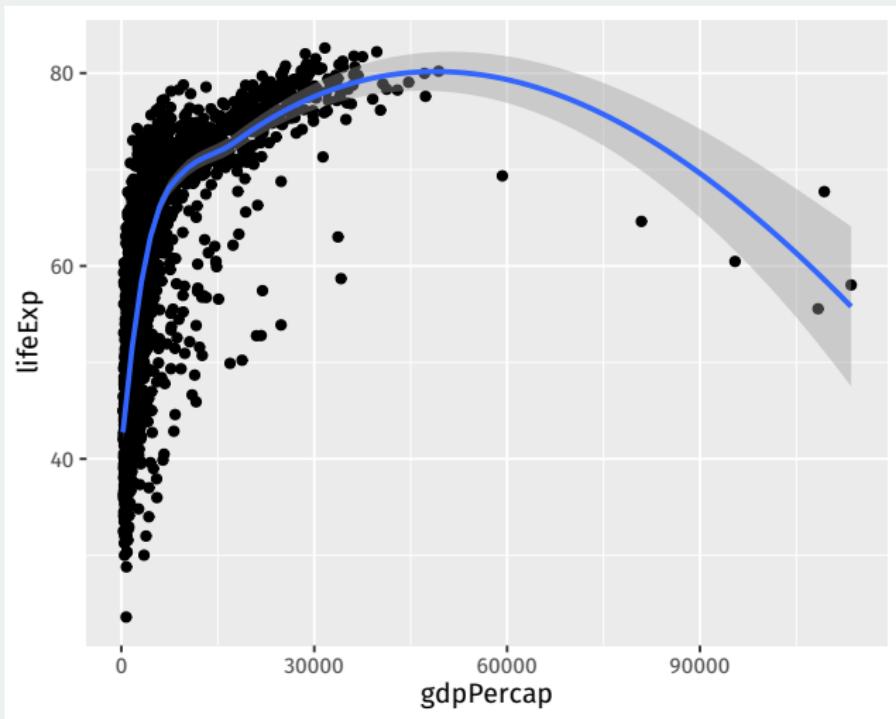
Gapminder data

```
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
```

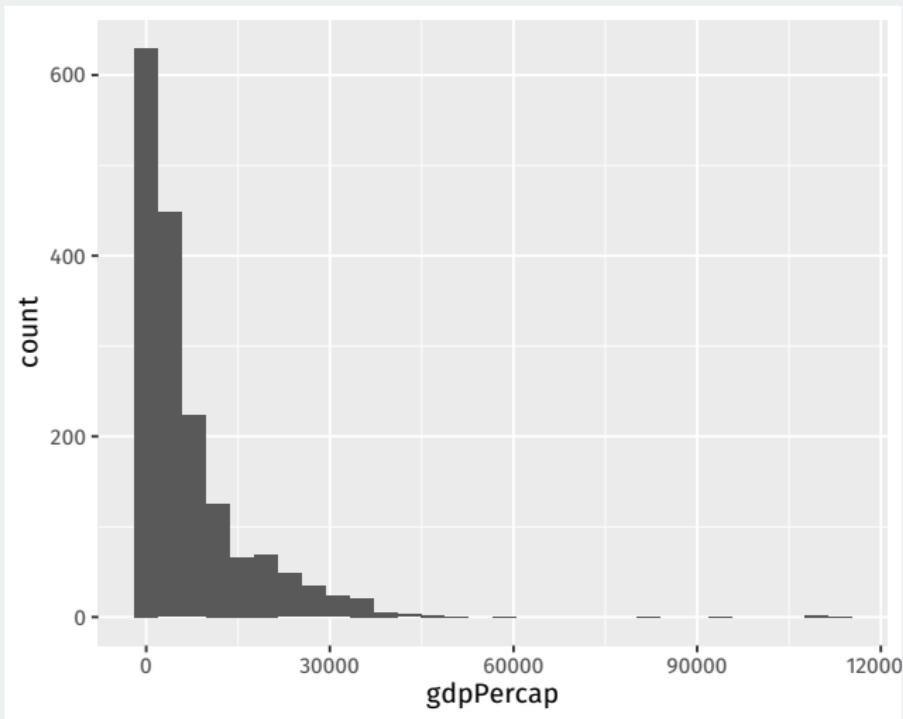
Plotting life expectancy over time

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



A histogram of GDP per capita

```
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>, percasiain <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:

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

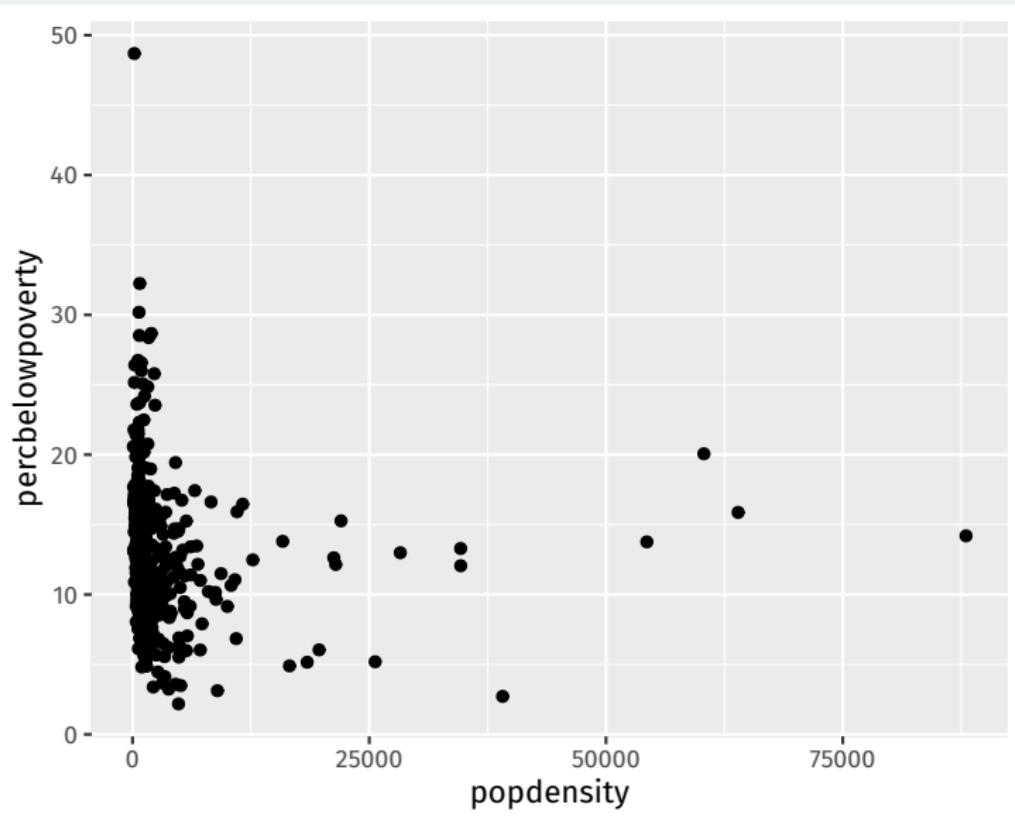
Mapping: tell ggplot what visual aesthetics correspond to which variables

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

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

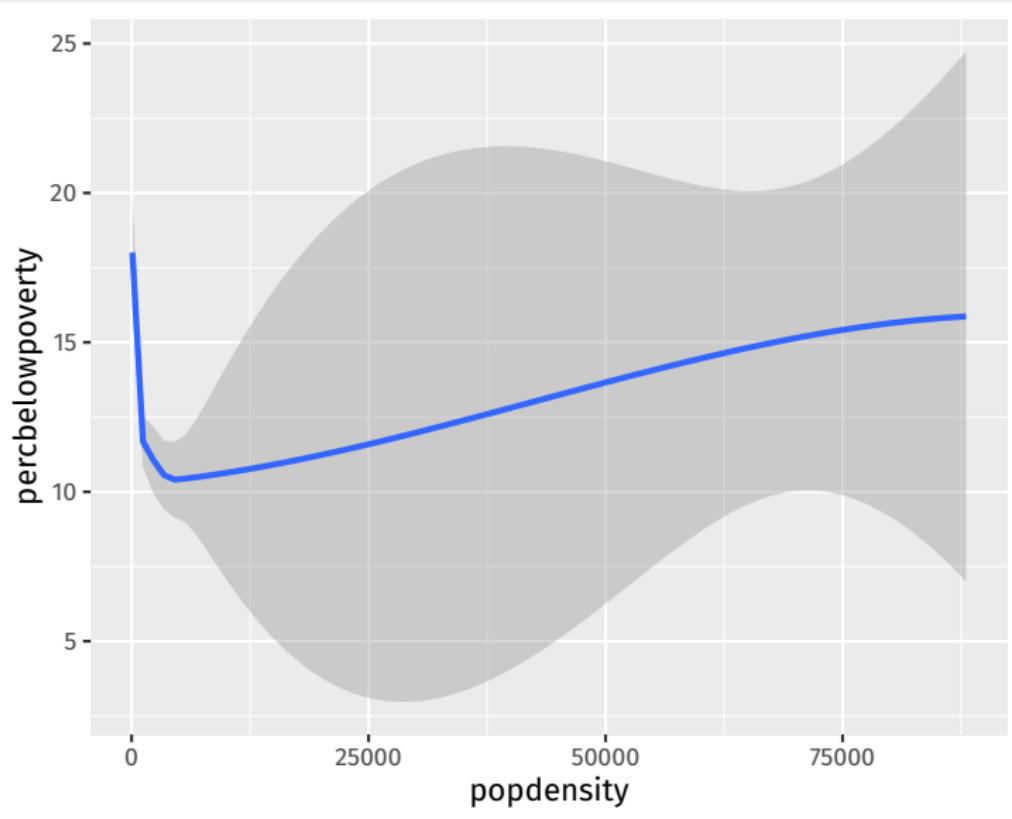
Adding a geom layer

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



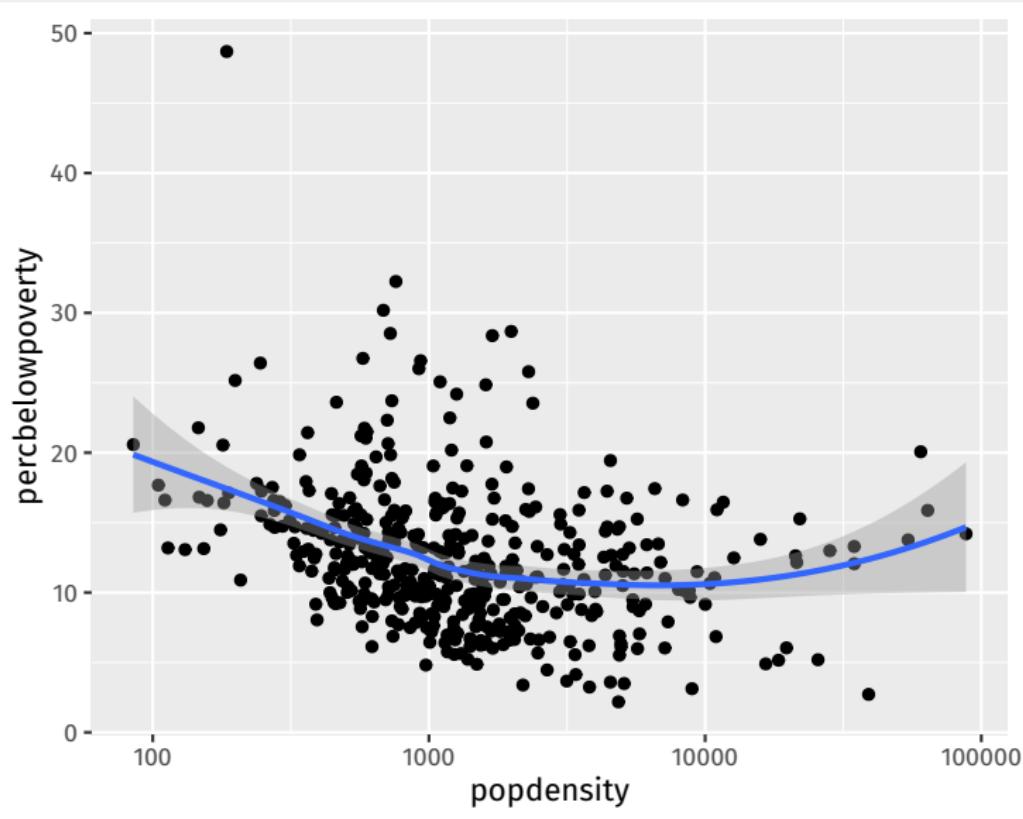
Trying a new geom

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



Layering geoms is additive

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

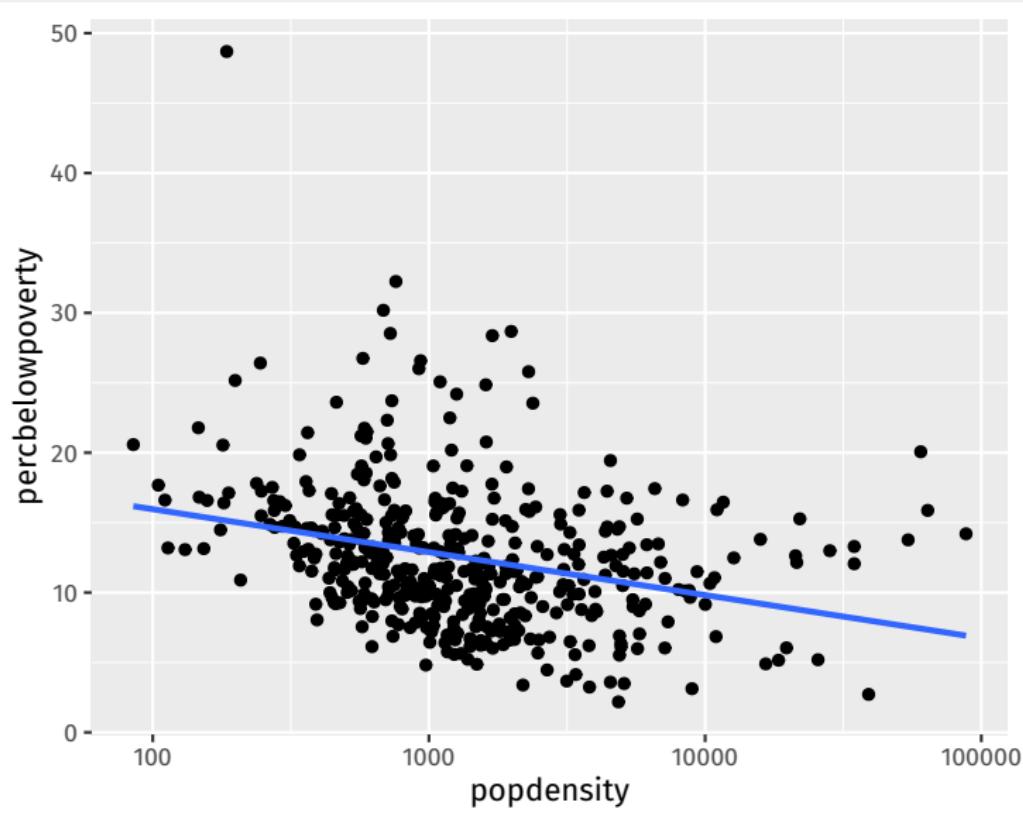


Geoms are functions

Geoms can take arguments:

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

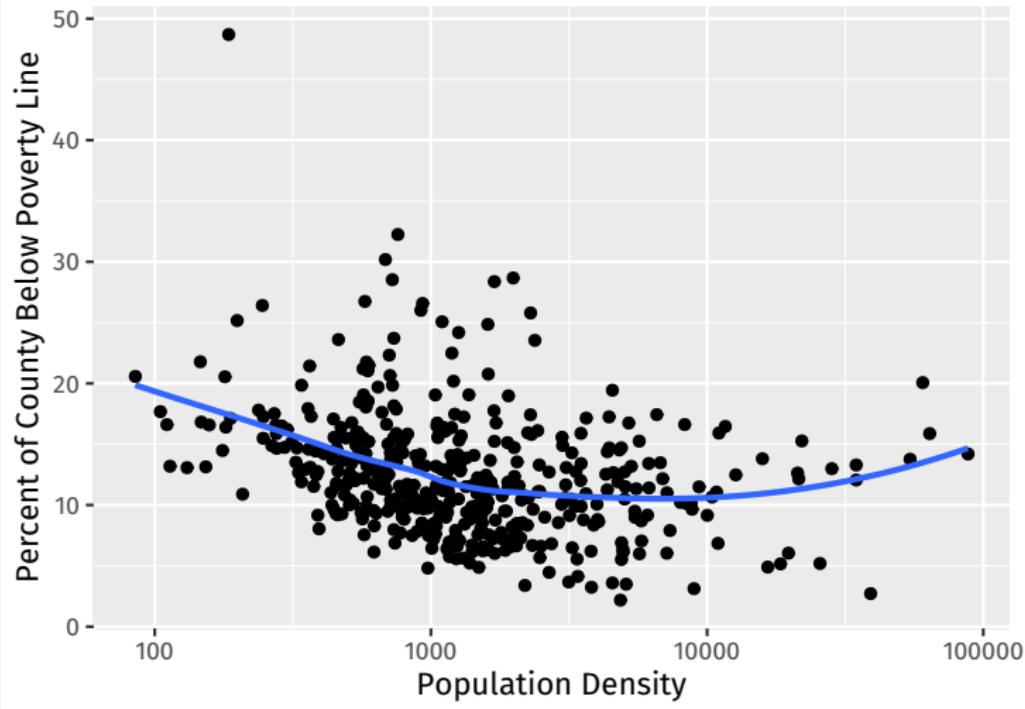
Tells geom_smooth to do a linear fit with no error region



Adding informative labels

```
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")
```

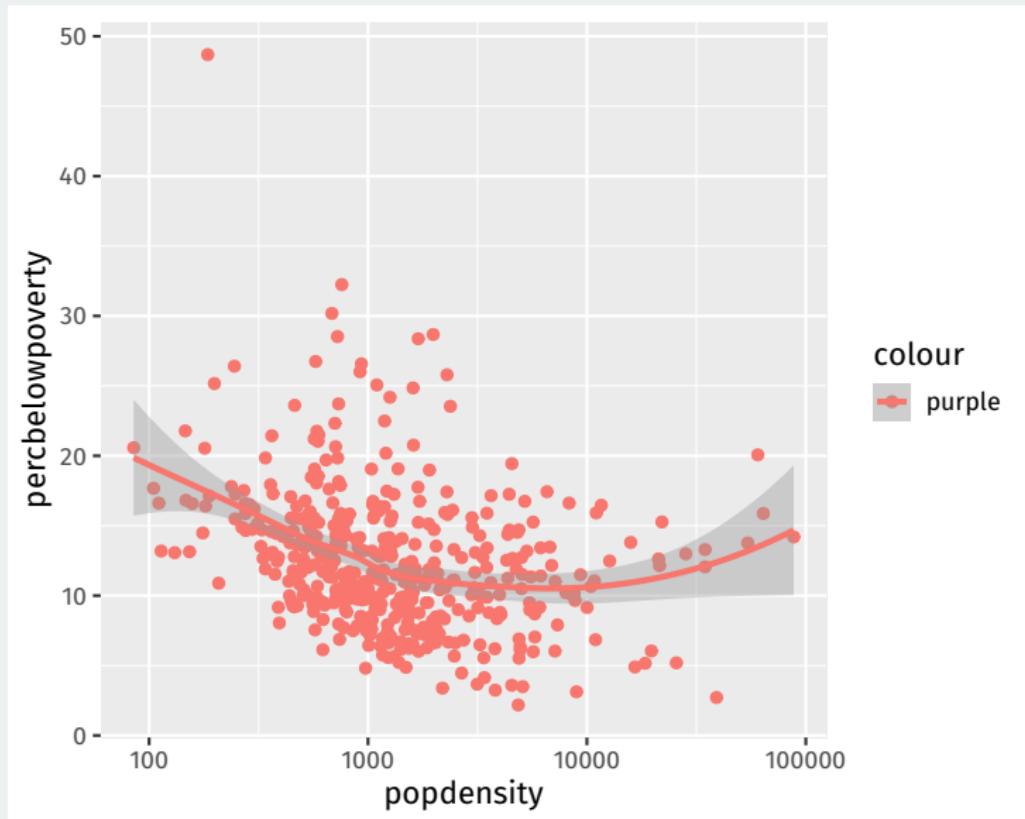
Poverty and Population Density Among Counties in the Midwest



Mapping vs setting aesthetics

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

Wait what?



Mapping always refers to variables

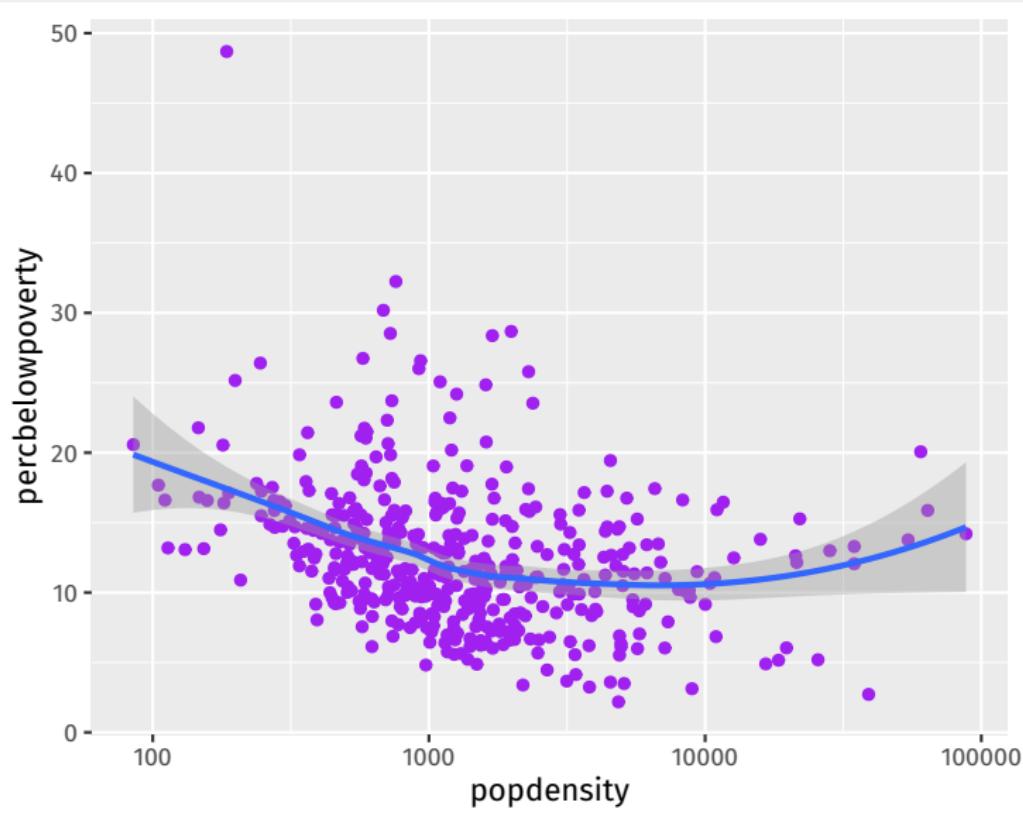
If passed a value other than a variable name, ggplot will implicitly create a variable with that value (in this case "purple" that is constant)

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

Setting aesthetics

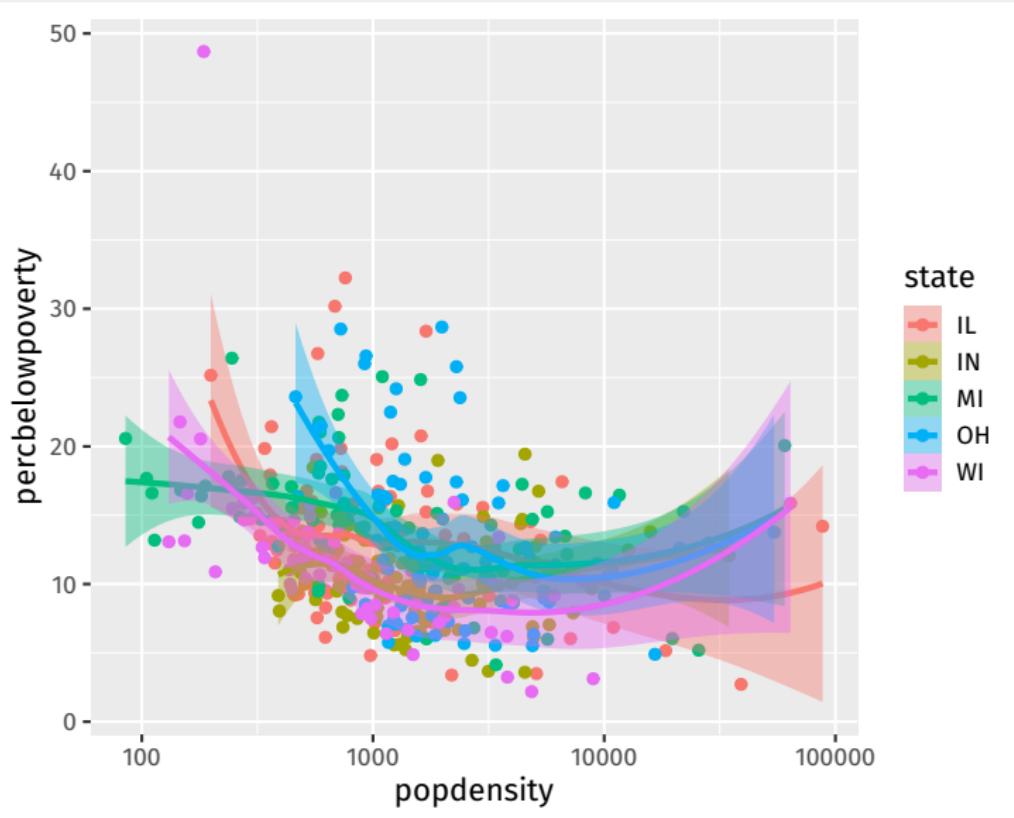
Set the color outside the `mapping = aes()` format.

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



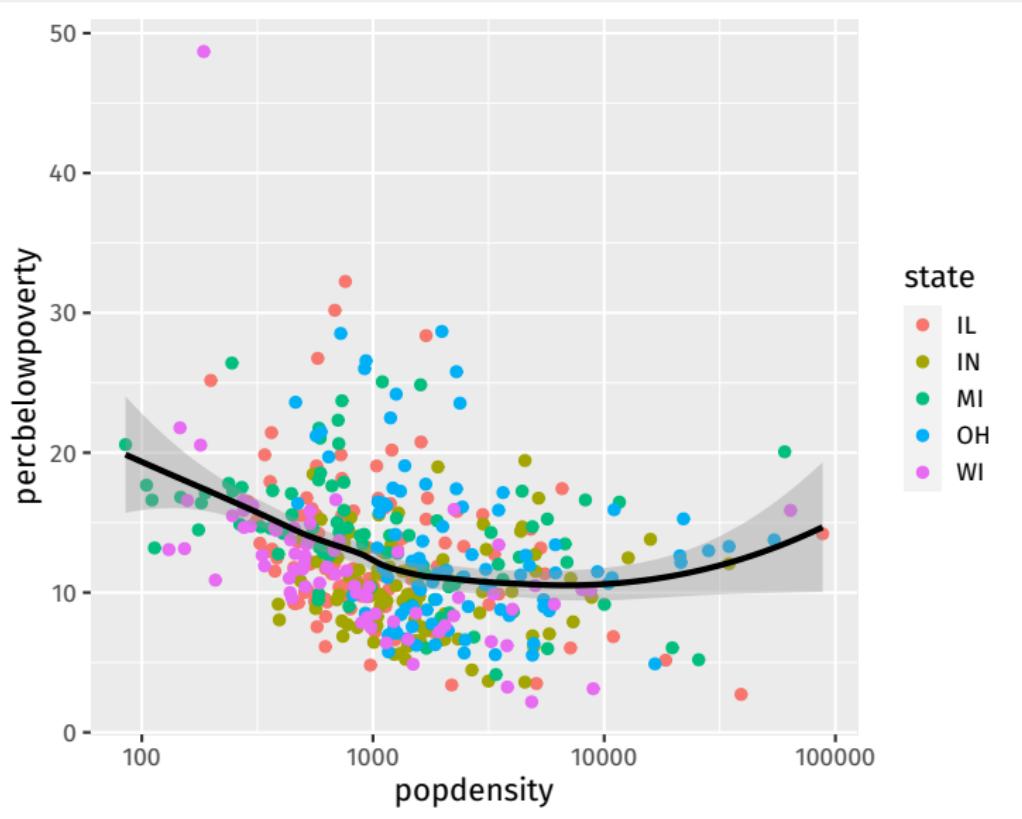
Mapping more aesthetics

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



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()
```



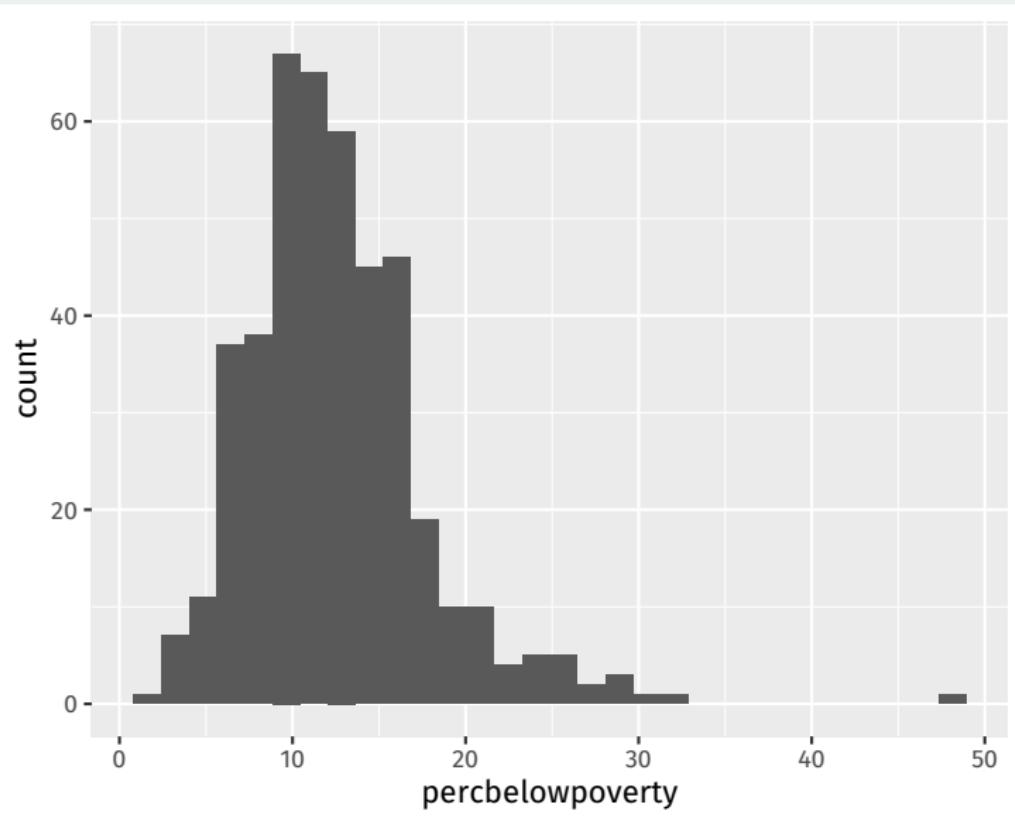
2/ Histograms and boxplots

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.

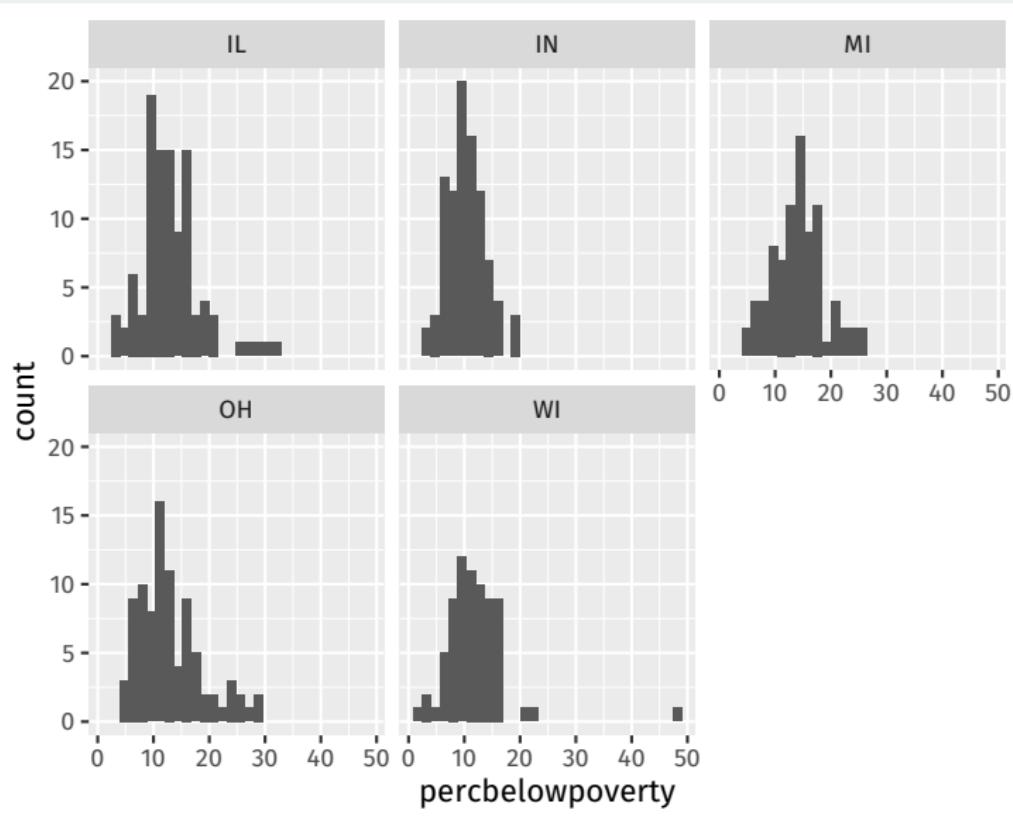


Creating small multiples with facets

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)
```

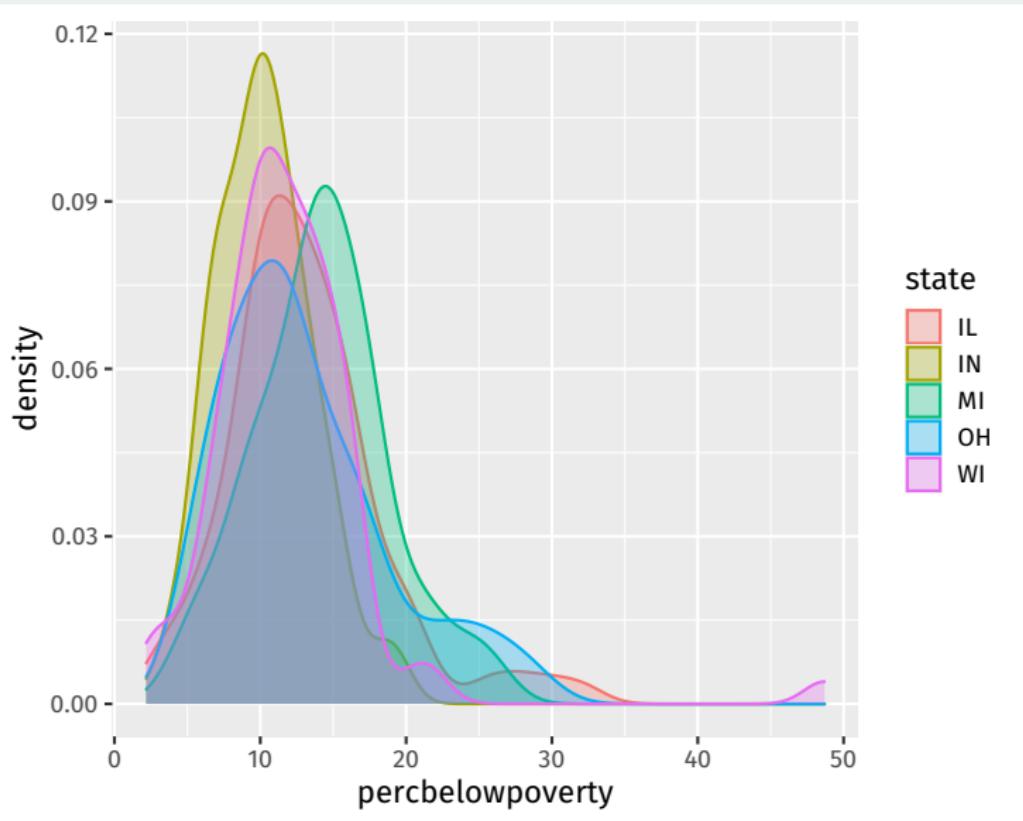
We'll see more of the `~` variable syntax (called a formula).



Density as alternative to histograms

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

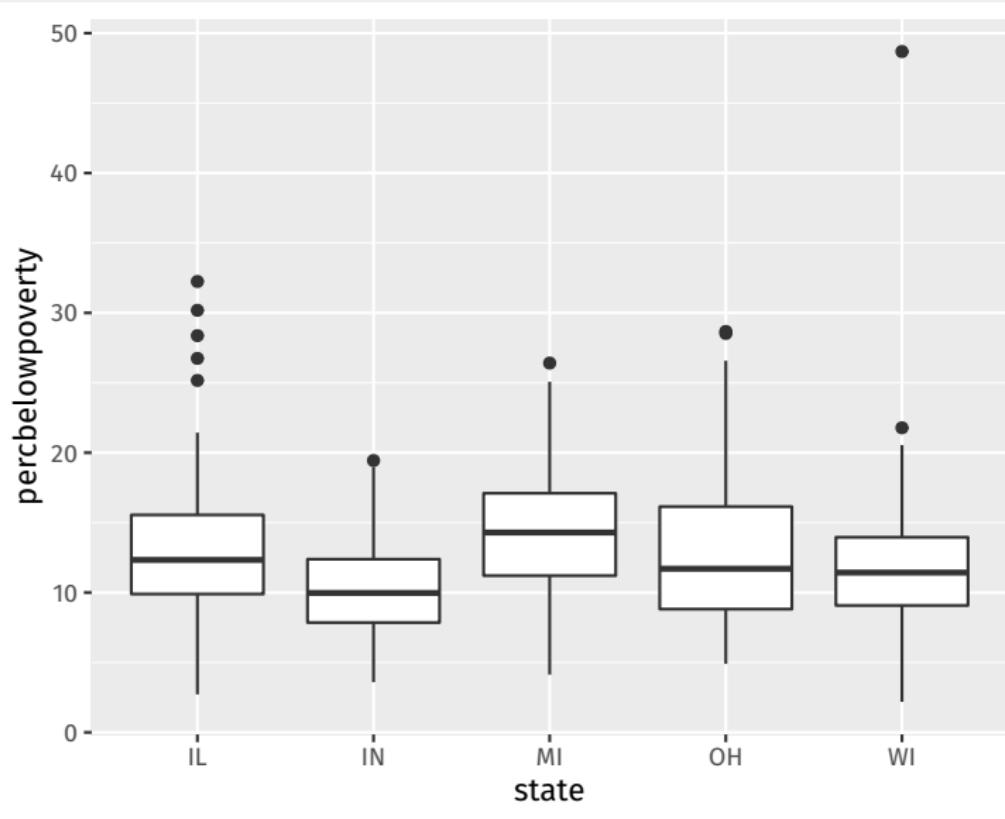
```
ggplot(data = midwest,  
       mapping = aes(x = percbelowpoverty,  
                      fill = state, color = state)) +  
  geom_density(alpha = 0.3)
```



Boxplots

Boxplots are another way to compare distributions across discrete groups.

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



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:
 - $1.5 \times \text{IQR}$ or max/min of the data, whichever is smaller.
 - Points beyond whiskers are outliers.

3/ Grouped data

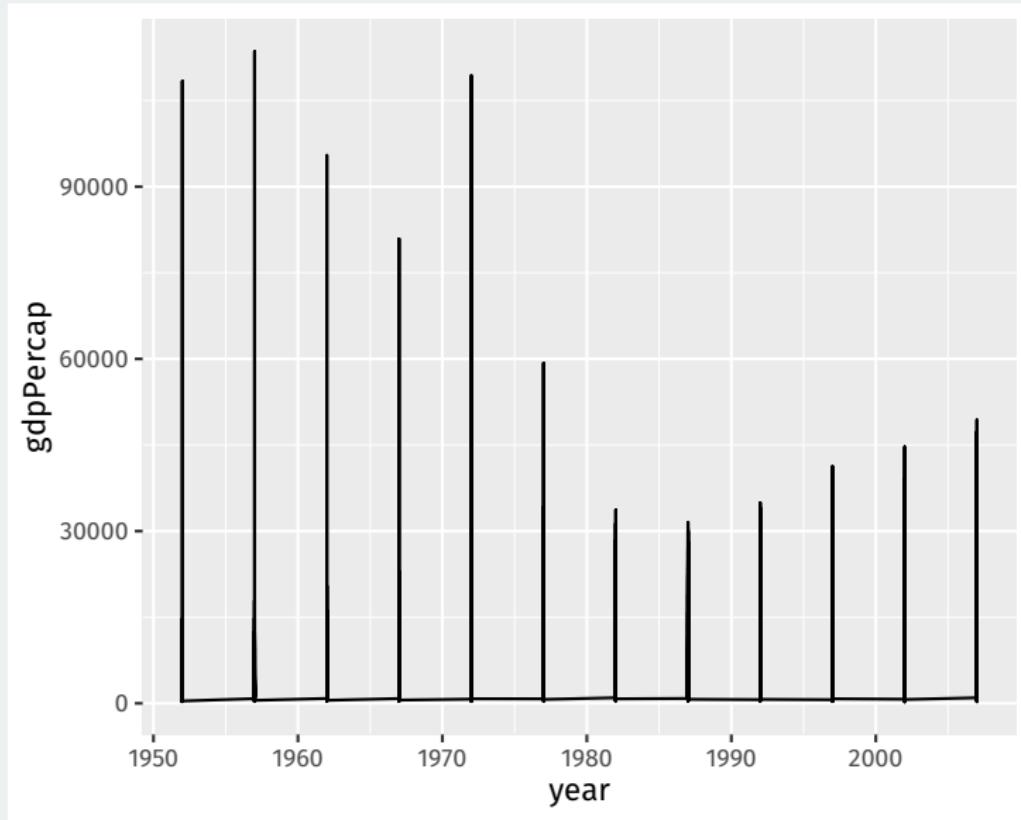
Back to the gapminder data

```
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

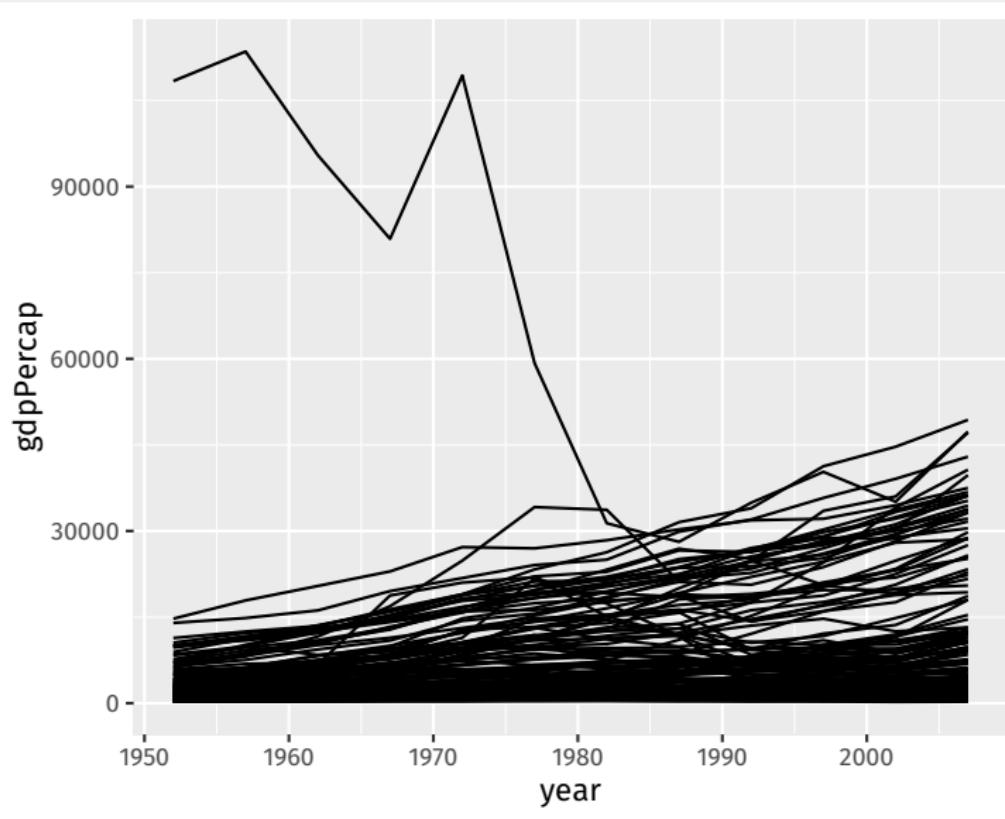
```
ggplot(data = gapminder,  
       mapping = aes(x = year,  
                      y = gdpPercap)) +  
  geom_line()
```



geom_line connects points from different countries in the same year.

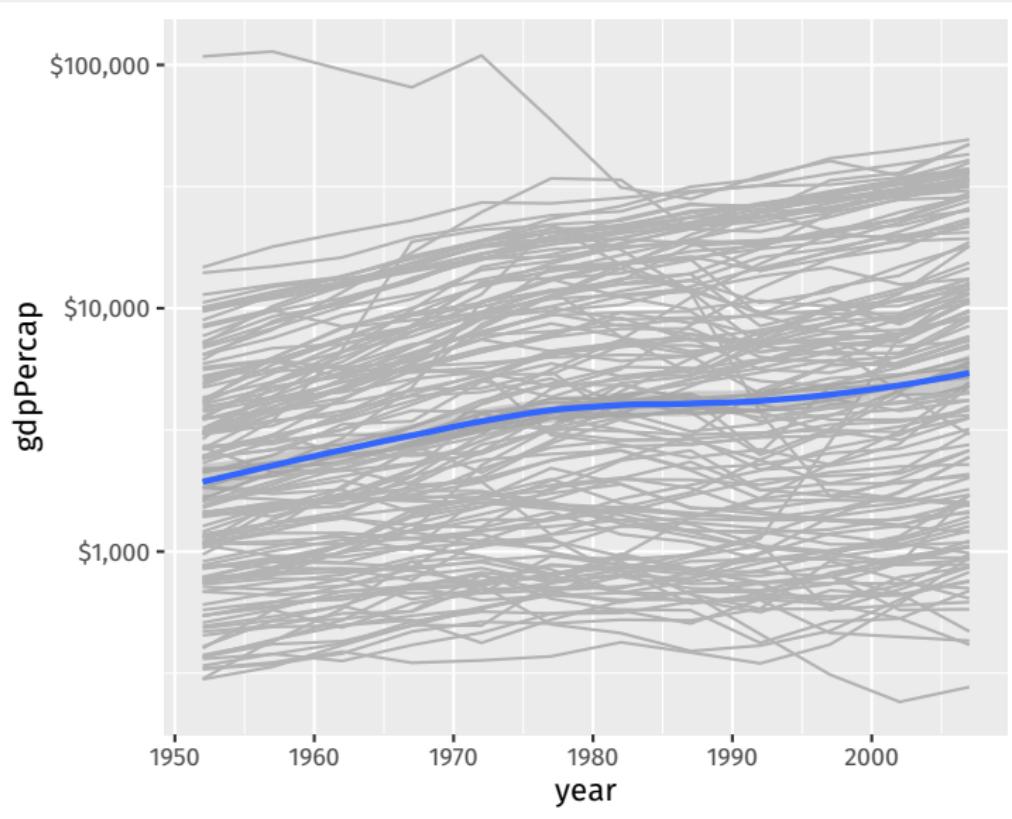
Tell geom_line how to group the lines

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



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)
```



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 16.4 17.3
## [14] 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3
## [27] 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 | gear | carb |
|------------------------|------|-----|-------|-----|------|------|------|----|----|------|------|
| ## Mazda RX4 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.62 | 16.5 | 0 | 1 | 4 | 4 |
| ## Mazda RX4 Wag | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.88 | 17.0 | 0 | 1 | 4 | 4 |
| ## Datsun 710 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.32 | 18.6 | 1 | 1 | 4 | 1 |
| ## Hornet 4 Drive | 21.4 | 6 | 258.0 | 110 | 3.08 | 3.21 | 19.4 | 1 | 0 | 3 | 1 |
| ## Hornet Sportabout | 18.7 | 8 | 360.0 | 175 | 3.15 | 3.44 | 17.0 | 0 | 0 | 3 | 2 |
| ## Valiant | 18.1 | 6 | 225.0 | 105 | 2.76 | 3.46 | 20.2 | 1 | 0 | 3 | 1 |
| ## Duster 360 | 14.3 | 8 | 360.0 | 245 | 3.21 | 3.57 | 15.8 | 0 | 0 | 3 | 4 |
| ## Merc 240D | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.19 | 20.0 | 1 | 0 | 4 | 2 |
| ## Merc 230 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.15 | 22.9 | 1 | 0 | 4 | 2 |
| ## Merc 280 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.44 | 18.3 | 1 | 0 | 4 | 4 |
| ## Merc 280C | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.44 | 18.9 | 1 | 0 | 4 | 4 |
| ## Merc 450SE | 16.4 | 8 | 275.8 | 180 | 3.07 | 4.07 | 17.4 | 0 | 0 | 3 | 3 |
| ## Merc 450SL | 17.3 | 8 | 275.8 | 180 | 3.07 | 3.73 | 17.6 | 0 | 0 | 3 | 3 |
| ## Merc 450SLC | 15.2 | 8 | 275.8 | 180 | 3.07 | 3.78 | 18.0 | 0 | 0 | 3 | 3 |
| ## Cadillac Fleetwood | 10.4 | 8 | 472.0 | 205 | 2.93 | 5.25 | 18.0 | 0 | 0 | 3 | 4 |
| ## Lincoln Continental | 10.4 | 8 | 460.0 | 215 | 3.00 | 5.42 | 17.8 | 0 | 0 | 3 | 4 |
| ## Chrysler Imperial | 14.7 | 8 | 440.0 | 230 | 3.23 | 5.34 | 17.4 | 0 | 0 | 3 | 4 |
| ## Fiat 128 | 32.4 | 4 | 78.7 | 66 | 4.08 | 2.20 | 19.5 | 1 | 1 | 4 | 1 |
| ## Honda Civic | 30.4 | 4 | 75.7 | 52 | 4.93 | 1.61 | 18.5 | 1 | 1 | 4 | 2 |
| ## Toyota Corolla | 22.8 | 4 | 71.1 | 65 | 4.23 | 1.83 | 19.9 | 1 | 1 | 4 | 1 |

tibbles: a tidyverse alternative

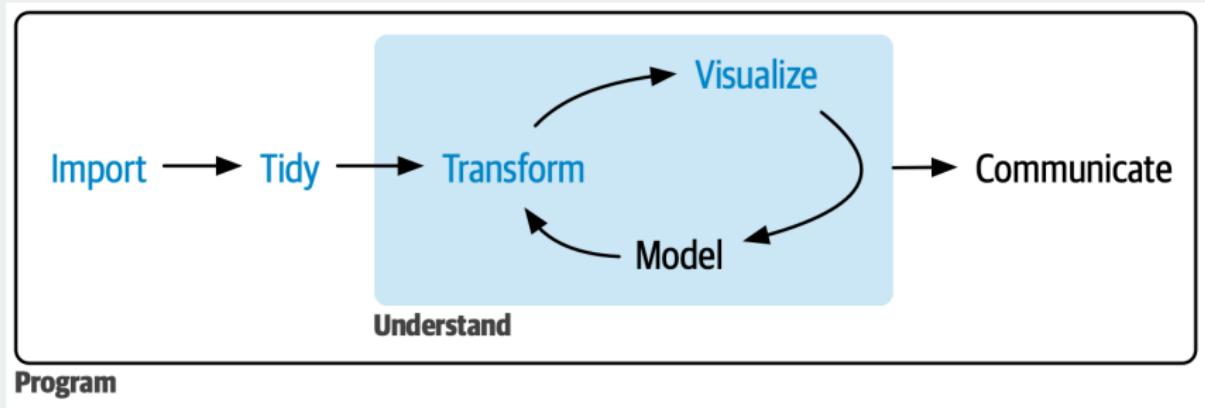
midwest

```
## # A tibble: 437 x 28 rows x columns
##   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>,
```

column types

abridged output

Transform-Visualize-Model cycle



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 conglomerate.

| 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) |

```
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
```

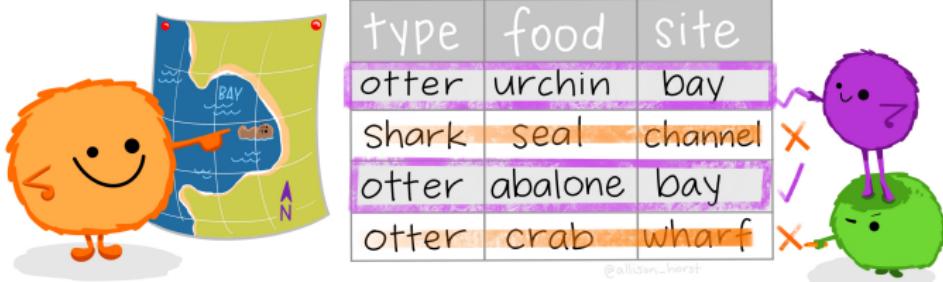
2/ Operating on rows

filter()

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

dplyr:: filter() KEEP ROWS THAT
satisfy
your CONDITIONS

keep rows from... this data... ONLY IF... type is "otter" AND site is "bay"
filter(df, type == "otter" & site == "bay")



| | type | food | site |
|---|-------|---------|---------|
| 1 | otter | urchin | bay |
| 2 | Shark | seal | channel |
| 3 | otter | abalone | bay |
| 4 | otter | crab | wharf |

© Allison Horst

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

```
## # A tibble: 626 x 10  
##   calls~1 affil~2 date      weekday ideology natio~3 local~4 sincl~5  
##   <chr>    <chr>   <date>     <ord>      <dbl>    <dbl>    <dbl>    <dbl>  
## 1 KPAX    CBS     2017-06-06 Tue        NA     0.00357  0.194     0  
## 2 KTAB    CBS     2017-06-06 Tue        NA     0.0945   0.109     0  
## 3 KAEF    ABC     2017-06-13 Tue       0.0242   0.180   0.234     1  
## 4 KBVU    FOX     2017-06-13 Tue       0.00894  0.186   0.245     1  
## 5 KBZK    CBS     2017-06-13 Tue       0.129    0.306   0.0763    0  
## 6 KCVU    FOX     2017-06-13 Tue       0.114    0.124   0.178     1  
## 7 KECI    NBC     2017-06-13 Tue       0.115    0.283   0.0926    1  
## 8 KHSL    CBS     2017-06-13 Tue       0.0821   0.274   0.248     0  
## 9 KNVN    NBC     2017-06-13 Tue       0.120    0.261   0.253     0  
## 10 KPAX   CBS     2017-06-13 Tue      0.0984   0.208   0.171     0  
## # ... with 616 more rows, 2 more variables: post <dbl>, month <ord>,  
## #   and abbreviated variable names 1: callsign, 2: affiliation,  
## #   3: national_politics, 4: local_politics, 5: sinclair2017
```

Multiple conditions means “and”

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



```
## # A tibble: 130 x 10
##   calls~1 affil~2 date      weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>   <date>     <ord>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 KBVU     FOX     2017-06-13 Tue       0.00894  0.186   0.245     1
## 2 KCVU     FOX     2017-06-13 Tue       0.114    0.124   0.178     1
## 3 WEMT     FOX     2017-06-13 Tue       0.235    0.149   0.155     1
## 4 WYDO     FOX     2017-06-13 Tue       0.0949   0.182   0.180     1
## 5 KBVU     FOX     2017-06-20 Tue      NA        0.0229   0.268     1
## 6 KCVU     FOX     2017-06-20 Tue      NA        0.0170   0.261     1
## 7 KXVA     FOX     2017-06-20 Tue      NA        0.0203   0.0939    0
## 8 WEMT     FOX     2017-06-20 Tue       0.268    0.134   0.151     1
## 9 WYDO     FOX     2017-06-20 Tue       0.0590   0.155   0.0943    1
## 10 KBVU    FOX     2017-06-27 Tue      NA        0.0601   0.279     1
## # ... with 120 more rows, 2 more variables: post <dbl>, month <ord>,
## # and abbreviated variable names 1: callsign, 2: affiliation,
## # 3: national_politics, 4: local_politics, 5: sinclair2017
```

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!

```
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"`?
```

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

```
## # A tibble: 1,525 x 10
##   calls~1 affil~2 date      weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>   <date>     <ord>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 KXVA     FOX     2017-06-05 Mon       NA     0.0393   0.0262    0
## 2 KTMF     ABC     2017-06-07 Wed      8.42e-2  0.152    0.129    0
## 3 KTXS     ABC     2017-06-07 Wed     -4.88e-4  0.0925   0.0791   1
## 4 KXVA     FOX     2017-06-07 Wed      NA     0.00718  0.00479   0
## 5 KAEF     ABC     2017-06-08 Thu     4.26e-2  0.213    0.228    1
## 6 KBVU     FOX     2017-06-08 Thu    -8.60e-2  0.169    0.247    1
## 7 KTMF     ABC     2017-06-08 Thu     4.33e-2  0.179    0.139    0
## 8 KTXS     ABC     2017-06-08 Thu     6.27e-2  0.158    0.115    1
## 9 KXVA     FOX     2017-06-08 Thu      NA     0.0124   0.0873   0
## 10 WCTI    ABC     2017-06-08 Thu    1.39e-1  0.225    0.0759   1
## # ... with 1,515 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: callsign,
## #   2: affiliation, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

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

## # A tibble: 66 x 10
##   calls~1 affil~2 date      weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>   <date>    <ord>     <dbl>    <dbl>    <dbl>    <dbl>
## 1 KAEF     ABC     2017-06-27 Tue     -0.0117   0.162   0.207     1
## 2 KECI     NBC     2017-06-27 Tue     -0.00362  0.177   0.0932    1
## 3 KHSL     CBS     2017-06-27 Tue     -0.0735   0.170   0.259     0
## 4 KNVN     NBC     2017-06-27 Tue     -0.0175   0.180   0.262     0
## 5 KPAX     CBS     2017-06-27 Tue     -0.134    0.219   0.120     0
## 6 KTXS     ABC     2017-06-27 Tue     -0.0307  0.129   0.100     1
## 7 WCTI     ABC     2017-06-27 Tue     -0.0308  0.187   0.119     1
## 8 WITN     NBC     2017-06-27 Tue     -0.0233  0.155   0.108     0
## 9 WJHL     CBS     2017-06-27 Tue     -0.00388 0.166   0.210     0
## 10 WNCT    CBS    2017-06-27 Tue     -0.130    0.181   0.125     0
## # ... with 56 more rows, 2 more variables: post <dbl>, month <ord>,
## #   and abbreviated variable names 1: callsign, 2: affiliation,
## #   3: national_politics, 4: local_politics, 5: sinclair2017
```

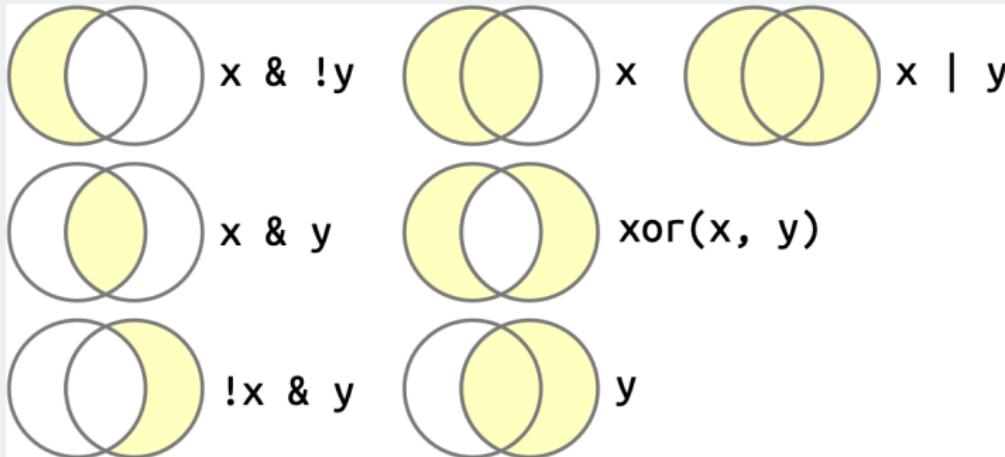
Combining %in%

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

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

```
## # A tibble: 1,253 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 KAEF      ABC     2017-06-09 Fri      0.0870    0.153     0.269     1  
## 5 KBVU      FOX     2017-06-09 Fri      NA      0.0553    0.384     1  
## 6 KECI      NBC     2017-06-09 Fri      0.115     0.216    0.108     1  
## 7 KPAX      CBS     2017-06-09 Fri      0.0882    0.315     0.128     0  
## 8 KRBC      NBC     2017-06-09 Fri      0.0929    0.152     0.147     0  
## 9 KTAB      CBS     2017-06-09 Fri      0.0588    0.0711   0.176     0  
## 10 KTMF     ABC     2017-06-09 Fri      NA      0.0495    0.0999    0  
## # ... with 1,243 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
```

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

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

```
## # 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 KAEF      ABC     2017-06-08 Thu      0.0426   0.213   0.228    1  
## 2 KAEF      ABC     2017-06-09 Fri      0.0870   0.153   0.269    1  
## 3 KAEF      ABC     2017-06-12 Mon     0.0135   0.149   0.188    1  
## 4 KAEF      ABC     2017-06-13 Tue     0.0242   0.180   0.234    1  
## 5 KAEF      ABC     2017-06-14 Wed     0.123    0.182   0.0968   1  
## 6 KAEF      ABC     2017-06-15 Thu     0.0778   0.114   0.203    1  
## 7 KAEF      ABC     2017-06-16 Fri     NA       0.109   0.176    1  
## 8 KAEF      ABC     2017-06-19 Mon     0.778    0.0823  0.179    1  
## 9 KAEF      ABC     2017-06-20 Tue     0.115    0.131   0.163    1  
## 10 KAEF     ABC     2017-06-21 Wed    -0.315   0.130   0.192   1  
## # ... 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
```

Which station-dates were the most liberal?

```
news |>
  arrange(ideology)

## # 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      2017-10-19 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
## 6 KAEF      ABC      2017-06-21 Wed     -0.315   0.130    0.192    1
## 7 KTMF      ABC      2017-12-01 Fri     -0.303   0.179    0.150    0
## 8 KWYB      ABC      2017-12-01 Fri     -0.303   0.160    0.161    0
## 9 KTVM      NBC      2017-09-01 Fri     -0.302   0.0507   0.106    1
## 10 KNVN     NBC      2017-12-08 Fri    -0.299   0.121    0.211    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
```

Which station-dates were the most conservative?

Use `desc()` to reverse the order:

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

```
## # 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 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  
## 6 KTMF      ABC     2017-11-06 Mon       0.455   0.138    0.154    0  
## 7 KAEF      ABC     2017-06-29 Thu       0.447   0.126    0.220    1  
## 8 KPAX      CBS     2017-11-23 Thu       0.437   0.125    0.128    0  
## 9 KTAB      CBS     2017-11-16 Thu       0.427   0.0631   0.104    0  
## 10 KCVU     FOX     2017-07-06 Thu       0.406   0.154    0.148    1  
## # ... 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
```

3/ Operating on columns

`select()`:

`select()` selects columns via their names.

Selecting based on names

```
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

```
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

```
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

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

```
## # A tibble: 3,137 x 5  
##   ideology national_politics local_politics sinclair2017 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
```

Combining multiple selections

```
news |>  
  select(callsign:weekday, ends_with("politics"))  
  
## # A tibble: 3,137 x 6  
##   callsign affiliation date     weekday national_politics local_p~1  
##   <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 name  
## #   1: local_politics
```

rename()

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

```
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.

```

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

```
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.

```
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

```
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

```
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 conglomerate.

| 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) |

```
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:

```
## 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:

```
## 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`

```
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.

```

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

```
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.

```
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

```
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

```
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
```

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

```
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

```
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
```

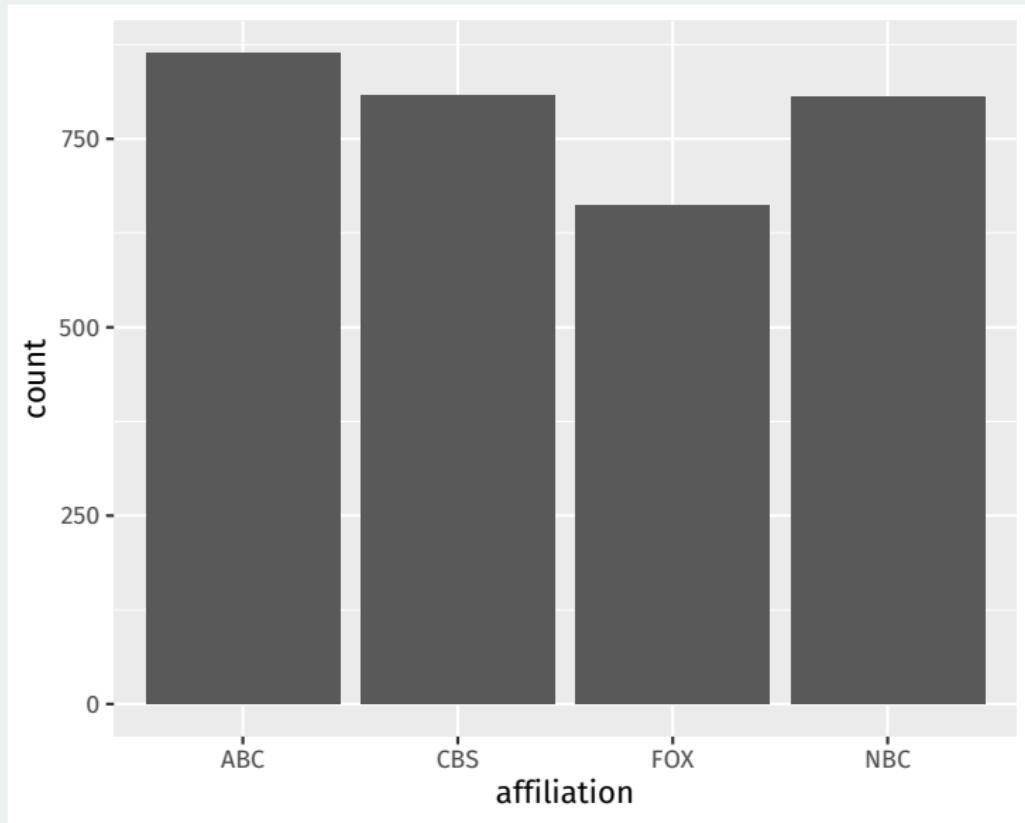
4| Creating barplots

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.

```
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.

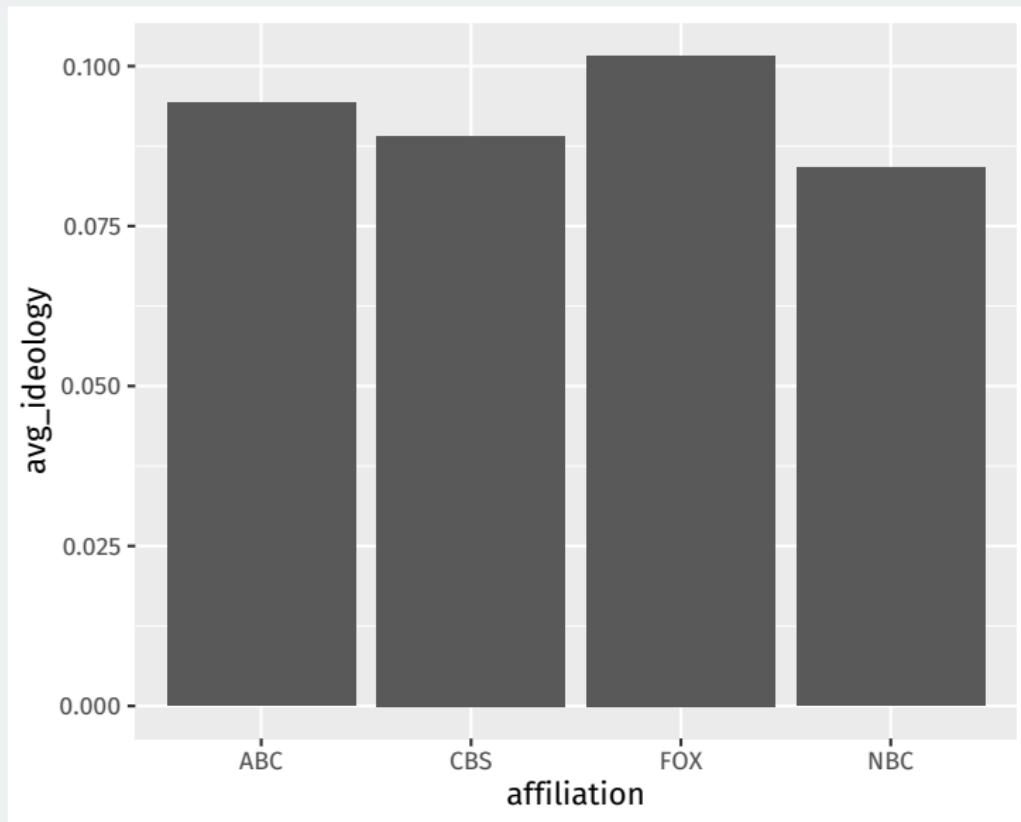
When the height of the bar is another variable in our data and not just a count, we set the x and y aesthetics and use `geom_col()` instead of `geom_bar()`.

Let's create a group summary:

```
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
```

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

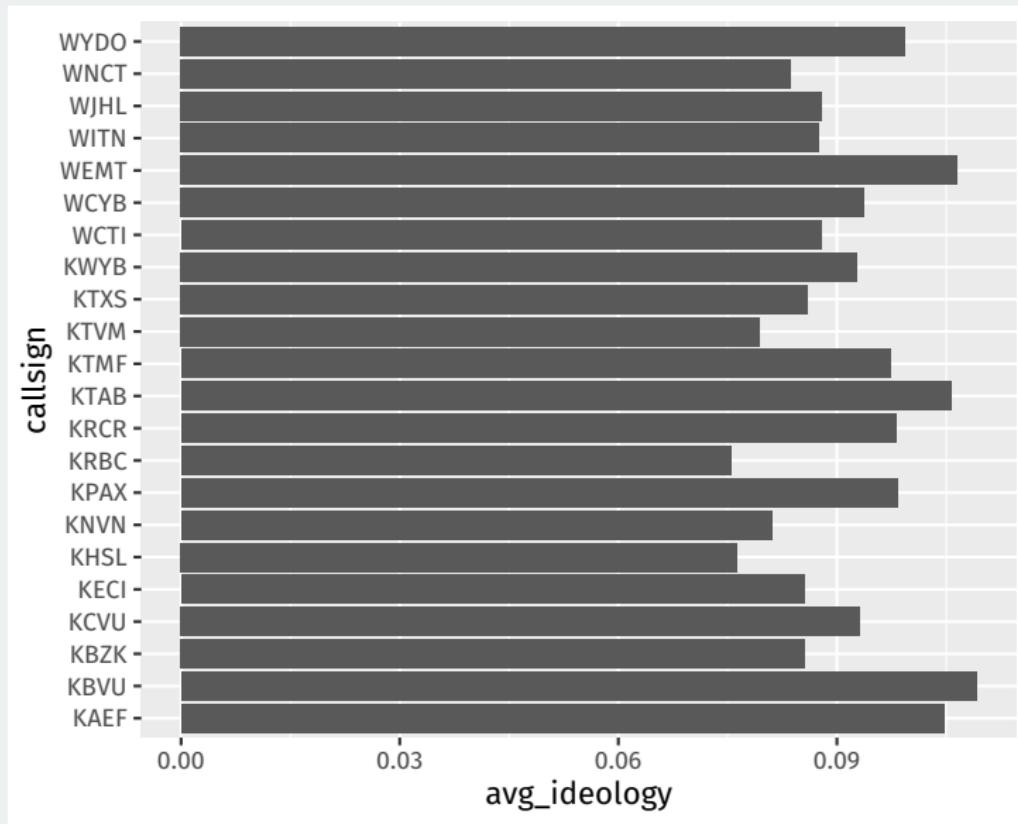


A more complicated example

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)
```

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

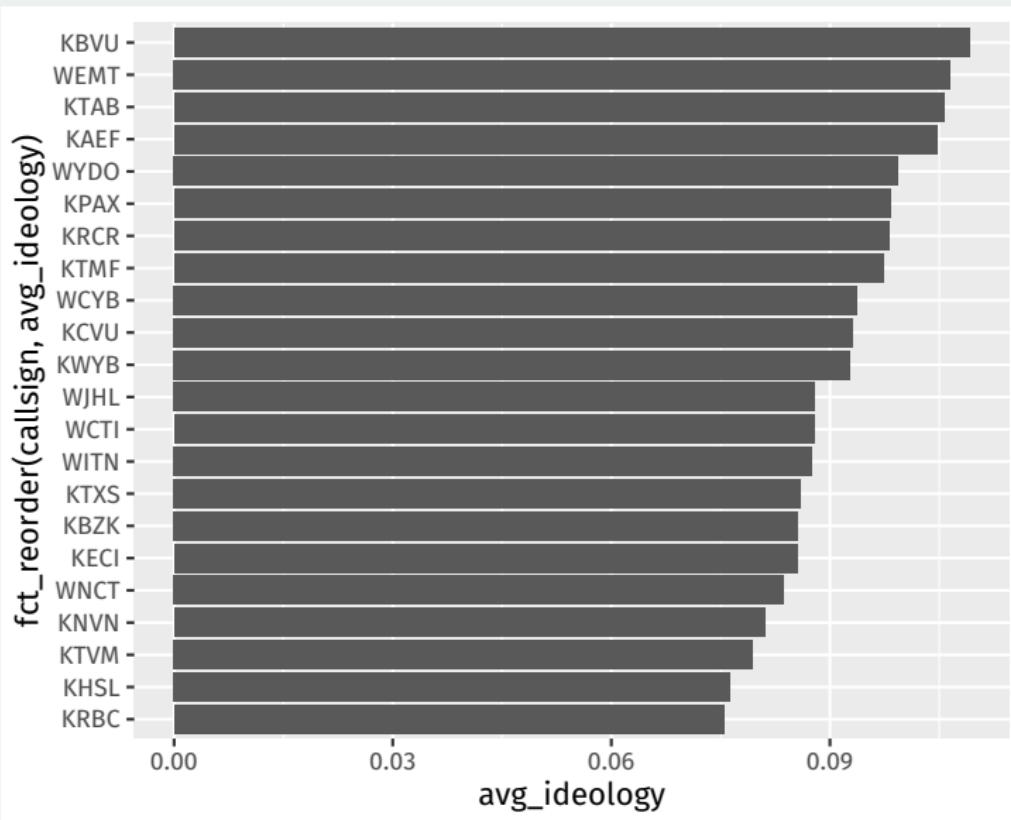


How do we reorder the stations?

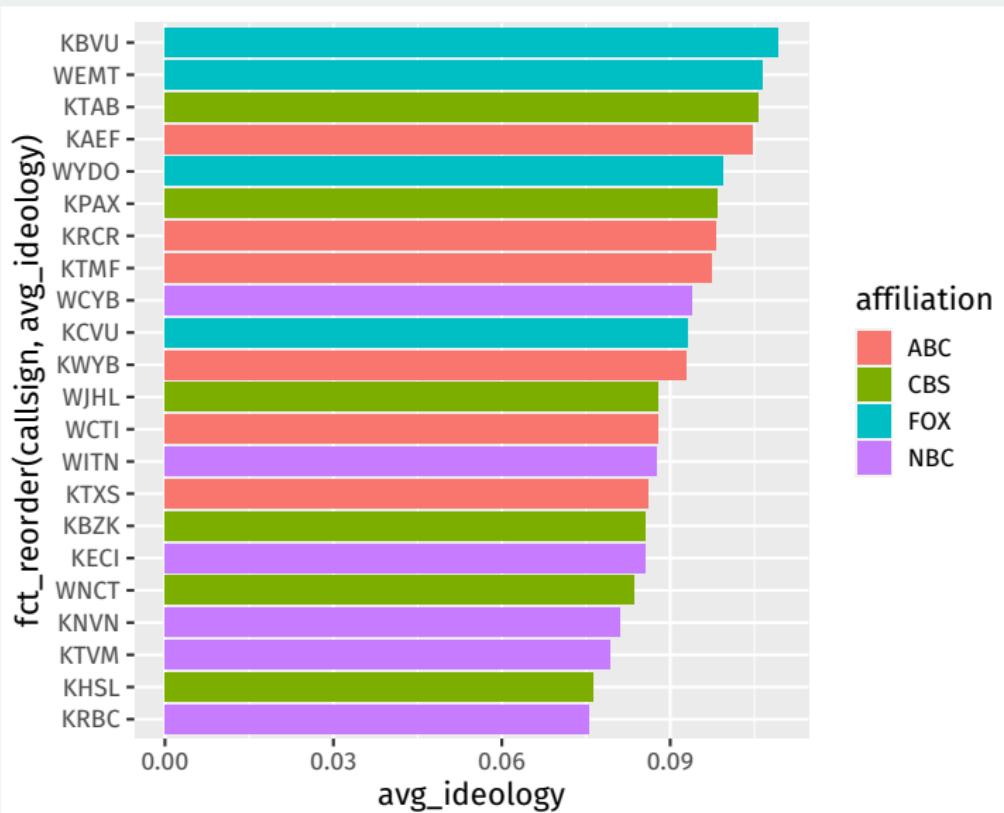
We would like to order the stations by ideology.

`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.

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



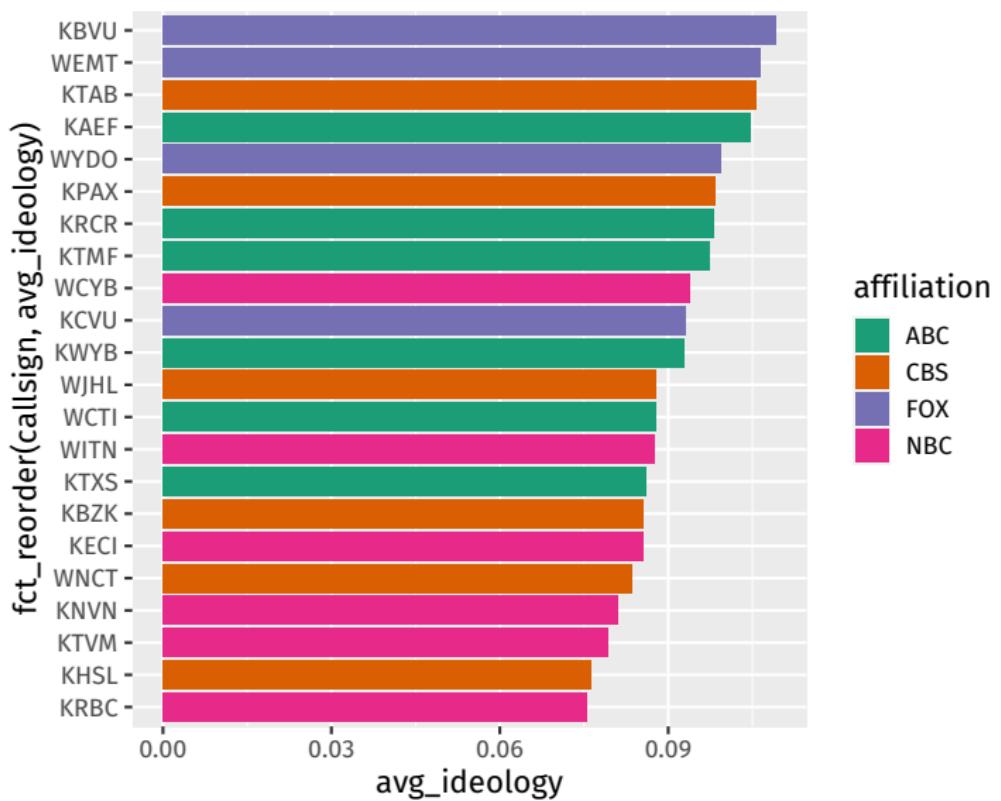
```
ggplot(station_ideology,  
       mapping = aes(x = avg_ideology,  
                      y = fct_reorder(callsign, avg_ideology))) +  
  geom_col(mapping = aes(fill = affiliation))
```



Setting the color palette

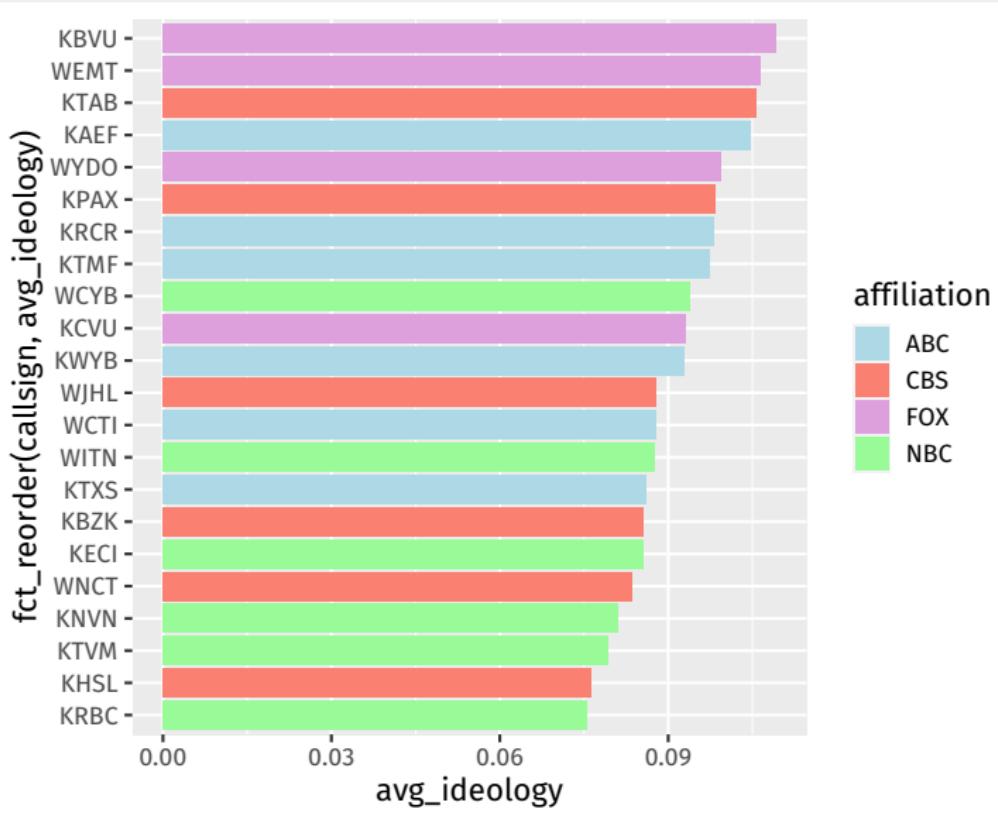
We can use color palettes from a project called ColorBrewer

```
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")
```



Manually setting the color palette

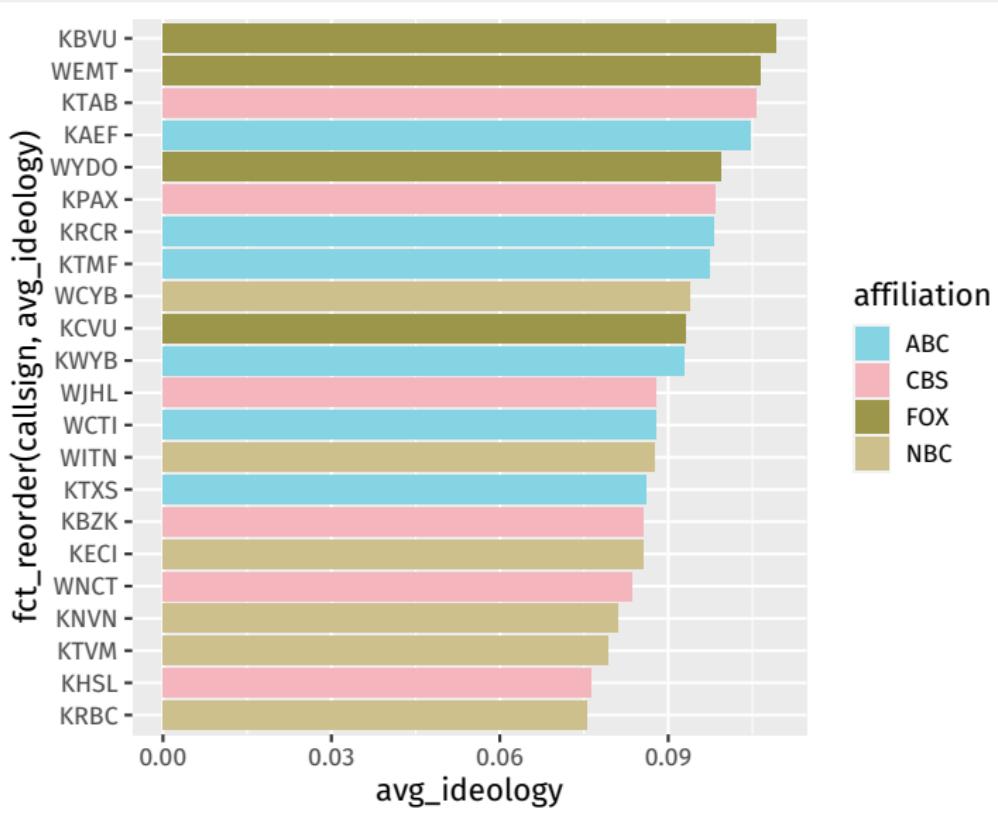
```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = c(ABC = "lightblue",
                               CBS = "salmon",
                      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



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"))
```



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 Brockman^{1*} and Joshua Kalla²

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 homophobic 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.

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

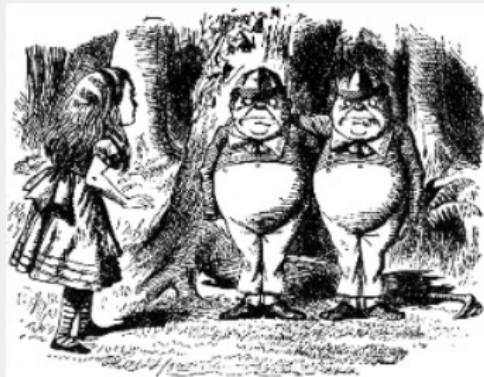
- What does “ T_i causes Y_i ” mean? \rightsquigarrow **counterfactuals**, “what if”
- Would respondent change their support based on the conversation?
- Two **potential outcomes**:
 - $Y_i(1)$: would respondent i support ND laws if they had trans rights script?
 - $Y_i(0)$: would respondent i support 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 than 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, \dots, 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**:

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

- $\sum_{i=1}^n$ 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:

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

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

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

Quantity of interest

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

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

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

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

- \bar{Y}_{treated} : sample average outcome for treated group
- \bar{Y}_{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:

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

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

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

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

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

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
 - \bar{X}_{treated} : average value of variable for treated group.
 - \bar{X}_{control} : average value of variable for control group.
 - Under randomization, $\bar{X}_{\text{treated}} - \bar{X}_{\text{control}} \approx 0$

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:
 - $\bar{Y}_{\text{treated, A}} - \bar{Y}_{\text{control}}$
 - $\bar{Y}_{\text{treated, B}} - \bar{Y}_{\text{control}}$
 - $\bar{Y}_{\text{treated, A}} - \bar{Y}_{\text{treated, B}}$
- If treatment arms are randomly assigned, these differences will be good estimators for each causal contrast.

3/ Calculating effects

Transphobia study data

```
## 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

```
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
```

```
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:

```
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

```
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
```

Gov 50: 7. Observational Studies

Matthew Blackwell

Harvard University

Roadmap

1. Calculating effects
2. Observational Studies

1/ Calculating effects

Transphobia study data

```
## 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

```
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
```

```
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:

```
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

```
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 ↗ New York Times, conservatives ↗ 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 → 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)? |

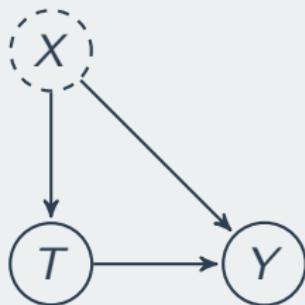
```
library(tidyverse)
library(gov50data)
newspapers

## # 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



- **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
 - \bar{Y}_{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 **research 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:

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

- Could there be confounders?

Cross-sectional design in 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 ↪ threat of **unmeasured confounding**

Statistical control in 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:

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

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

Before and after in 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 happened to **treatment group** without treatment.
- DiD estimate:

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

- 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

```
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

```
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

```
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

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

- **Median:**

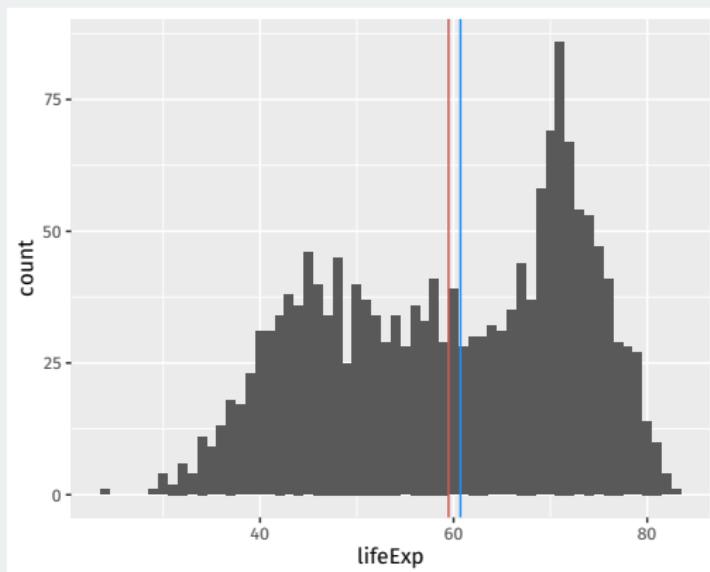
$$\text{median} = \begin{cases} \text{middle value} & \text{if number of entries is odd} \\ \frac{\text{sum of two middle values}}{2} & \text{if number of entries is even} \end{cases}$$

- 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?

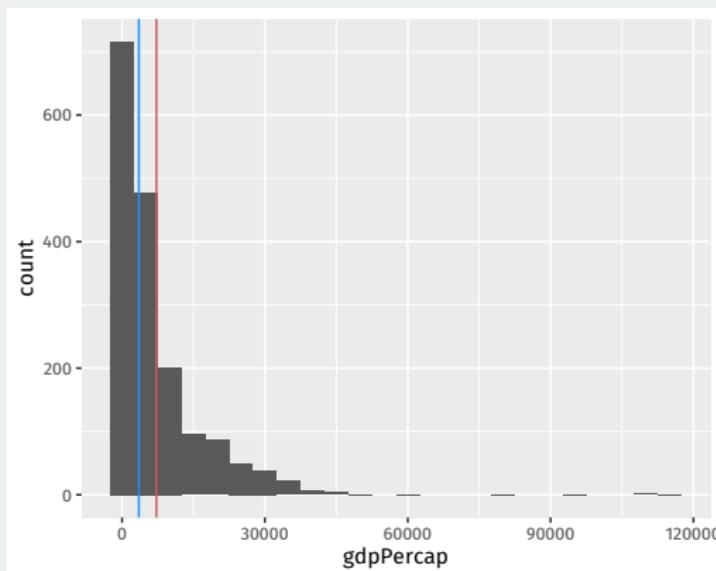
```
ggplot(gapminder, aes(x = lifeExp)) +  
  geom_histogram(binwidth = 1) +  
  geom_vline(aes(xintercept = mean(lifeExp)), color = "indianred") +  
  geom_vline(aes(xintercept = median(lifeExp)), color = "dodgerblue")
```



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

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

```
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")
```

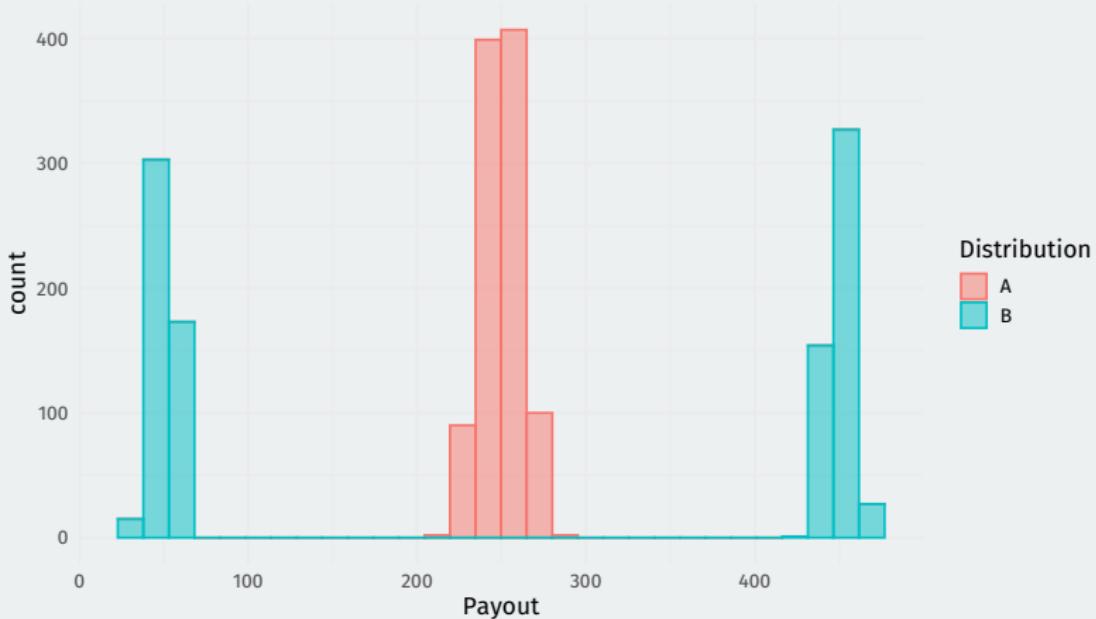


```
summary(gapminder$gdpPercap)
```

| | 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 deviation²
- 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 ↪ **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

```
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

```
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

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

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

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

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

```
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 \Rightarrow more non-response:

```
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:

```
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

```
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

```
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

```
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

```
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

```
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

```
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
```

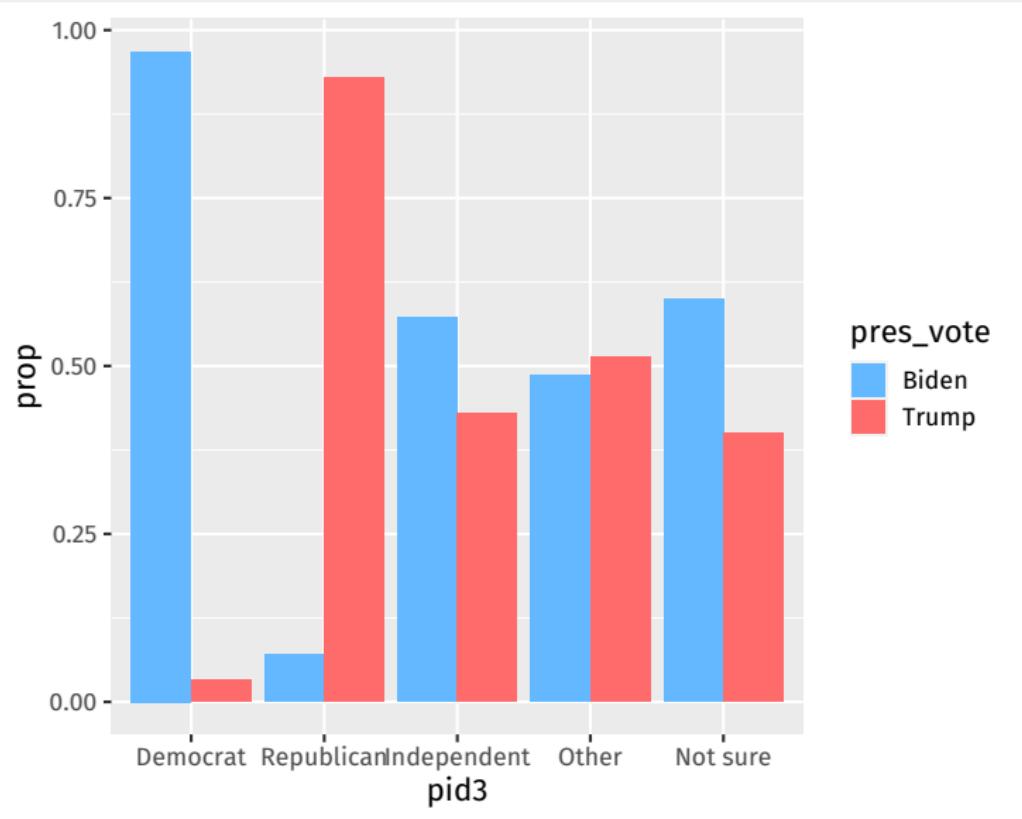
```
## # A tibble: 10 x 3
## # Groups: pid3 [5]
##   pid3      pres_vote    prop
##   <fct>      <chr>     <dbl>
## 1 Democrat    Biden    0.968
## 2 Democrat    Trump    0.0319
## 3 Republican Biden    0.0712
## 4 Republican Trump    0.929
## 5 Independent Biden    0.571
## 6 Independent Trump    0.429
## 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"))
```



Pivoting to create cross-tab

```
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

```
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`
```

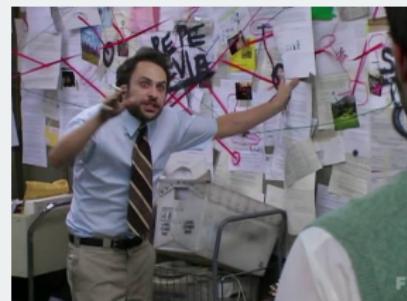
2/ Measurement

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

VZW Wi-Fi 18:23 33% gop.com

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.

[Large empty rectangular box for writing]

- **Bias:** systematic errors for all units in the same direction.
- individual measurement = exact value + bias + chance error.
- “What did you eat yesterday?”
~~ 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

litan 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 noted 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.
 - ↗ 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 ↵ **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 ↽ **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

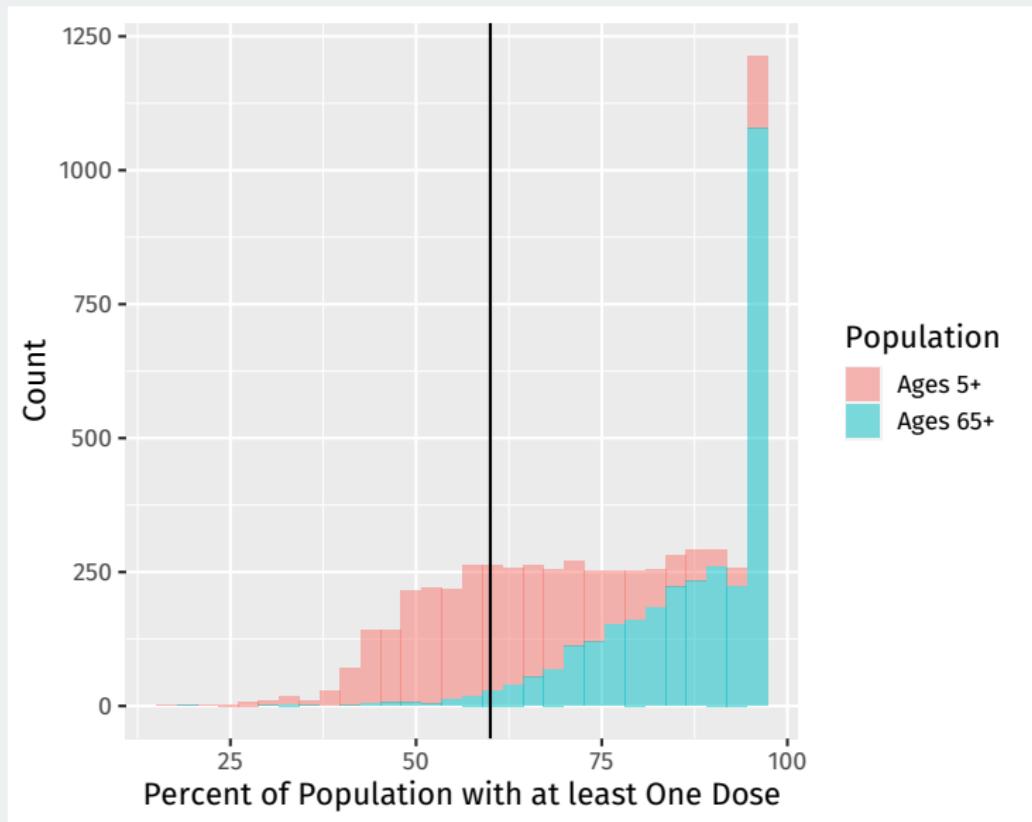
1/ Z-scores and standardization

COVID vaccination rates and votes

```
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     95.0    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        71.0     95.0    32.1    47.2
## 6 01067 Henry County AL       58.5     85.5    18.2    40.1
## 7 27037 Dakota County MN       81.0     95.0    49.5    46.9
## 8 27115 Pine County MN       56.5     85.0    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
```

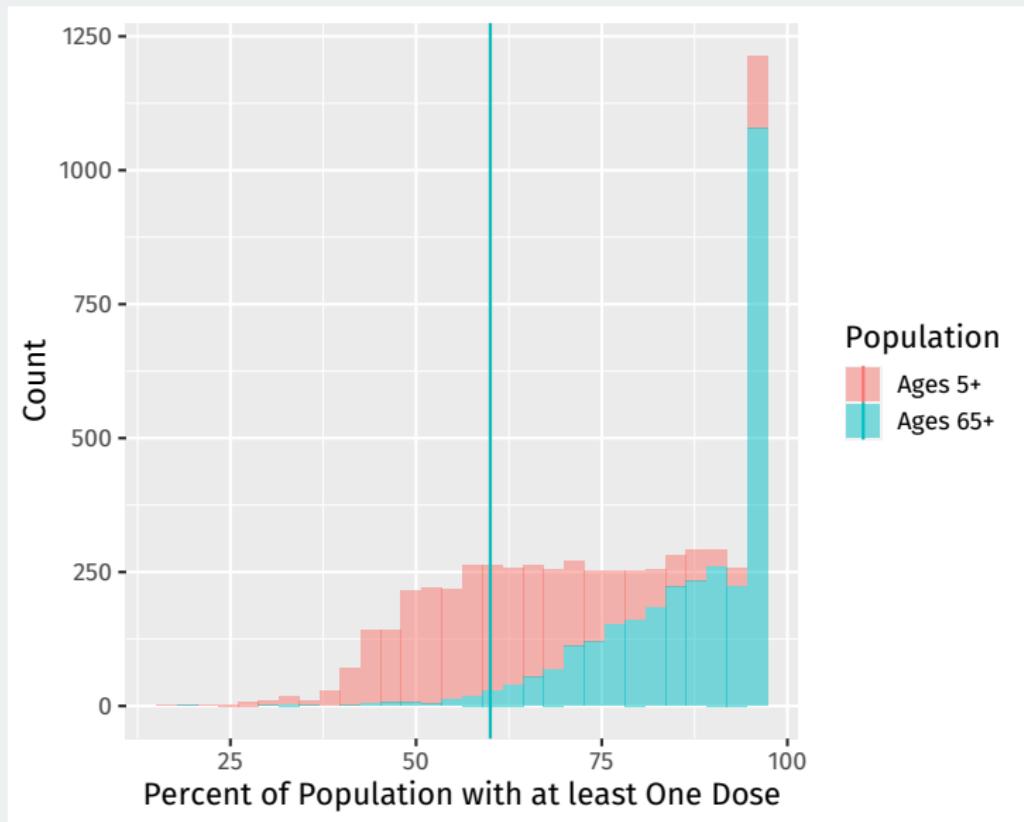
Is 60% vaccinated a lot?



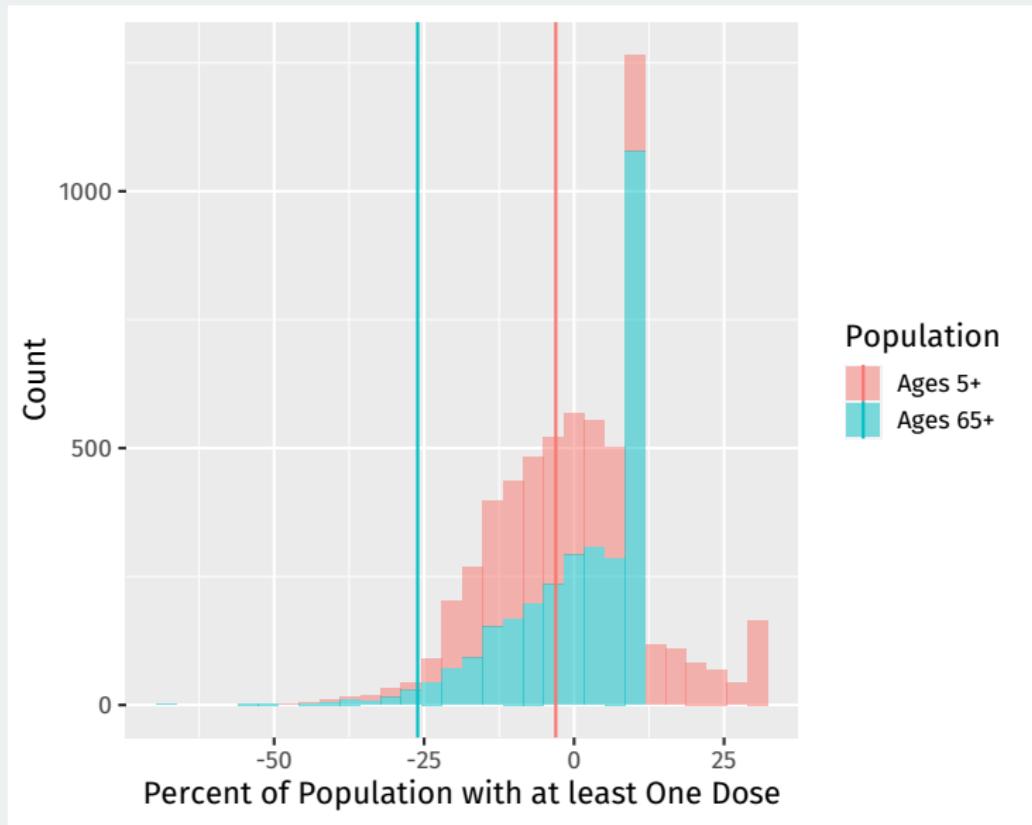
How large is large?

- 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.

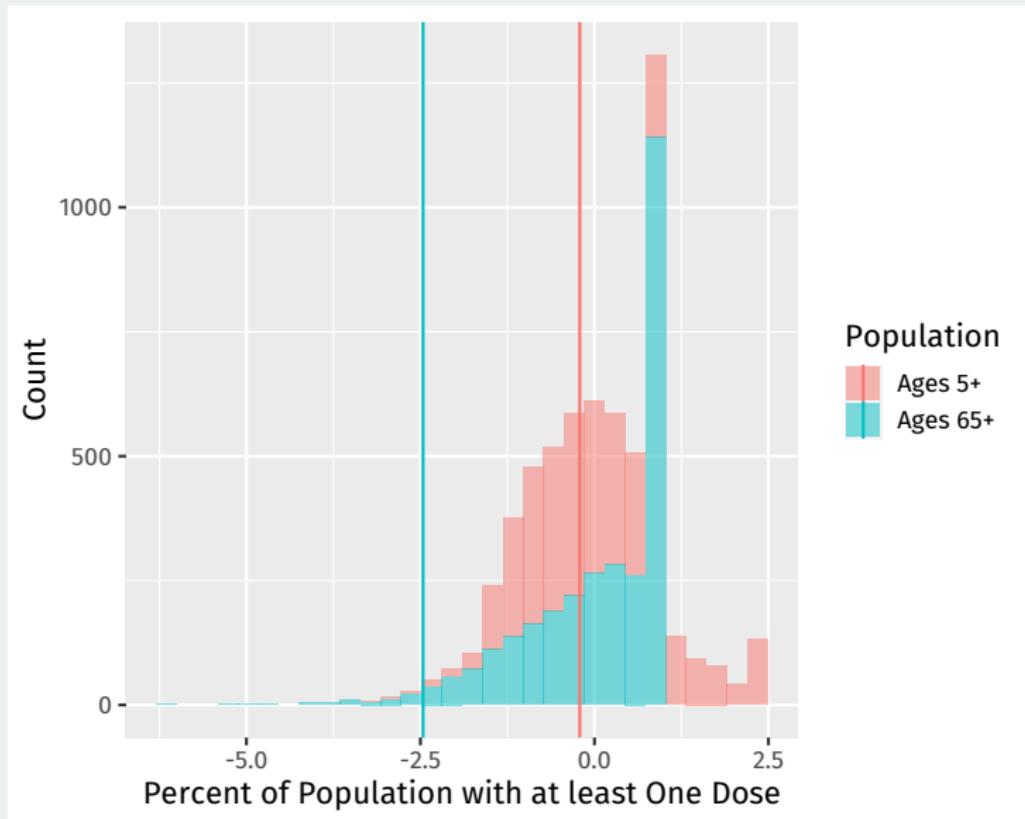
Original distributions



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

```
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_dose_centered  
##   <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

```
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_dose_z
##   <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

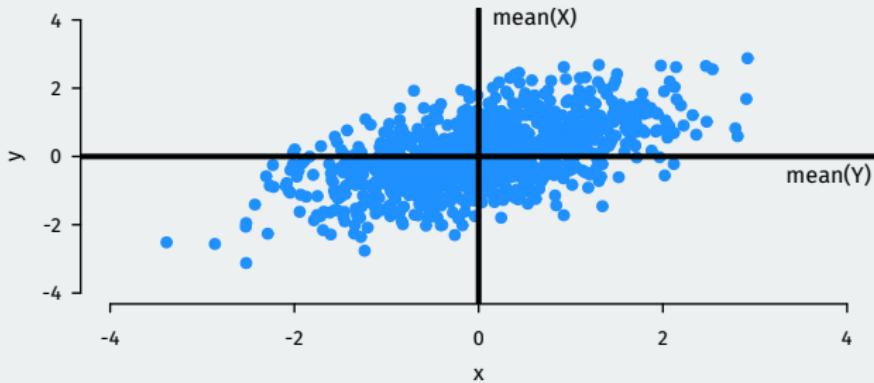
Correlation

- 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**:

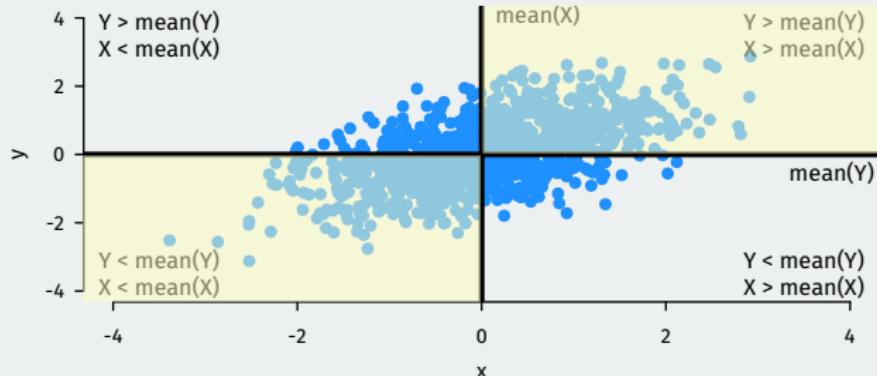
$$\frac{1}{n-1} \sum_{i=1}^n [(\text{z-score for } x_i) \times (\text{z-score for } y_i)]$$

- 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

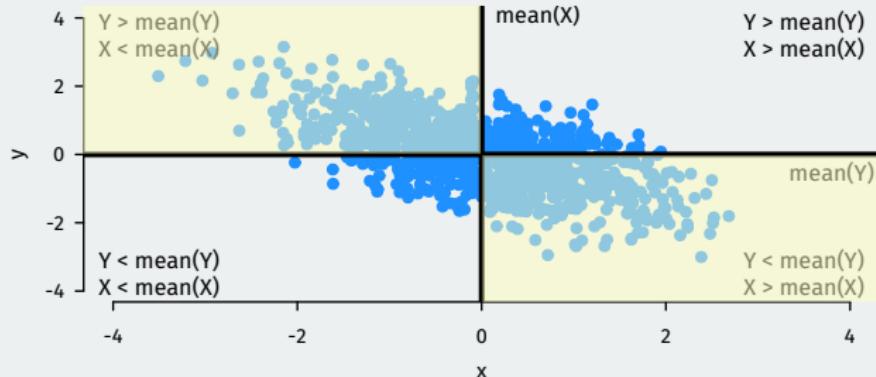


Correlation intuition



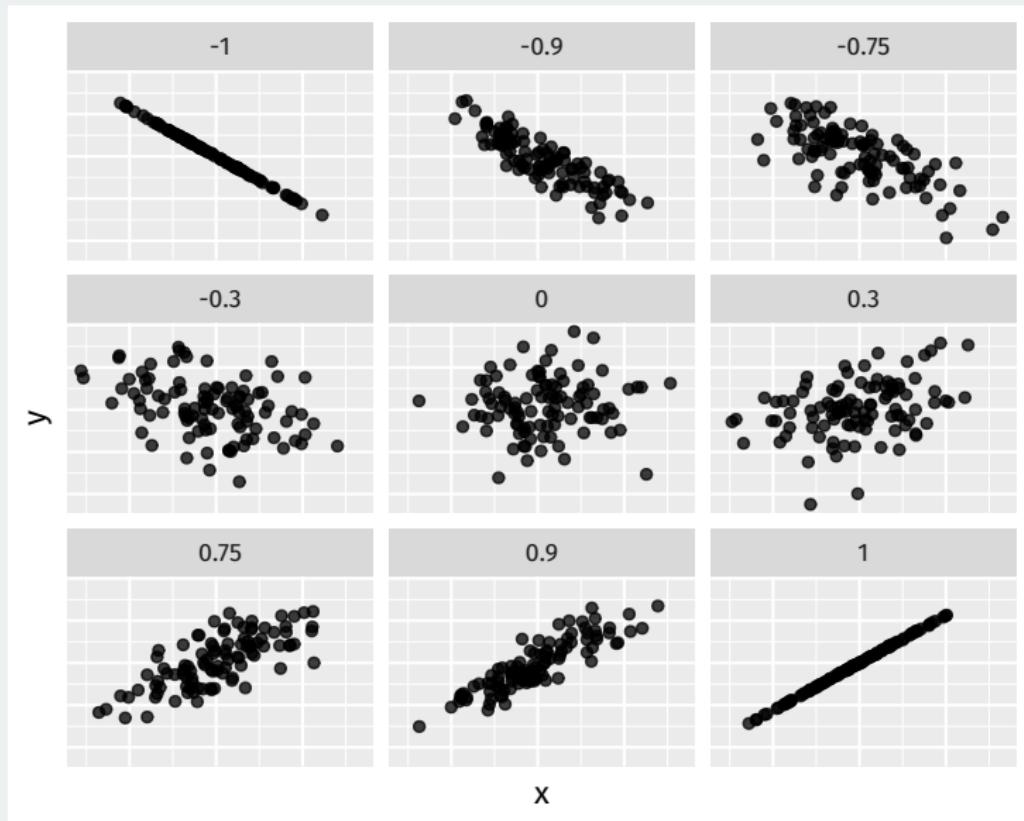
- Large values of X tend to occur with large values of Y :
 - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{pos. num.}) \times (\text{pos. num.}) = +$
- Small values of X tend to occur with small values of Y :
 - $(\text{z-score for } x_i) \times (\text{z-score for } y_i) = (\text{neg.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

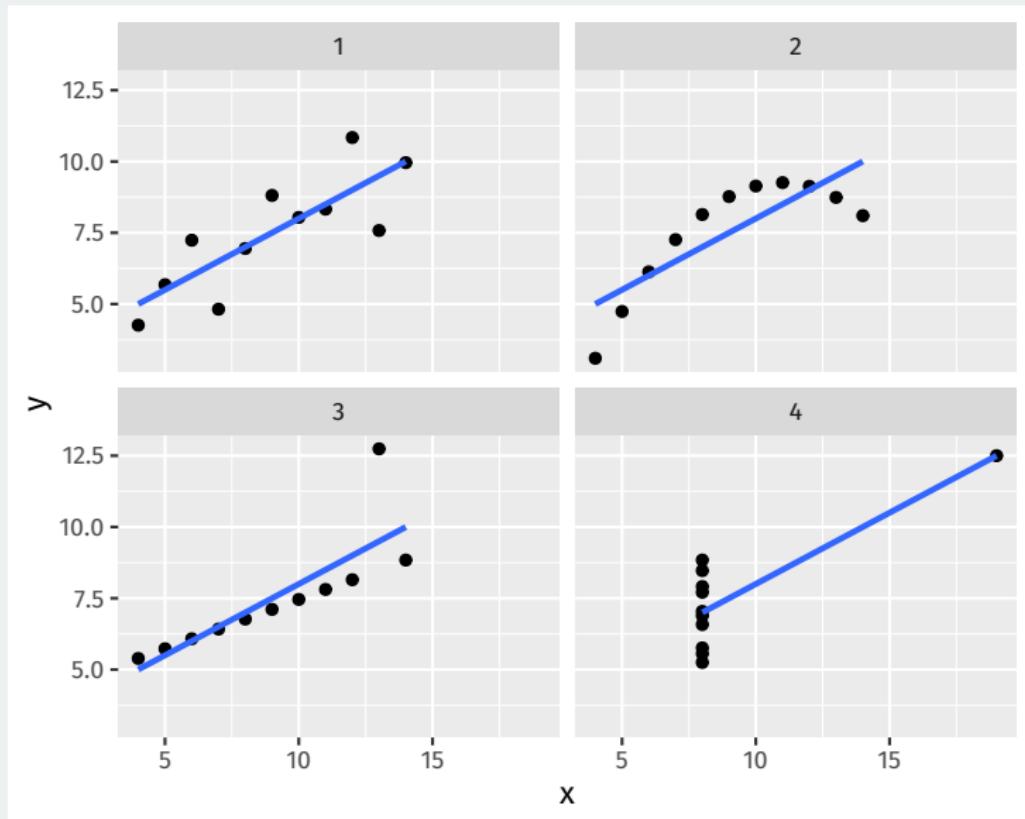
Correlation examples



Properties of correlation coefficient

- Correlation measures **linear** association.
- Order doesn't matter: $\text{cor}(x, y) = \text{cor}(y, x)$
- Not affected by changes of scale:
 - $\text{cor}(x, y) = \text{cor}(ax+b, cy+d)$
 - Celsius vs. Fahrenheit; 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
```

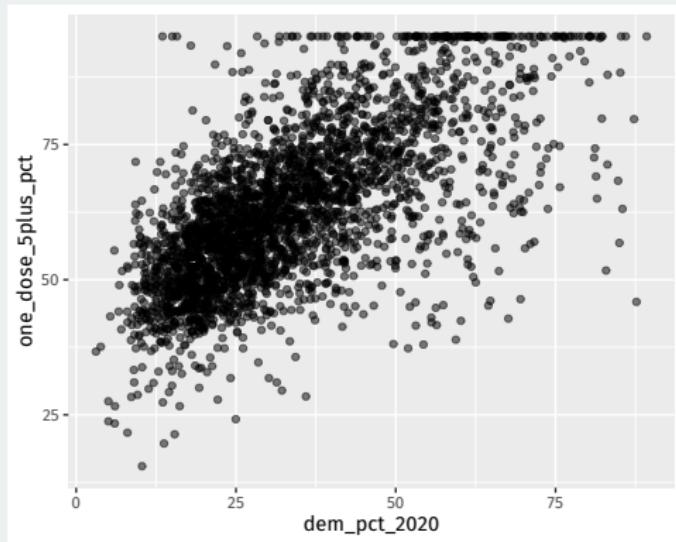
Missing values: set the `use = "pairwise"` → available case analysis

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

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

Comparing correlations

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

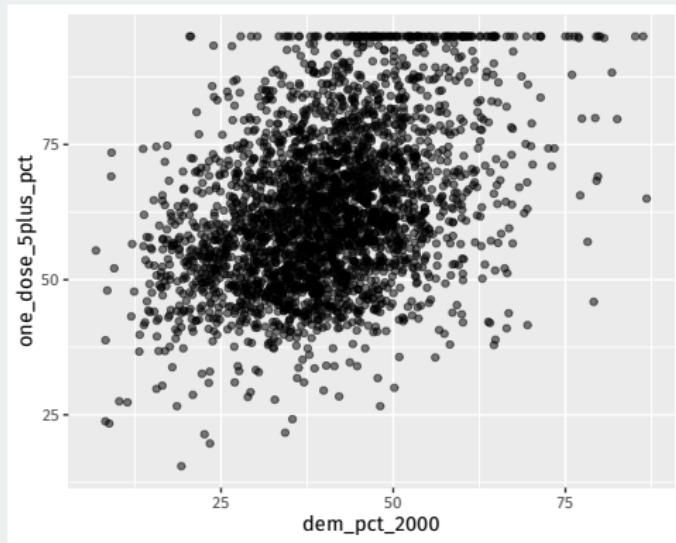


```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2020,  
  use = "pairwise")
```

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

Comparing correlations

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

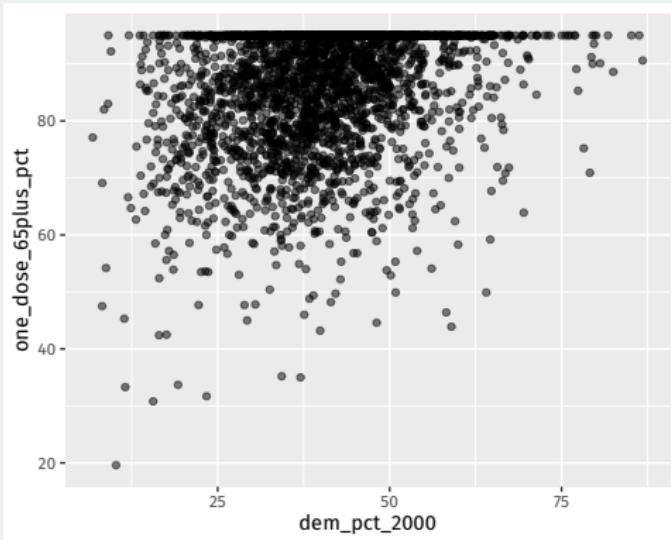


```
cor(covid_votes$one_dose_5plus_pct, covid_votes$dem_pct_2000,  
  use = "pairwise")
```

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

Comparing correlations

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



```
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:

```
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:

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

Check that it seems to work:

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

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

Cleaning up our code

```
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!

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

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

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

```
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

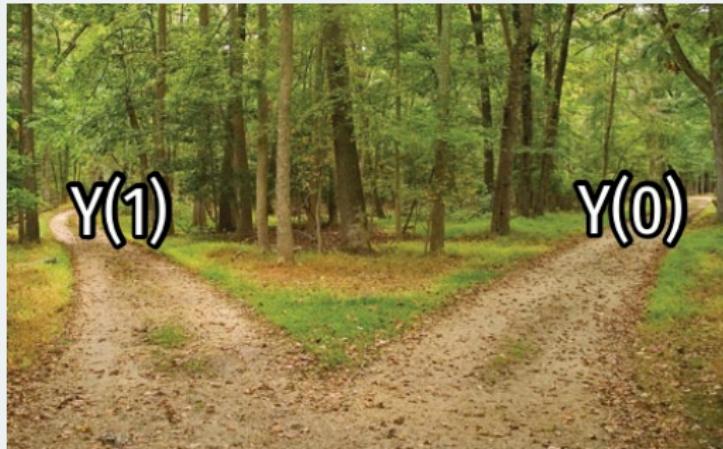
Harvard University

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

```
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

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

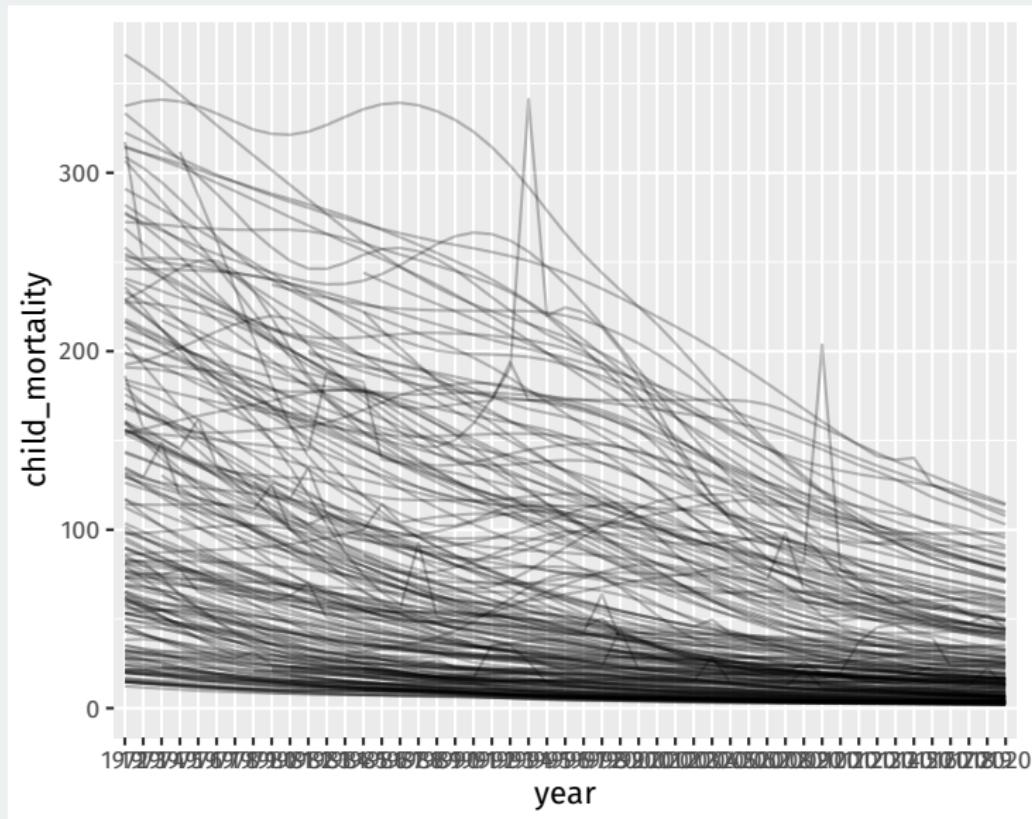
```
## # A tibble: 10,633 x 4
##   country country_code year child_mortality
##   <chr>    <chr>      <chr>        <dbl>
## 1 Aruba    ABW        1972         NA
## 2 Aruba    ABW        1973         NA
## 3 Aruba    ABW        1974         NA
## 4 Aruba    ABW        1975         NA
## 5 Aruba    ABW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

```
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:

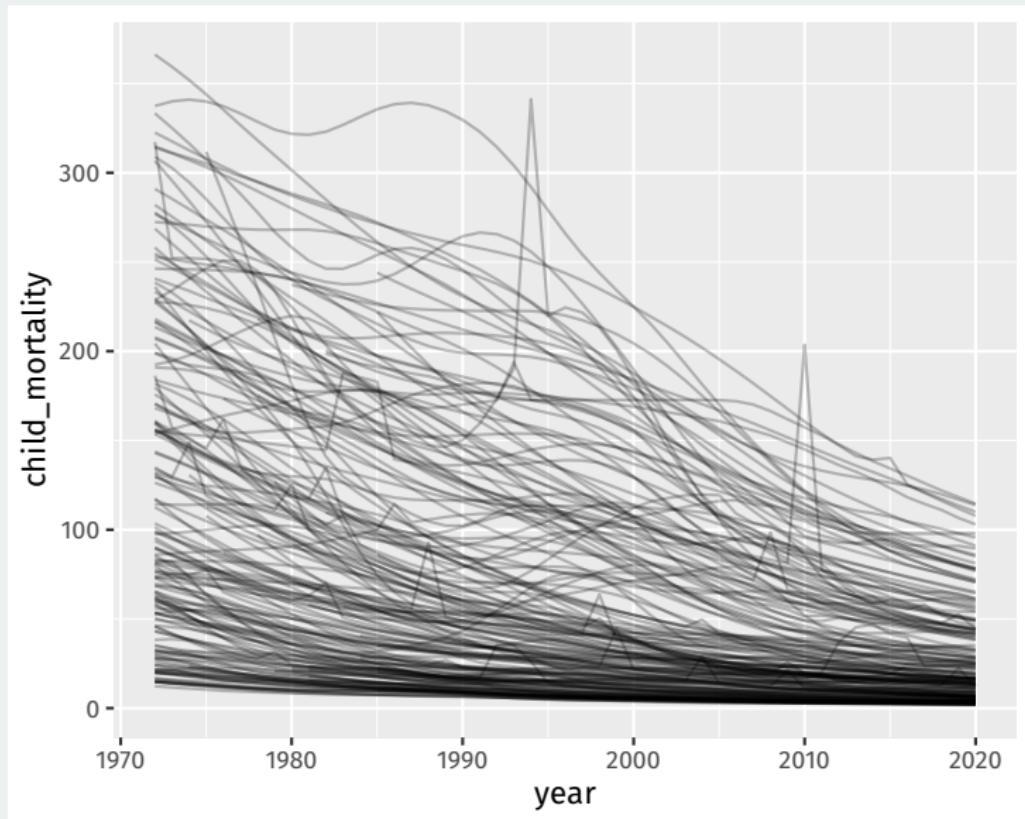
```
mortality_long <- mortality |>
  select(-indicator) |>
  pivot_longer(
    cols = `1972`:`2020`,
    names_to = "year",
    values_to = "child_mortality"
  ) |>
  mutate(year = as.integer(year))
mortality_long
```

```
## # 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!

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

There we go



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
## # ... with 25,470 more rows
```

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

```
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
```

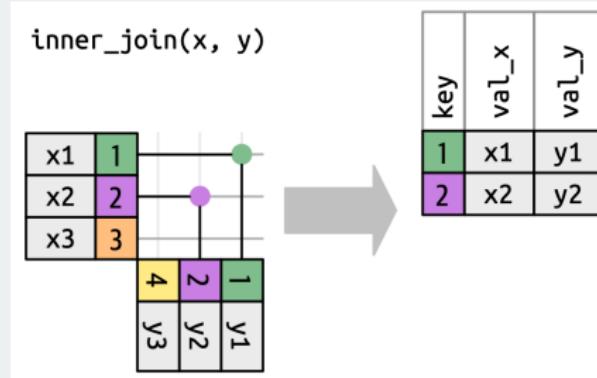
How inner joins work

Two data sets:

| x | y |
|---|----|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

| y | y1 |
|---|----|
| 1 | y1 |
| 2 | y2 |
| 4 | y3 |

Find matching keys:

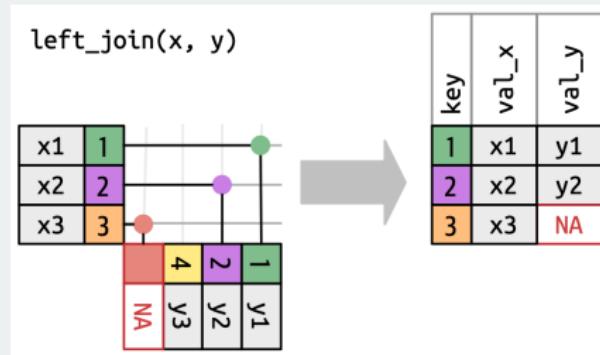


How left joins work

Two data sets:

| | x | y |
|---|----|----|
| 1 | x1 | y1 |
| 2 | x2 | y2 |
| 3 | x3 | y3 |

Keep all x keys:



More complicated example

```
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?

```
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

```
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

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

## Joining, by = "tailnum"

## # A tibble: 336,776 x 10
##   year time_hour      origin dest tailnum carrier manufacture~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

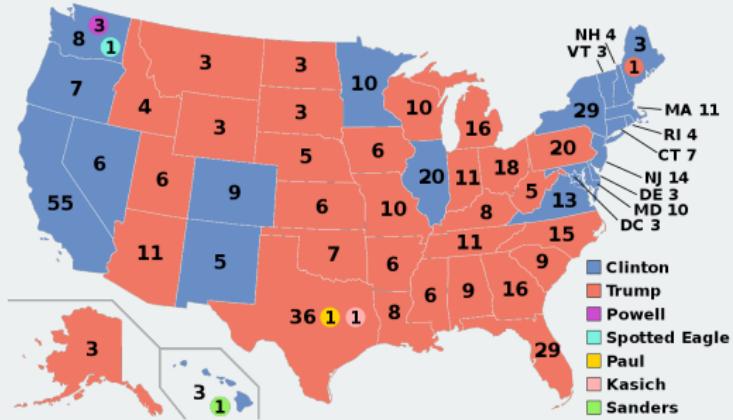
1/ Prediction

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
 - $538 = 435 \text{ (House of Representatives)} + 100 \text{ (Senators)} + 3 \text{ (DC)}$
 - 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?

```
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

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

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

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

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

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

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

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

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

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

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

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

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

Subsetting with brackets

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

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

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

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

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

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

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

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

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

Manually changing values, alternative

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

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

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

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

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

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

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

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

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

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

```
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

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

```
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

```
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

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

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

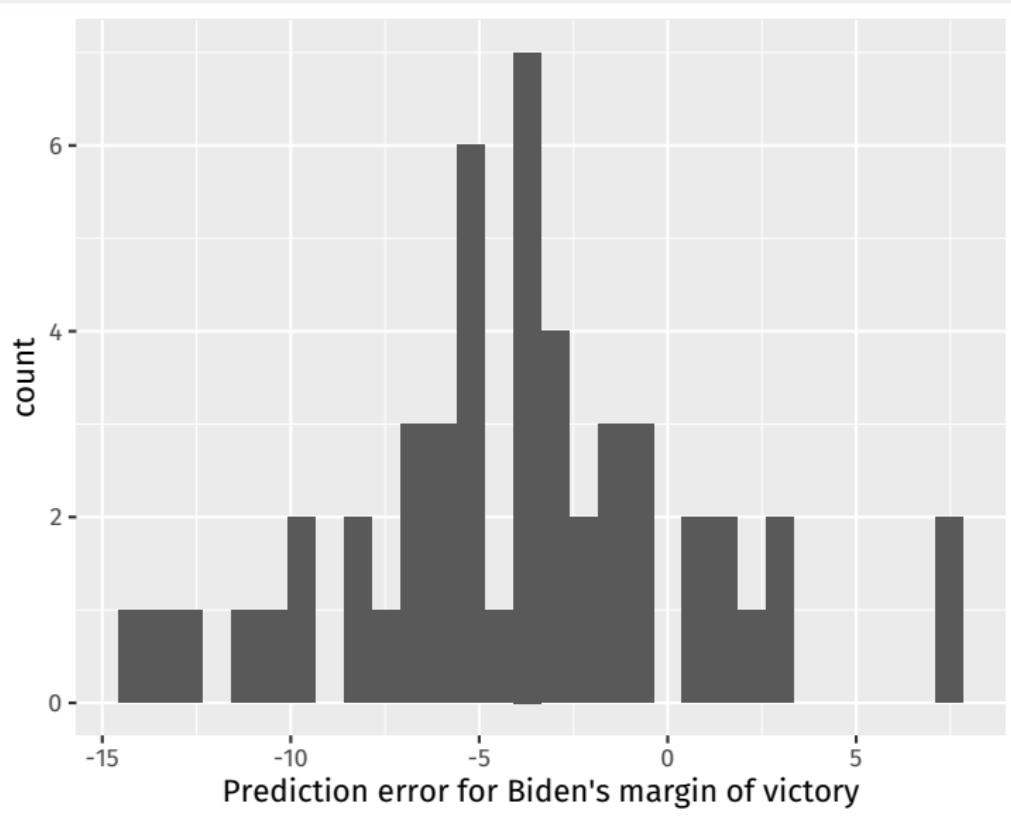
Root mean-square error: average magnitude of the prediction error

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

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

Histogram of the errors

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

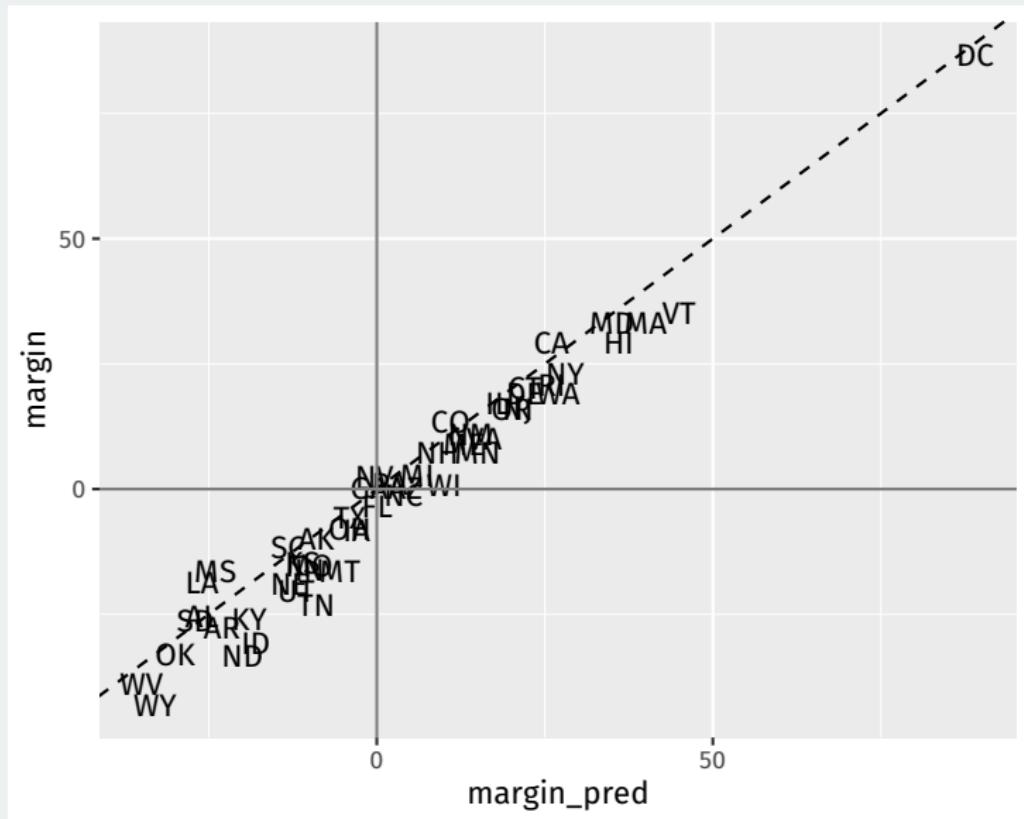


Comparing polls to outcome

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

```
## 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:

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

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

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

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

- Prediction of binary outcome variable = **classification problem**
- Wrong prediction ↽ 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
```

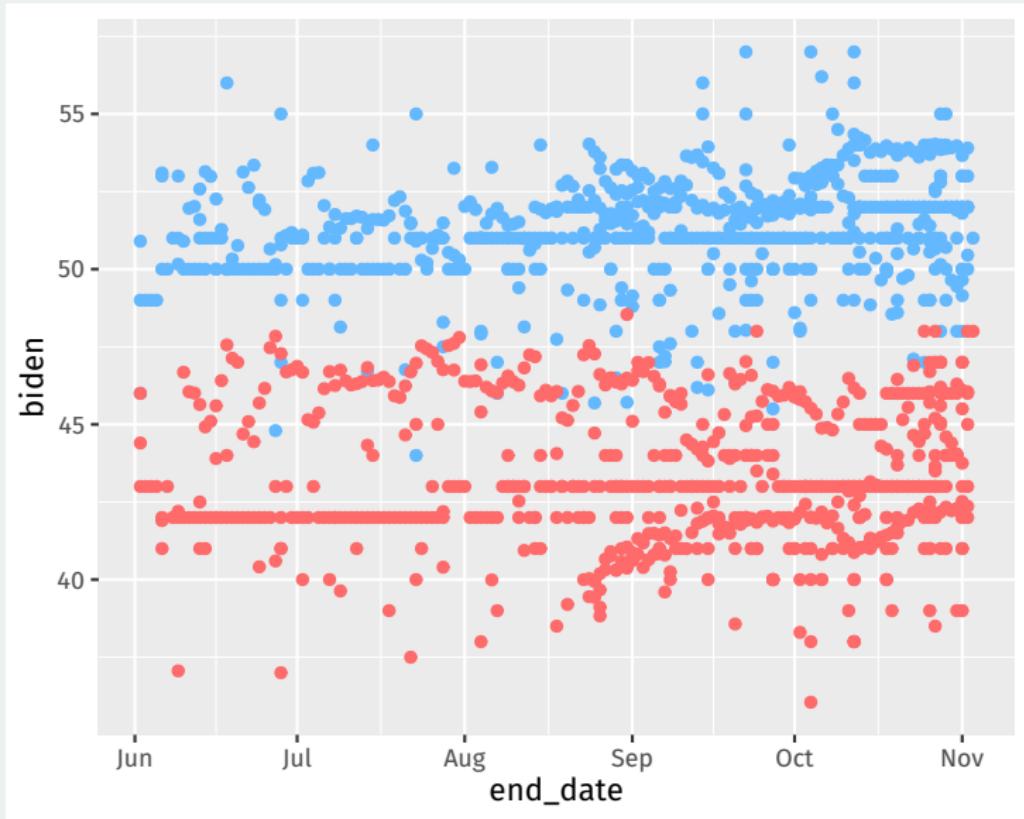
```
## # A tibble: 654 x 5
##   end_date   pollster      sampl~1 biden trump
##   <date>     <chr>        <dbl>  <dbl> <dbl>
## 1 2020-11-03 Lake Research    2400    51    48
## 2 2020-11-02 Research Co.   1025    50    42
## 3 2020-11-02 YouGov       1363    53    43
## 4 2020-11-02 Ipsos         914     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

```
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:

```
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"
```

```
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 convert 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:

```
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

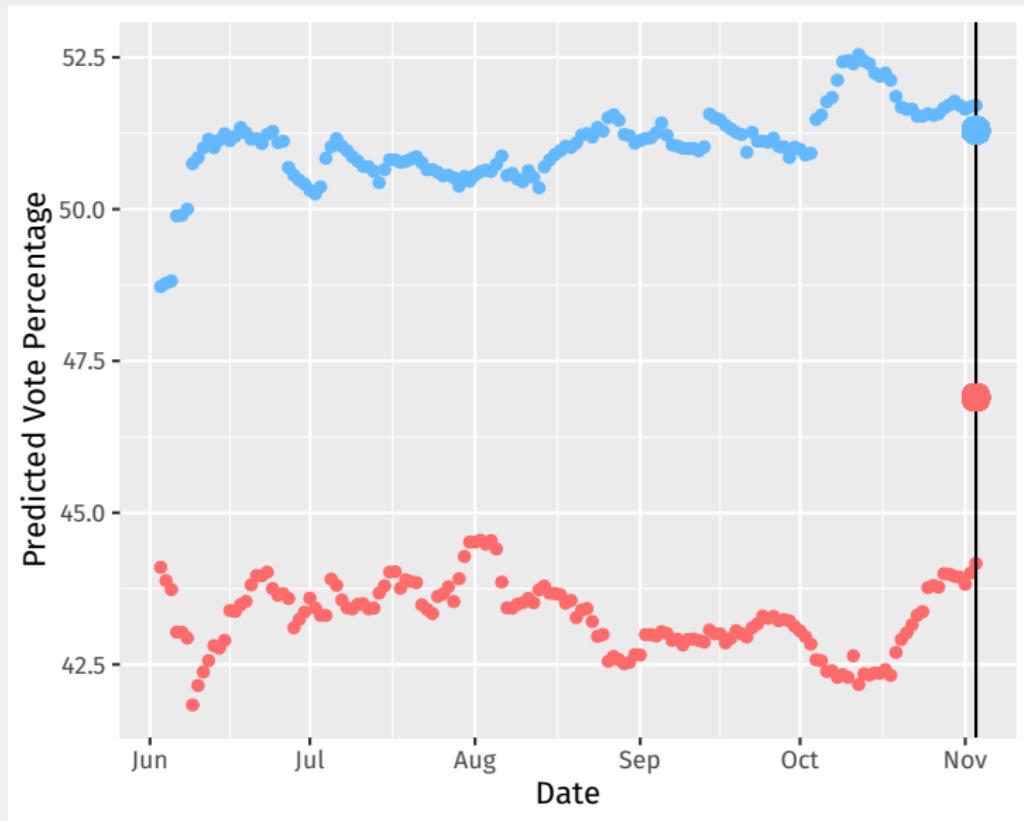
output

```
## # 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

```
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 = 1) +
  geom_point(aes(x = election_day, y = 46.9), color = "indianred1", size = 1) +
  labs(
    x = "Date",
    y = "Predicted Vote Percentage"
  )
```

Let's plot



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

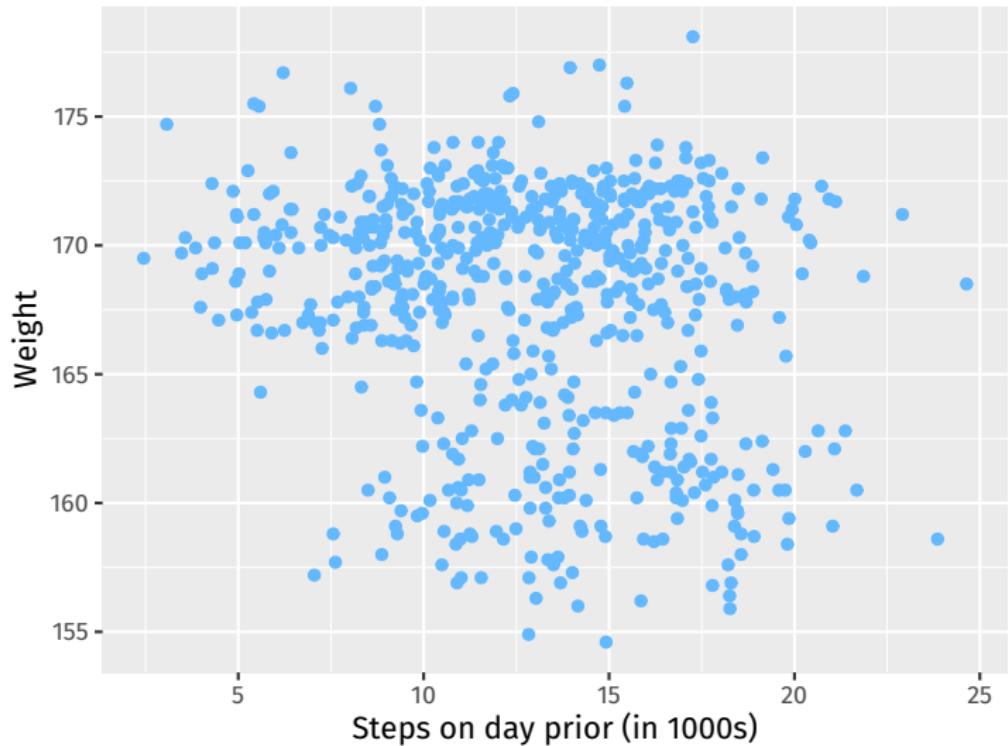
- Load the data:

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

- Plot the data:

```
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



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:

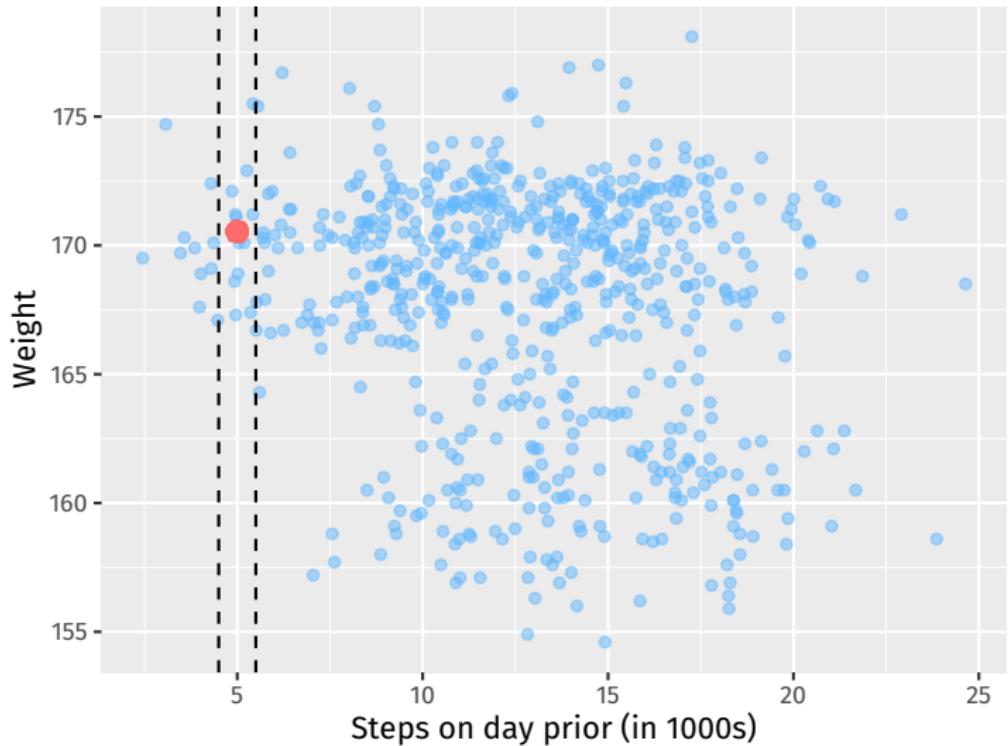
```
mean_wt_5k_steps <- health |>  
  filter(round(steps_lag) == 5) |>  
  summarize(mean(weight)) |>  
  pull()  
mean_wt_5k_steps
```

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

Plotting the best guess

```
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

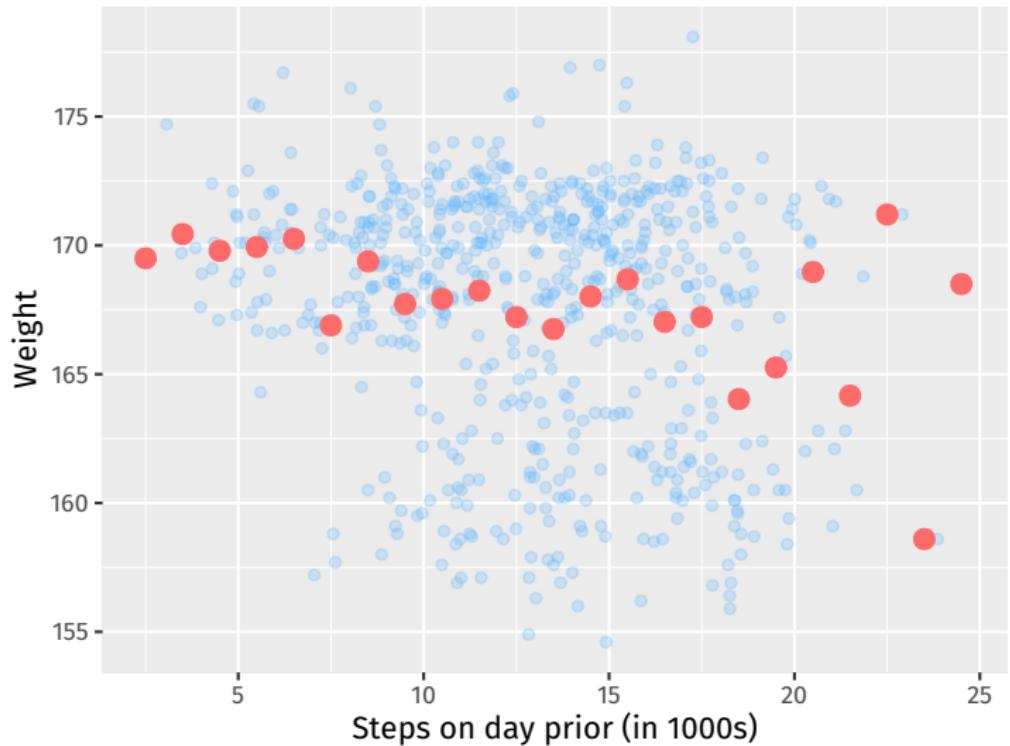


Binned means

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

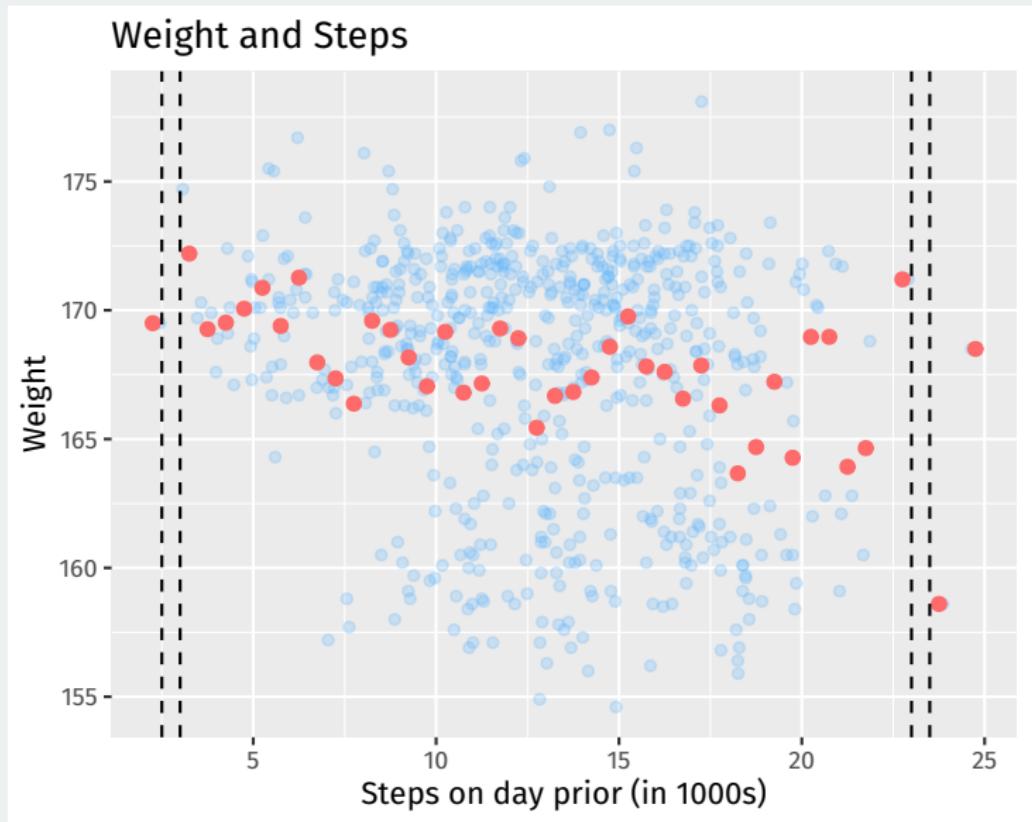


Smaller bins

But what happens when we make the bins too small?

```
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.

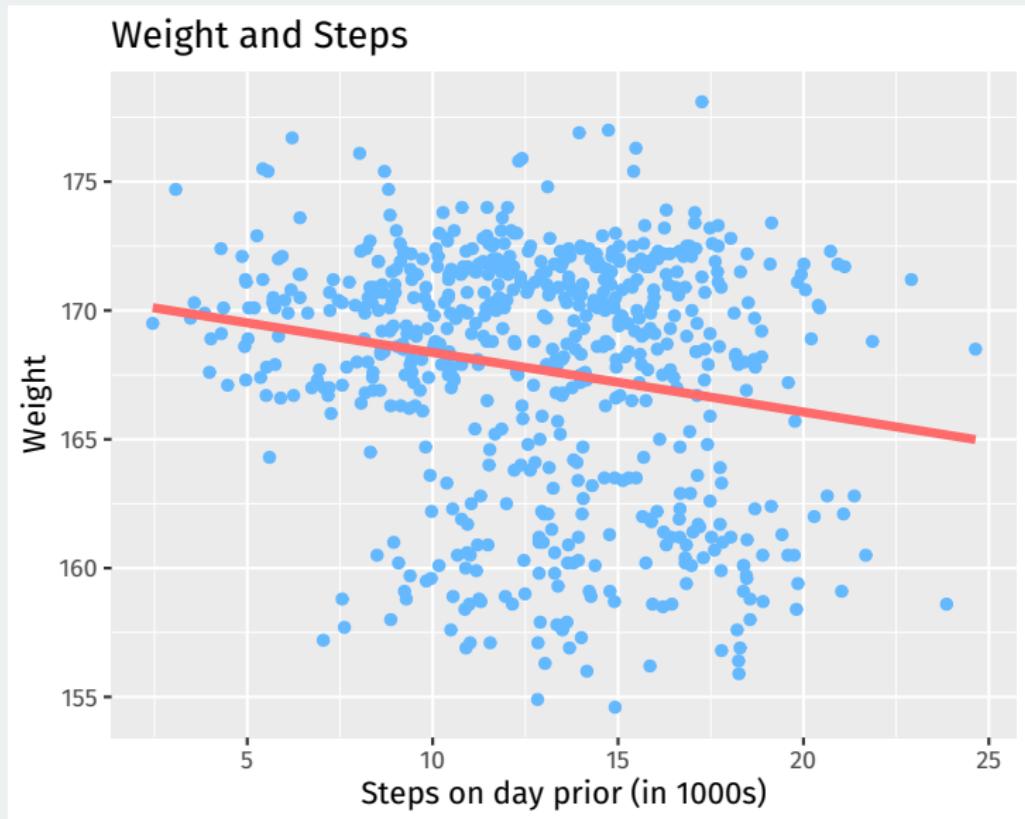
Estimated coefficients

- 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 .

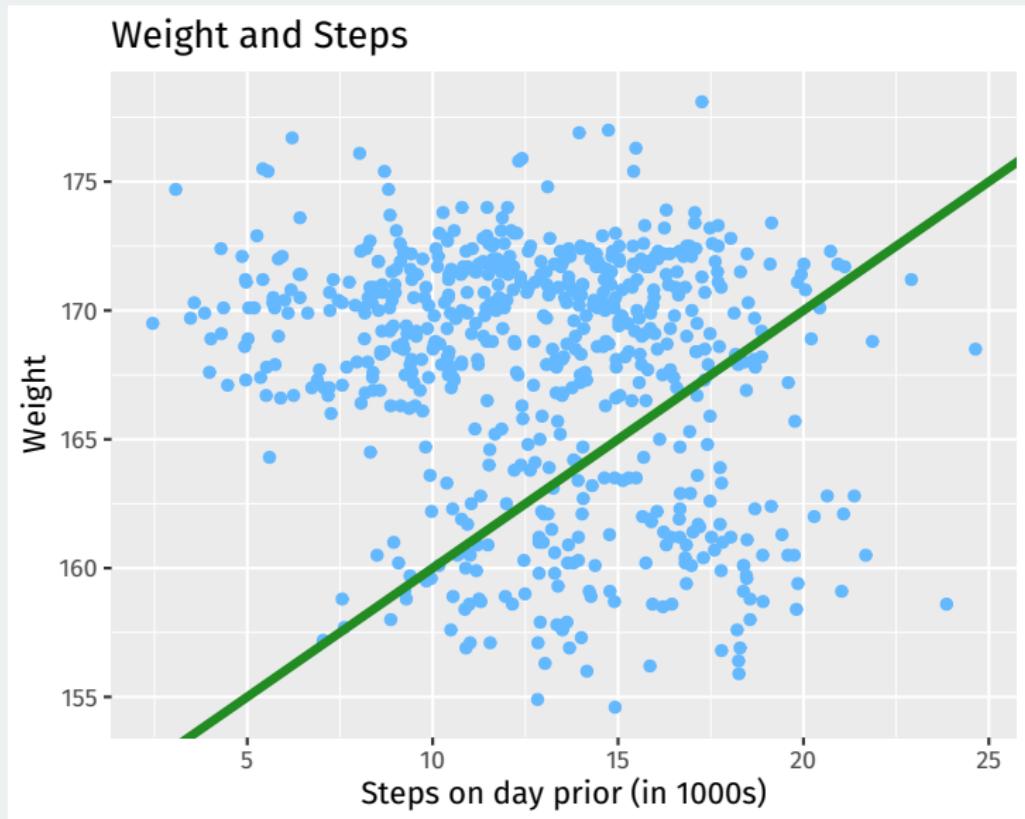
Line of best fit

```
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 .

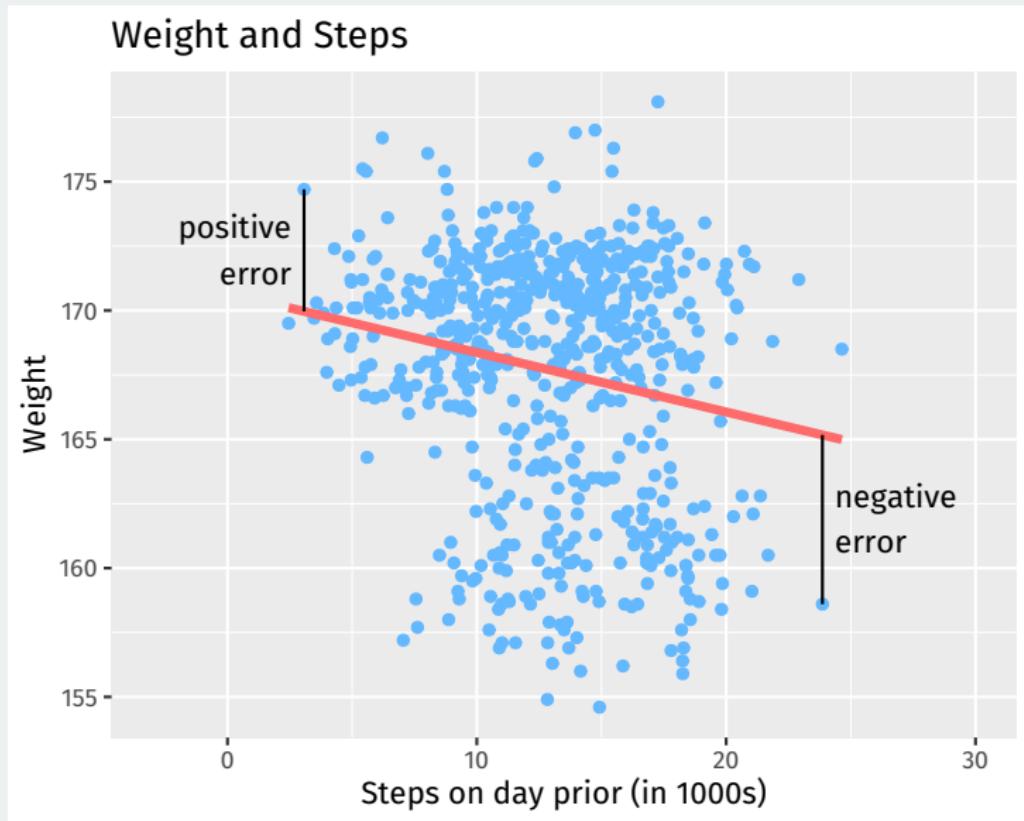
Fitted/predicted value for unit i :

$$a + b \cdot X_i$$

Preditction error (residual):

$$\text{error} = \text{actual} - \text{predicted} = Y_i - (a + b \cdot X_i)$$

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 variable
 - `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()
```

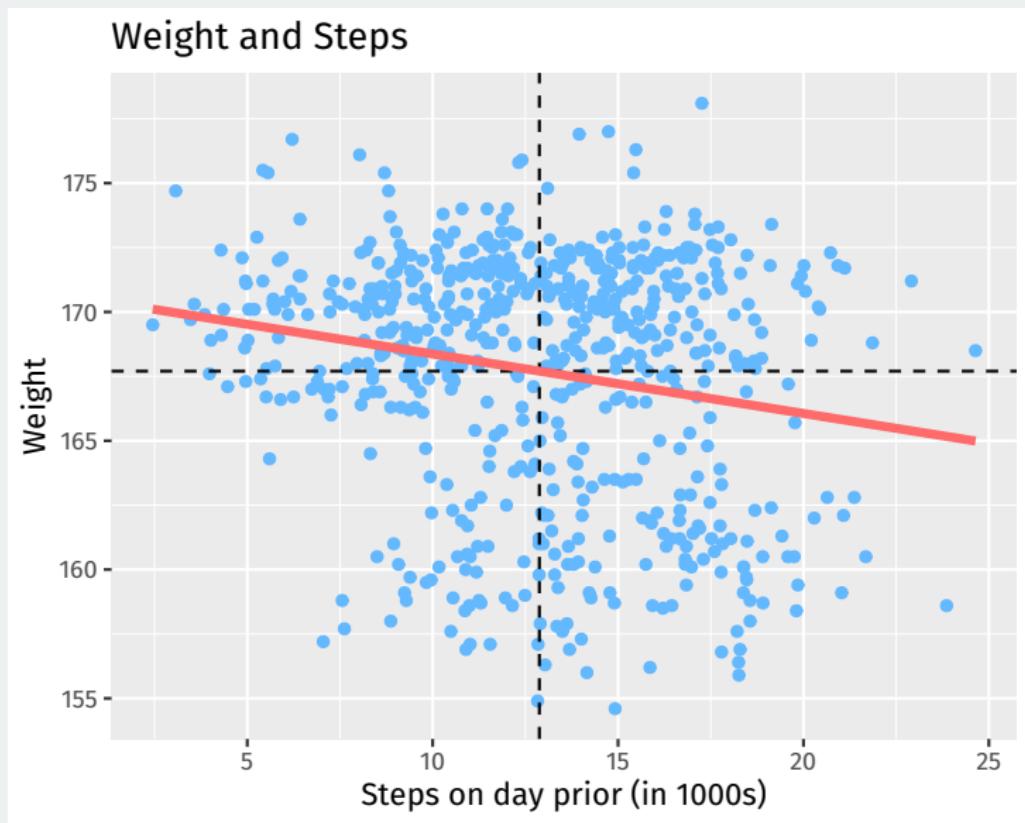
```
## # A tibble: 6 x 8
##   weight steps_lag .fitted   .resid     .hat .sigma   .cooksdf
##   <dbl>     <dbl>   <dbl>    <dbl>    <dbl>   <dbl>    <dbl>
## 1 169.      17.5    167.    2.46    0.00369  4.68  5.13e-4
## 2 168.      18.4    166.    1.57    0.00463  4.68  2.64e-4
## 3 167.      19.6    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 (\bar{X}, \bar{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 (\bar{X}, \bar{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.

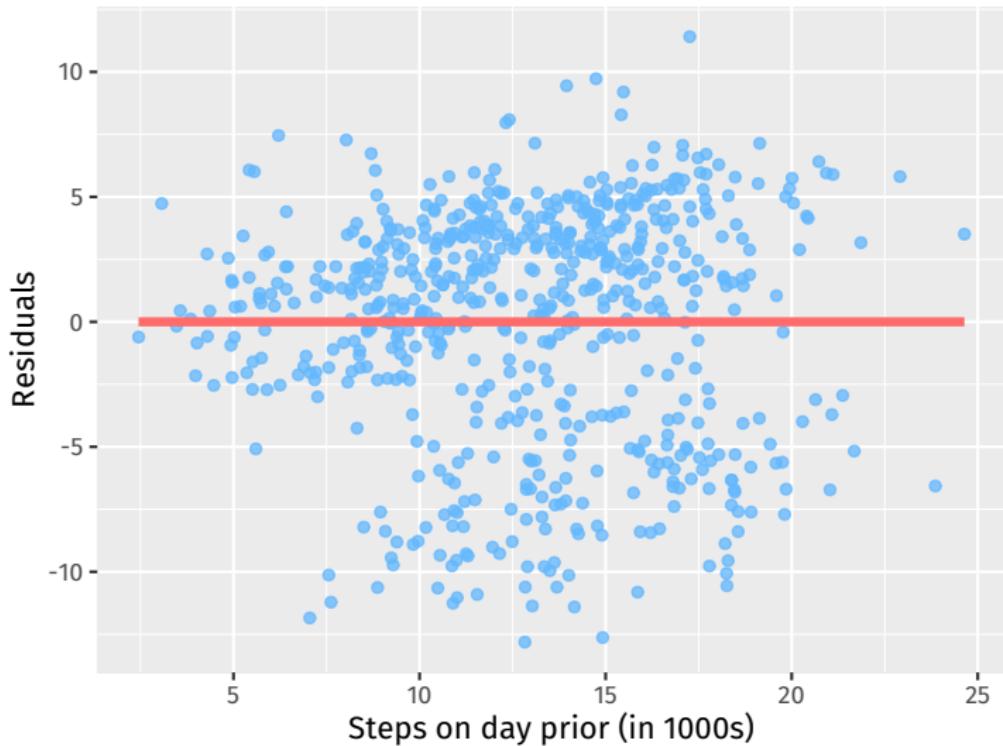
```
augment(fit) |>  
  summarize(mean(.resid))
```

```
## # A tibble: 1 x 1  
##   `mean(.resid)`  
##       <dbl>  
## 1      -1.21e-13
```

Plotting the residuals

```
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

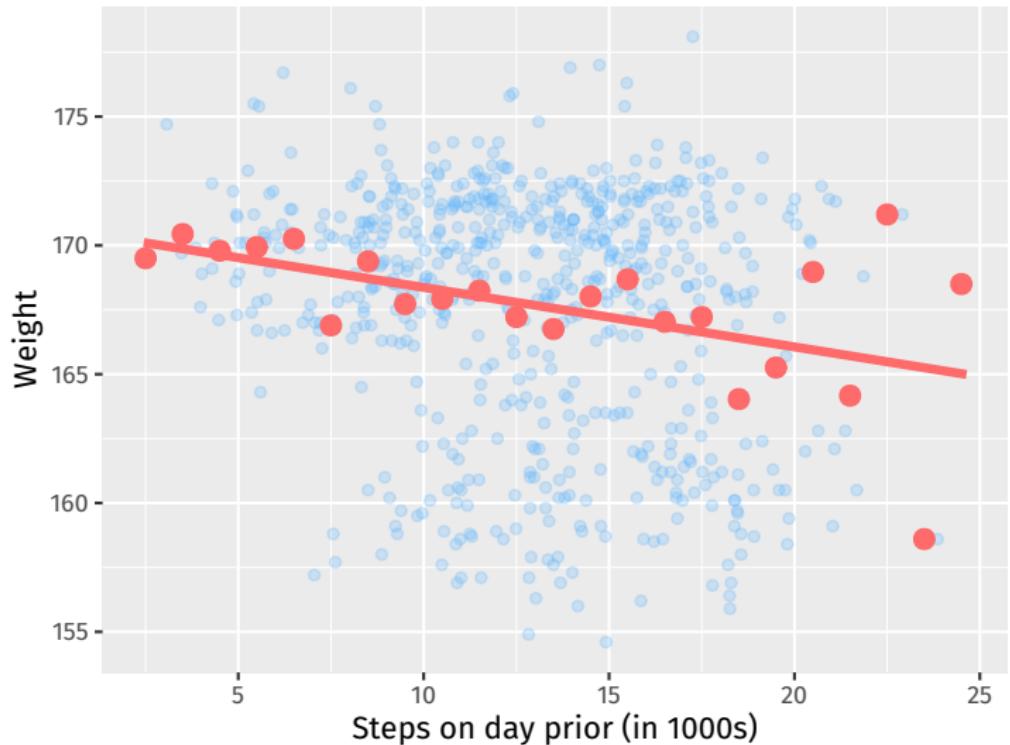


Smoothed graph of averages

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



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 |

```
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:

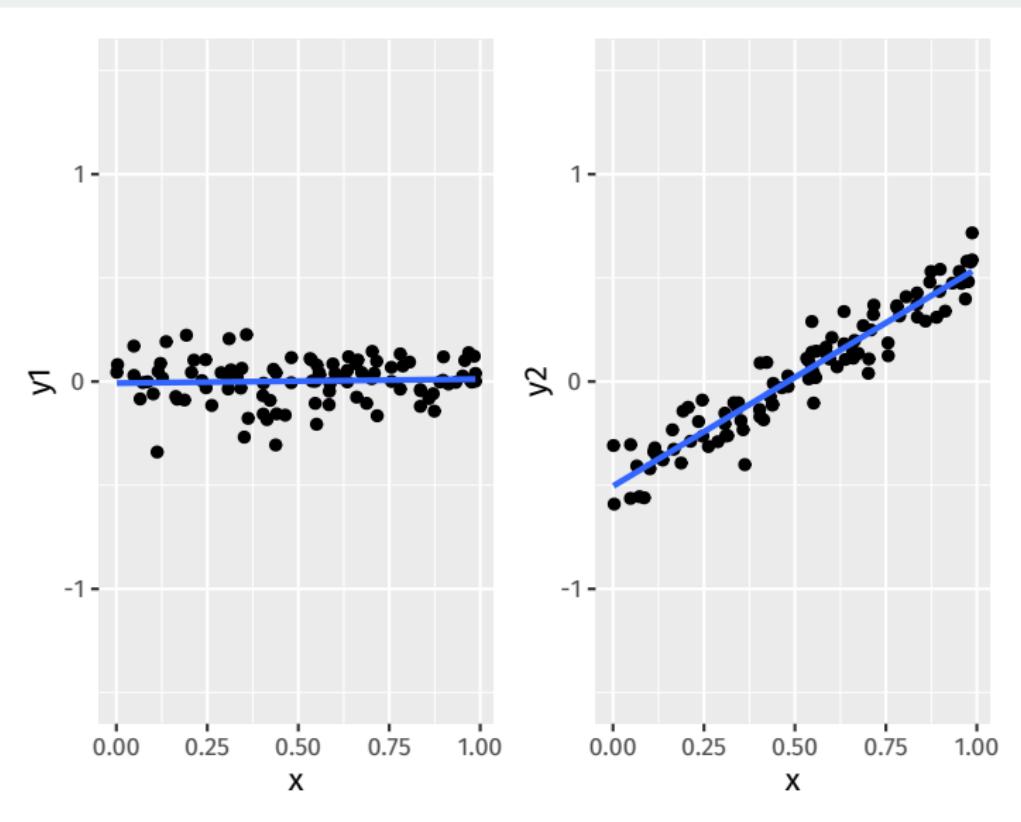
$$\text{prediction error} = \sum_{i=1}^n (\text{actual}_i - \text{predicted}_i)^2$$

Prediction error for regression: **Sum of squared residuals**

$$\text{SSR} = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

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**:

$$\frac{\text{reduction in prediction error using model}}{\text{baseline prediction error}}$$

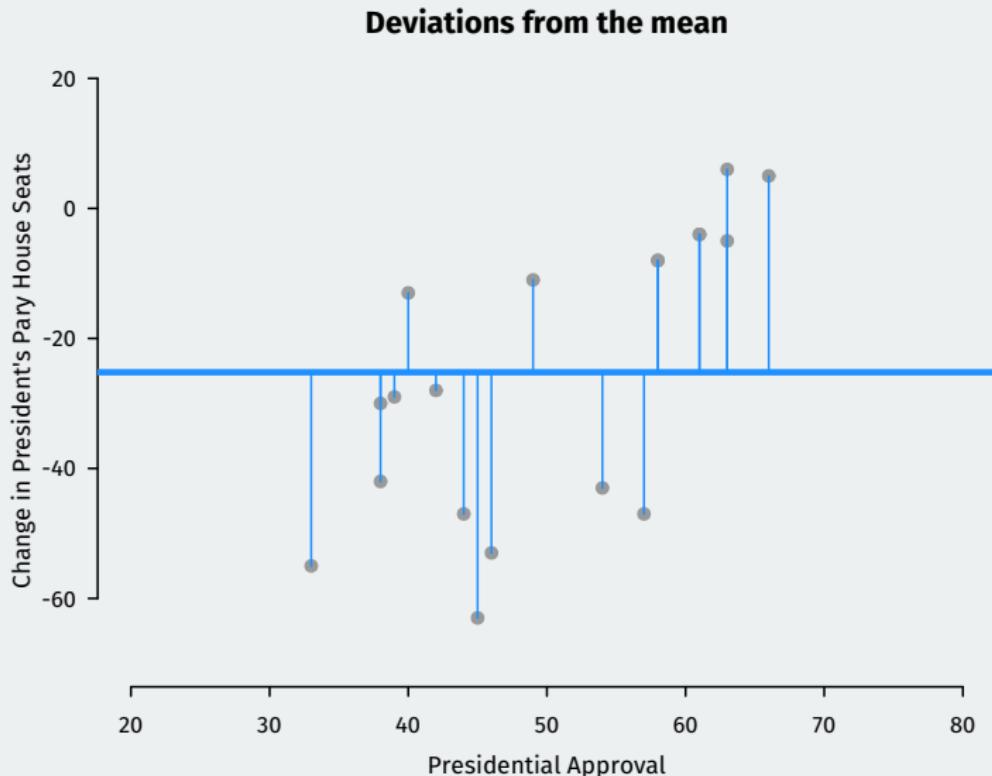
Baseline prediction error without a regression is using the mean of Y to predict. This is called the **Total sum of squares**:

$$\text{TSS} = \sum_{i=1}^n (Y_i - \bar{Y})^2$$

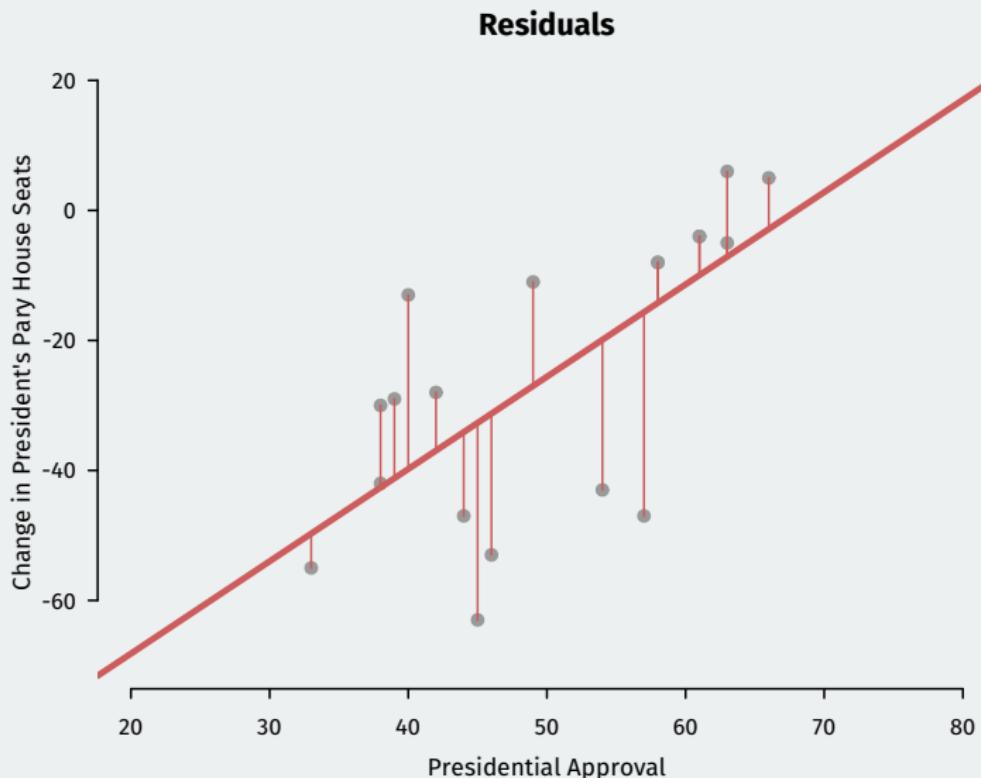
Leads to the **coefficient of determination**, R^2 , one summary of LS model fit:

$$R^2 = \frac{TSS - SSR}{TSS} = \frac{\text{how much smaller LS prediction errors are vs mean prediction error using the mean}}{\text{prediction error using the mean}}$$

Total SS vs SSR



Total SS vs SSR



Model fit in R

- To access R^2 from the `lm()` output, use the `summary()` function:

```
fit.app.sum <- summary(fit.app)
fit.app.sum$r.squared
```

```
## [1] 0.45
```

- Compare to the fit using change in income:

```
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`:

```
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>
## 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
```

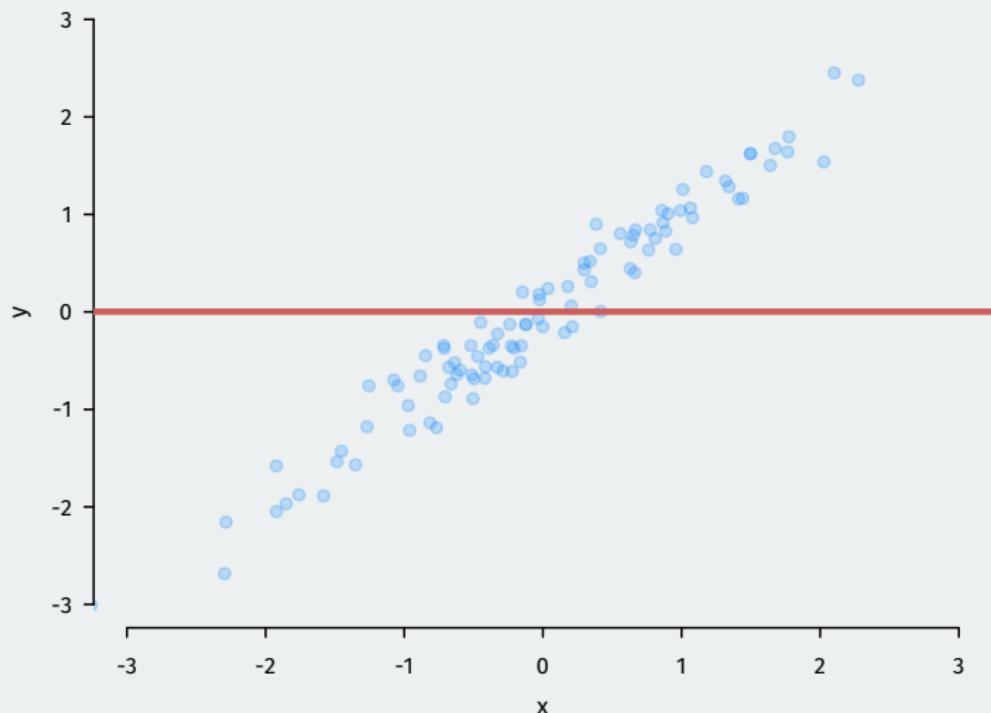
Fake data, better fit

- Little hard to see what's happening in that example.
- Let's look at fake variables x and y :

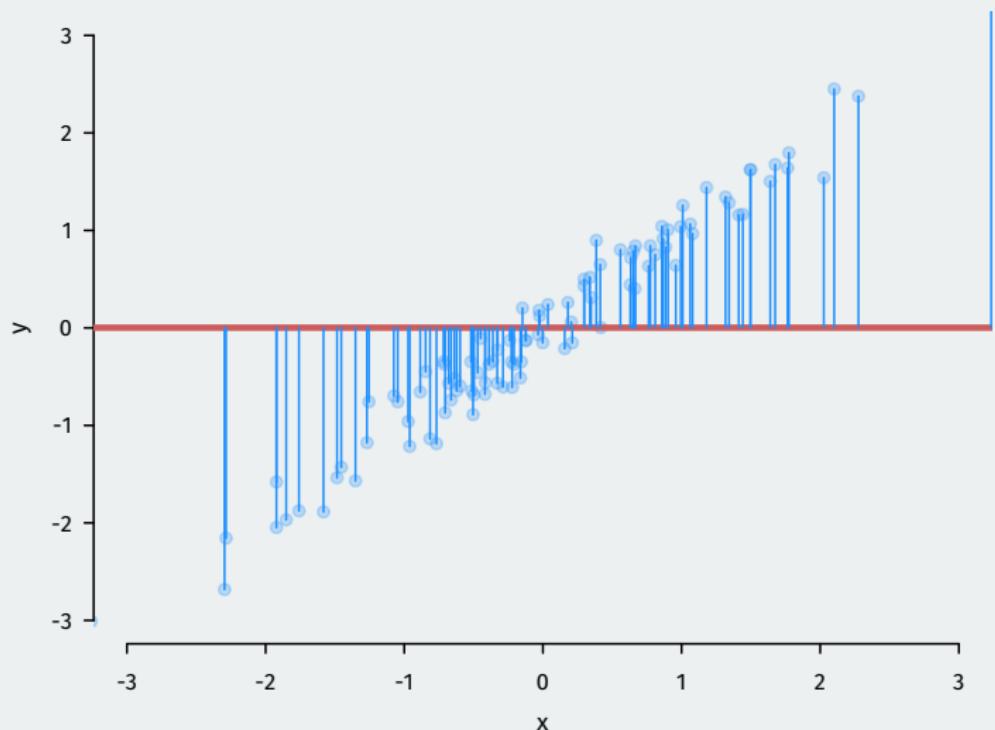
```
fit.x <- lm(y ~ x)
```

- Very good model fit: $R^2 \approx 0.95$

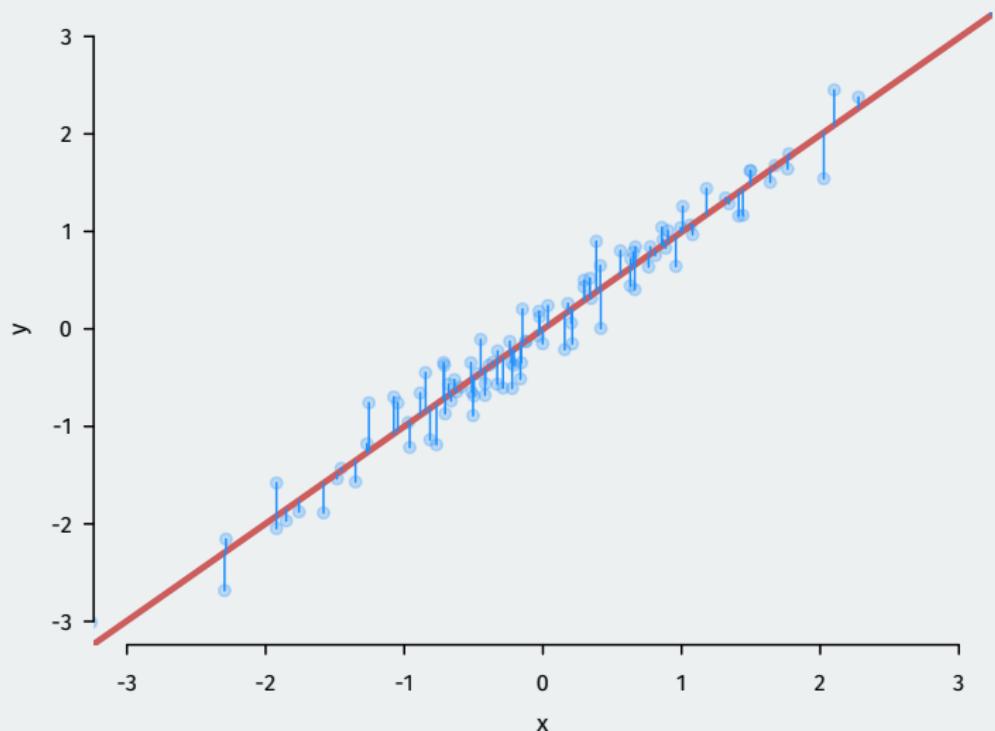
Fake data, better fit



Fake data, better fit

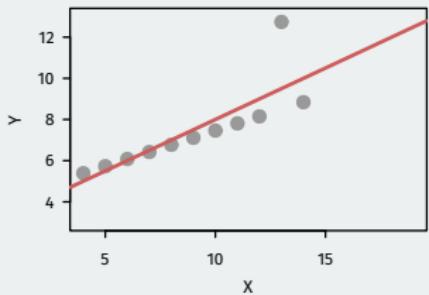
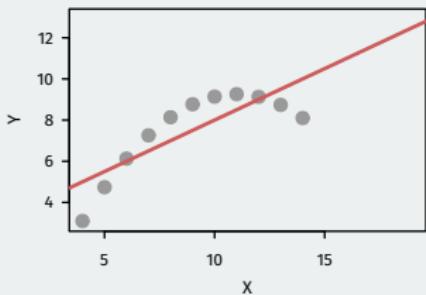
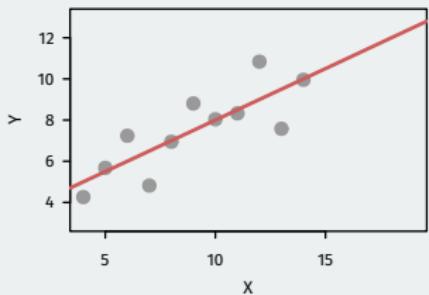


Fake data, better fit



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?

$$\text{seat_change}_i = \alpha + \beta_1 \text{approval}_i + \beta_2 \text{rdi_change}_i + \epsilon_i$$

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

```
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:

$$Y_i - \hat{Y}_i = \text{seat_change}_i - \hat{\alpha} - \hat{\beta}_1 \text{approval}_i - \hat{\beta}_2 \text{rdi_change}_i$$

- 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 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

```
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
```

```
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
```