

# Polling data and the upcoming election

Week 8, Lecture 14

<https://live.ds306.org>

October 22, 2026

# A blue wave in November?

---

- “Will the midterms be a ‘blue wave’ for Democrats?” [WBUR / Here & Now, October 15, 2026](#)
- “Midterms a ‘referendum on Trump’ that could bring a ‘blue wave’ of Democrat wins” [ABC News, October 11, 2026](#)

(A “blue wave” means large Democratic gains across many races.)

# What would convince you?

---

“The big unknown behind Democrats’ polling leads” [Axios, October 10, 2026](#)

- How often can random sampling put the eventual loser ahead?
- What changes if some groups are more likely to vote?
- How can we investigate a polling process that is too complicated to work out by hand?

Over the next two weeks, we’ll build simulations to investigate these questions.

# Skills we'll focus on for the next two weeks

---

- Iteration (for loops and vectorization)
- Probability simulations and modeling
- Debugging code

# Were the polls wrong?

---

Michigan, just before the 2024 presidential election:

- Research Co. put Harris **2 percentage points** ahead: 49% to 47%.
- Its reported margin of error was  **$\pm 4.6$  percentage points** for a candidate's share.
- Trump won the state by **1.41 percentage points**.

Poll, November 2–3 • Certified election result

# Other late Michigan polls

| Pollster     | Field dates, 2024 | Responses | Harris's lead |
|--------------|-------------------|-----------|---------------|
| Marist       | Oct. 27–30        | 1,214     | +3 points     |
| Emerson      | Oct. 30–Nov. 2    | 790       | +2 points     |
| Research Co. | Nov. 2–3          | 450       | +2 points     |
| NYT/Siena    | Oct. 29–Nov. 2    | 998       | Tied          |
| AtlasIntel   | Nov. 3–4          | 1,113     | –1.5 points   |

[Marist](#) • [Emerson](#) • [AtlasIntel](#)

We'll use Research Co.'s sample size for our first simulation.

# Were the polls wrong?

---

The poll put Harris ahead by two points, reported a margin of error of  $\pm 4.6$  points for each candidate's share, and Trump won by 1.41 points. Does that outcome surprise you? How could we work out how often a poll would put the losing candidate ahead?

# Today

---

1. How often can a poll put the losing candidate ahead?
2. Can differences in who actually turns out to vote change the leader?

# The 2024 Michigan poll

---

Research Co. surveyed **450 likely voters**.

```
poll_example |>
  select(harris_pct, trump_pct, harris_lead, reported_moe)
```

The margin of error describes uncertainty in each candidate's share.

[Original polling tables, page 4](#)

# Does the reported margin of error make sense?

For an estimated proportion  $\hat{p}$ , the approximate 95% margin of error is  $1.96\sqrt{\hat{p}(1 - \hat{p})/n}$ .

Research Co. surveyed 450 people, and 49% supported Harris. Using the binomial model, what margin of error would we expect? Does it match the reported  $\pm 4.6$  percentage points? Give your calculation and answer.

# Check the calculation

---

```
100 * 1.96 * sqrt(.49 * (1 - .49) / 450)
```

The result is approximately **4.62 percentage points**, matching the reported **±4.6** after rounding.

Our simulation uses the same sample size and a Harris proportion close to 50%, giving approximately the same margin of error.

# Q1: How often can a poll put the losing candidate ahead?

---

How easy is it for a poll to put the losing candidate ahead? We'll study how to answer this question using both theoretical calculations and simulations.

# Start with the election result

---

`election_result` gives the certified Harris and Trump vote counts.

```
election_result
```

For our first model, keep these two candidates and treat their final proportions as fixed.

# One response in our model

---

What is the probability that a randomly selected Harris-or-Trump voter supports Harris?

```
p_harris <- 2736533 / (2736533 + 2816636)  
p_harris
```

Each vote supports Harris with probability **about 49.28%**.

# The binomial distribution

---

Let  $H$  be the number of Harris responses in a poll of 450 people.

$$H \sim \text{Binomial}(450, p_{\text{Harris}}).$$

- There are 450 independent responses.
- Each has the same probability of supporting Harris.
- The binomial distribution gives the probability of every possible count.

# Run an imaginary poll

---

`rbinom()` generates a random count from a binomial model.

```
harris_votes <- rbinom(n = 1, size = 450, prob = p_harris)
harris_votes
```

- `n = 1` asks for one imaginary poll.
- `size = 450` gives the number of people polled.
- `prob = p_harris` gives each person's chance of supporting Harris.

The result is how many of those 450 people support Harris.

# From a count to a lead

---

The remaining responses support Trump. Convert the counts to a lead:

```
harris_votes <- rbinom(1, size = 450, prob = p_harris)
trump_votes <- 450 - harris_votes
lead <- 100 * (harris_votes - trump_votes) / 450
lead
```

Run it several times. Does your imaginary poll always put Trump ahead?

# Many imaginary polls

---

Changing the `n = ...` asks R to run more polls.

```
harris_counts <- rbinom(10, size = 450, prob = p_harris)
harris_counts
100 * (2 * harris_counts / 450 - 1)
```

About what fraction of the time is Harris leading in these imaginary polls? Run the code a few times and use the results to estimate the fraction.

# How often is Harris ahead?

---

Generate enough polls to estimate that frequency.

```
harris_counts <- rbinom(10000, size = 450, prob = p_harris)
harris_leads <- 100 * (2 * harris_counts / 450 - 1)
100 * mean(harris_leads > 0)
```

The answer should be around **36%**. Run it again: why does the estimate change?

# Why simulate?

---

- For simple models, we can mathematically calculate margins of error and the probability that a poll puts the winning candidate ahead.
- What if we add age-dependent turnout, or inaccurate polling responses as in PS6? The resulting distribution becomes harder to work out.
- For more complex models, we typically rely on **simulation**: generate many possible outcomes and use them to estimate the quantities we need.

We'll build the poll one response at a time so we can add these steps.

# Draw one poll

---

`sample()` randomly chooses responses. The vote counts provide their relative probabilities through `prob`.

```
votes <- sample(election_result$candidate, size = 450,  
               replace = TRUE, prob = election_result$votes)  
head(votes)
```

`replace = TRUE` keeps the probabilities the same on every independent draw.

# Calculate Harris's lead

---

`votes == "Harris"` checks each response. Its mean is Harris's share.

```
votes <- sample(election_result$candidate, size = 450,  
               replace = TRUE, prob = election_result$votes)  
harris_share <- mean(votes == "Harris")  
harris_lead <- 100 * (2 * harris_share - 1)  
harris_lead
```

The next sample will generally give us a different lead.

# Run your first poll

---

```
election_result
```

```
# Draw one poll and calculate Harris's lead.
```

- Draw 450 independent responses, using the election vote counts as sampling weights.
- Calculate Harris's share of the responses.
- Display Harris's lead in percentage points; say which candidate your poll

# One completed poll

---

## ▼ [Draw 450 independent responses]

```
votes <- sample(election_result$candidate, size = 450,  
               replace = TRUE, prob = election_result$votes)
```

## ▼ [Calculate Harris's share]

```
harris_share <- mean(votes == "Harris")
```

## ▼ [Calculate Harris's lead]

```
harris_lead <- 100 * (2 * harris_share - 1)  
harris_lead
```

# Try it again

---

Run the completed poll several times.

- Does it always put Trump ahead?
- How could we estimate the fraction of polls that put Harris ahead?
- What would we need to save from each run?

# Save the results

---

We want a place to record each polling lead.

`numeric(3)` creates a vector with three entries, initially zero.

```
leads <- numeric(3)
leads
```

We'll replace each entry with the result of one simulated poll.

# Record the first lead

---

Square brackets select a position in a vector. `leads[1]` is its first entry.

```
leads <- numeric(3)
votes <- sample(election_result$candidate, size = 450,
               replace = TRUE, prob = election_result$votes)
leads[1] <- 100 * (2 * mean(votes == "Harris") - 1)
leads
```

What would we change to record the second poll?

# Repeat with a for loop

---

`for (i in 1:3)` runs the code inside the braces three times. The value of `i` is 1, then 2, then 3.

```
leads <- numeric(3)
for (i in 1:3) {
  votes <- sample(election_result$candidate, size = 450,
                  replace = TRUE, prob = election_result$votes)
  leads[i] <- 100 * (2 * mean(votes == "Harris") - 1)
}
leads
```

# Trace the loop

---

- When  $i = 1$ , we draw a poll and store its lead in `leads[1]`.
- When  $i = 2$ , we draw a new poll and store its lead in `leads[2]`.
- When  $i = 3$ , we draw another poll and store its lead in `leads[3]`.

The population and sample size stay fixed. The sampled responses change.

# Generate 10,000 polls

---

```
leads <- numeric(10000)
for (i in 1:10000) {
  # Draw a poll of 450 people.
  # Save Harris's lead in leads[i].
}
```

- Use a for loop to draw 10,000 independent polls of 450 people.
- Use the election vote counts as the sampling weights in every poll.
- Store each poll's Harris lead in the corresponding entry of leads.

# Our repeated-poll code

## ▼ [Repeat the poll and save 10,000 leads]

```
leads <- numeric(10000)
for (i in 1:10000) {
  votes <- sample(election_result$candidate, size = 450,
                  replace = TRUE, prob = election_result$votes)
  leads[i] <- 100 * (2 * mean(votes == "Harris") - 1)
}
```

## ▼ [Put the leads into a data table]

```
simulated_polls <- tibble(harris_lead = leads)
```

`tibble()` puts our vector into a column named `harris_lead`. Each row is one simulated poll.

# A table of repeated polls

---

```
simulated_polls |> head(5)
```

What should its histogram look like, based on the distribution we calculated?

# Plot the polling leads

---

```
simulated_polls  
  
# Plot Harris's lead across the simulated polls.
```

Make a histogram of Harris's lead across the 10,000 simulated polls. Label the horizontal axis in percentage points. Submit the plot and describe how it relates to the imaginary polls generated with `rbinom()`.

# What the simulation produces

---

```
simulated_polls |>  
  ggplot(aes(harris_lead)) +  
  geom_histogram(fill = "#346888", color = "white") +
```

## ▼ [Mark zero and label the axes]

```
  geom_vline(xintercept = 0, color = "#B85D24", linewidth = 1) +  
  coord_cartesian(xlim = c(-20, 20)) +  
  labs(x = "Harris's polling lead (percentage points)",  
       y = "Number of polls")
```

# How often did our poll put Harris ahead?

---

```
simulated_polls |> head(3)  
  
# Find the percentage of polls putting Harris ahead.
```

- Identify polls with a positive Harris lead.
- Calculate the percentage of simulated polls putting Harris ahead.
- Explain whether the result agrees with the earlier `rbinom()` simulation.

# Two ways to simulate the same poll

---

```
simulated_polls |>  
  summarize(percent = 100 * mean(harris_lead > 0))
```

Both approaches should put Harris ahead in about **36%** of polls. More repetitions make the estimate more stable.

# Would a larger poll settle it?

---

What do you think will happen if we increase the number of people in each poll? Predict how the distribution of polling leads will change and explain your reasoning.

# Try a larger sample

```
n <- 450
```

▼ [Repeat the poll at sample size n]

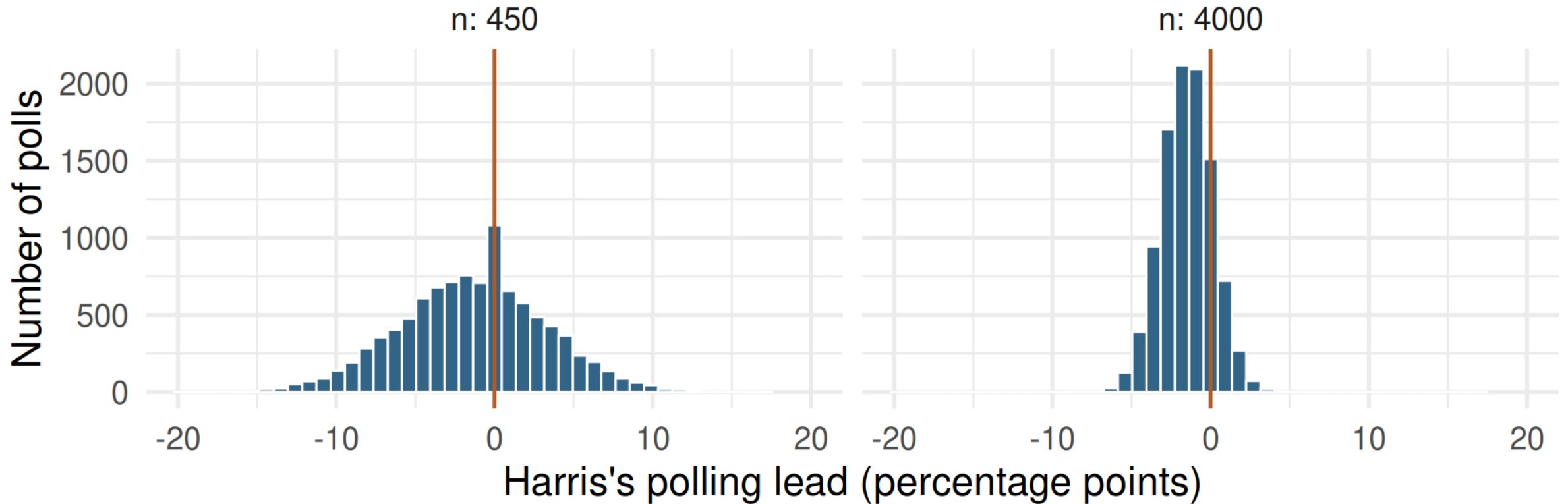
```
leads <- numeric(10000)
for (i in 1:10000) {
  votes <- sample(election_result$candidate, size =
                  replace = TRUE, prob = election_re
  leads[i] <- 100 * (2 * mean(votes == "Harris") - 1
}
large_polls <- tibble(harris_lead = leads)
```

```
# Calculate how often Harris is ahead.
```

- Run the simulation at  $n = 450$ ,  $n = 790$ , and  $n = 1214$ .
- Record the percentage of polls putting Harris ahead at each sample size.
- Explain how much the larger samples reduce that frequency under our model.

# More people in each poll

---



The distribution narrows around the same underlying lead.

# Were the polls wrong?

---

- The Research Co. point estimate put the eventual loser ahead.
- Under our simple model, that happens in **about 36%** of polls of 450 people.
- A small polling lead is compatible with the other candidate winning.

What changes if the people answering the poll differ from the people who vote?

# Q2: Can differences in turnout change who's ahead?

---

- A poll asks people whom they support.
- The election depends on which of those people vote.
- If turnout varies with age, how could that change the lead?

We'll use Census data to give our model Michigan's age distribution.

# Michigan's voting-age citizens

---

michigan\_ages contains Census estimates of Michigan residents who are U.S. citizens aged 18 or older, grouped by age.

```
michigan_ages
```

About **one in five** are aged 18–29.

2024 American Community Survey, table B29001

# Sample ages in Michigan

---

The age groups have different sizes. We want those sizes reflected in our random sample.

`prob` tells `sample()` how much weight to give each possible age group. We can use the Census counts as weights.

```
ages <- sample(michigan_ages$age_group, size = 1000,  
              replace = TRUE, prob = michigan_ages$citizens)  
table(ages)
```

# Specify support and turnout

---

Keep the Census age distribution. For this experiment, assume:

- Harris support is 65% at ages 18–29 and 47.5% at older ages.
- Turnout is 40% at ages 18–29 and 80% at older ages.

These support and turnout rates are classroom assumptions.

Within each age group, simulate preference and participation independently.

# Predict the effect of turnout

---

Younger voters favor Harris in this model, but are less likely to vote. How should the lead among actual voters differ from the lead among all sampled citizens? Describe the steps needed to simulate that change.

# Simulating a coin toss

---

A coin toss has two outcomes. Code heads as 1 and tails as 0.

```
rbinom(n = 1, size = 1, prob = .5)
```

- `n = 1` simulates one toss.
- `size = 1` gives one trial per toss.
- `prob = .5` makes heads and tails equally likely.

Run it several times. What would changing `prob` to `.4` do?

# Generate a random decision

---

We need to decide whether each person votes.

Treat voting as a coin toss with a 40% chance of returning 1. Here 1 means the person votes; 0 means they stay home.

```
rbinom(n = 1, size = 1, prob = .4)  
rbinom(n = 10, size = 1, prob = .4)
```

# Build one simulated electorate

---

Sample 1,000 ages, then attach our support and turnout assumptions.

```
people <- tibble(  
  age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,  
                    prob = michigan_ages$citizens)  
) |>  
  mutate(  
    harris_prob = ifelse(age_group == "18-29", .65, .475),  
    turnout_prob = ifelse(age_group == "18-29", .4, .8)  
  )  
head(people)
```

# Simulate preferences and participation

Use separate coin tosses for preference and turnout. Giving `rbinom()` a column of probabilities lets each person have their own chance of a 1.

▼ [Sample ages and attach the model probabilities]

```
people <- tibble(
  age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,
                    prob = michigan_ages$citizens)
) |>
  mutate(harris_prob = ifelse(age_group == "18-29", .65, .475),
         turnout_prob = ifelse(age_group == "18-29", .4, .8))

people <- people |>
  mutate(harris = rbinom(n = 1000, size = 1, prob = harris_prob),
         voted = rbinom(n = 1000, size = 1, prob = turnout_prob))
head(people)
```

`harris` records preference; `voted` records participation.

# Whose lead are we calculating?

## ▼ [Generate the people, preferences, and participation]

```
people <- tibble(  
  age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,  
                    prob = michigan_ages$citizens)  
) |>  
  mutate(harris_prob = ifelse(age_group == "18-29", .65, .475),  
         turnout_prob = ifelse(age_group == "18-29", .4, .8),  
         harris = rbinom(n = 1000, size = 1, prob = harris_prob),  
         voted = rbinom(n = 1000, size = 1, prob = turnout_prob))  
  
people |> summarize(lead = 100 * (2 * mean(harris) - 1))  
people |> filter(voted == 1) |>  
  summarize(lead = 100 * (2 * mean(harris) - 1))
```

The second calculation includes only people who voted.

# Reuse the simulation

---

We want to rerun this process at different younger-voter turnout rates. A **function** names the procedure and gives us an input we can change.

`function(young_turnout)` defines that input. The last expression supplies the result.

```
turnout_poll <- function(young_turnout) {  
  people <- tibble(  
    age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,  
                      prob = michigan_ages$citizens)  
  ) |>  
  mutate(harris_prob = ifelse(age_group == "18-29", .65, .475),  
         turnout_prob = ifelse(age_group == "18-29", young_turnout, .8),  
         harris = rbinom(n = 1000, size = 1, prob = harris_prob),  
         voted = rbinom(n = 1000, size = 1, prob = turnout_prob))  
  voters <- people |> filter(voted == 1)  
  100 * (2 * mean(voters$harris) - 1)  
}  
turnout_poll(.4)
```

# Repeat the turnout experiment

The loop now calls our function once per repetition.

## ▼ [Define one turnout simulation]

```
turnout_poll <- function(young_turnout) {  
  people <- tibble(  
    age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,  
                      prob = michigan_ages$citizens)  
  ) |>  
  mutate(harris_prob = ifelse(age_group == "18-29", .65, .475),  
         turnout_prob = ifelse(age_group == "18-29", young_turnout, .8),  
         harris = rbinom(n = 1000, size = 1, prob = harris_prob),  
         voted = rbinom(n = 1000, size = 1, prob = turnout_prob))  
  voters <- people |> filter(voted == 1)  
  100 * (2 * mean(voters$harris) - 1)  
}
```

```
leads <- numeric(2000)  
for (i in 1:2000) {  
  leads[i] <- turnout_poll(.4)  
}  
turnout_results <- tibble(harris_lead = leads)  
turnout_results |>  
  summarize(mean_lead = mean(harris_lead),
```

# Inspect the resulting distribution

## ▼ [Generate 2,000 turnout simulations]

```
turnout_poll <- function(young_turnout) {  
  people <- tibble(  
    age_group = sample(michigan_ages$age_group, 1000, replace = TRUE,  
                      prob = michigan_ages$citizens)  
  ) |>  
  mutate(harris_prob = ifelse(age_group == "18-29", .65, .475),  
         turnout_prob = ifelse(age_group == "18-29", young_turnout, .8),  
         harris = rbinom(n = 1000, size = 1, prob = harris_prob),  
         voted = rbinom(n = 1000, size = 1, prob = turnout_prob))  
  voters <- people |> filter(voted == 1)  
  100 * (2 * mean(voters$harris) - 1)  
}  
leads <- numeric(2000)  
for (i in 1:2000) {  
  leads[i] <- turnout_poll(.4)  
}  
  
ggplot(turnout_results, aes(harris_lead)) +  
  geom_histogram(binwidth = 1, fill = "#346888", color = "white") +  
  geom_vline(xintercept = 0) +  
  labs(x = "Harris lead among voters (percentage points)", y = "Simulations")
```

# Change younger-voter turnout

---

Open the simulation fold on the preceding slide and change .4 to .8 in the function call.

- Predict what will happen before running it.
- Repeat the simulation and inspect the histogram.
- Calculate the mean lead and the percentage of positive leads.

Which parts of the model stayed fixed? What changed the outcome?

# What did turnout change?

---

Under these assumptions:

- Harris leads by about **1.8 points** among all citizens in the model.
- At 40% younger turnout and 80% older turnout, the average voter lead is about **-1.2 points**.
- Equal turnout rates bring the average lead back to about **+1.8 points**.

We generated the distribution by repeatedly executing the sampling, preference, and turnout steps.

# Explain the result

---

Explain how Harris can lead among the people sampled but trail among those who vote in our model. Use a result from the simulation. What would you need to learn about Michigan voters to choose more realistic turnout probabilities?

# The analysis we built

---

1. Generate imaginary polls using a binomial model.
2. Build the same simulation one response at a time inside a loop.
3. Add age-dependent preferences and random participation.
4. Put that process into a function.
5. Change a turnout assumption and investigate the resulting distribution.