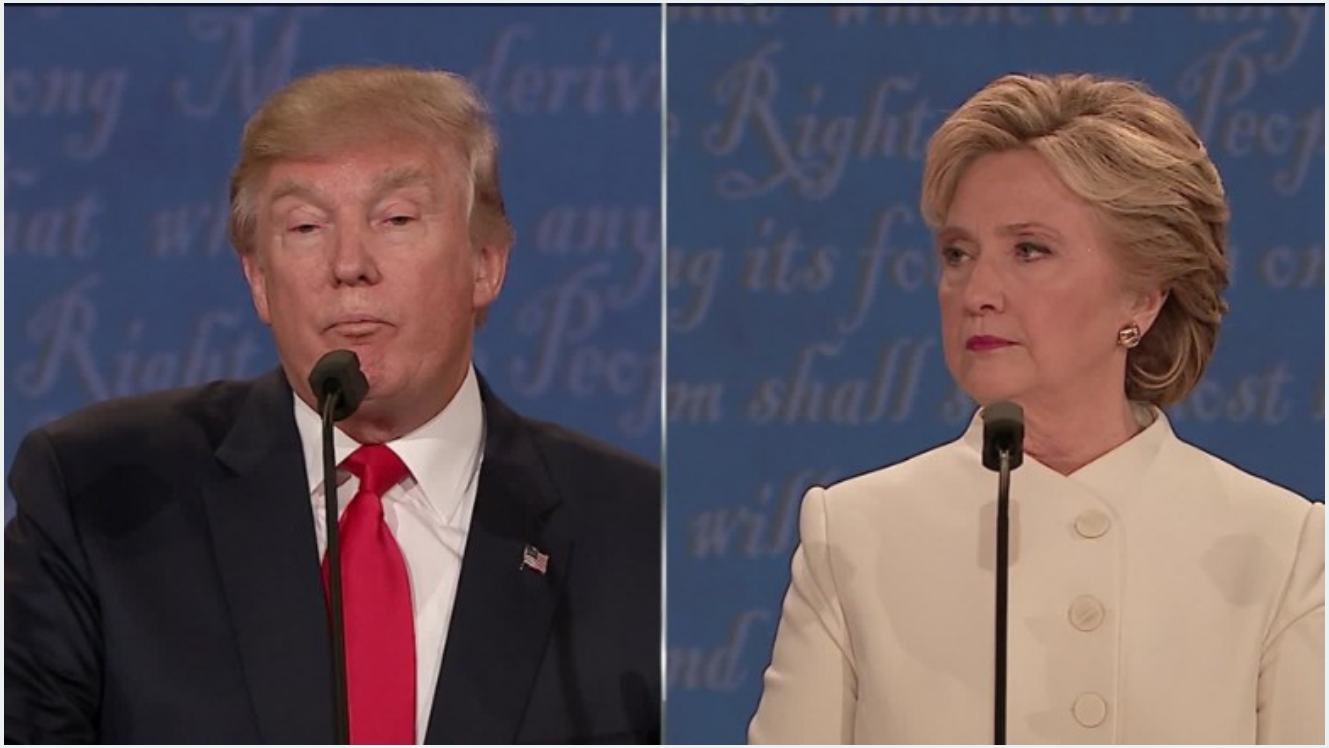
# Gov 51: Loops & Predicting Elections

Matthew Blackwell

Harvard University

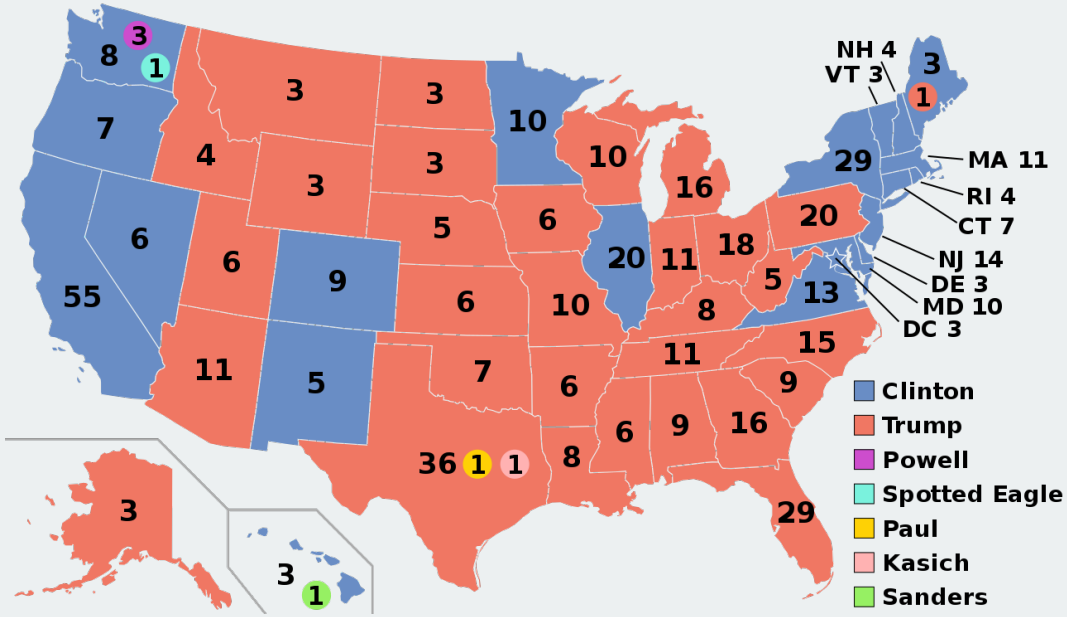
# 2016 US Presidential Election



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

# Predicting US Presidential Elections

- **Electoral college system**
  - Must win an absolute majority of 538 electoral votes
  - 538 = 435 (House of Representatives) + 100 (Senators) + 3 (DC)
  - Must win at least 270 votes
  - nobody wins an absolute majority ⇝ 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...

# Multiplication

```
values <- c(2, 4, 6)
```

- Let's create a new variable that multiplies each value in a vector by 2.
  - Easy in R: `values * 2`
  - Pretend you didn't know this approach

# Manually changing values

```
values <- c(2, 4, 6)
```

# Manually changing values

```
values <- c(2, 4, 6)

## number of values
n <- length(values)
```

# Manually changing values

```
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, times = n)
```

# Manually changing values

```
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, times = n)

## multiply each value by 2
results[1] <- values[1] * 2
```

# Manually changing values

```
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, times = n)

## multiply each value by 2
results[1] <- values[1] * 2
results[2] <- values[2] * 2
```

# Manually changing values

```
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, times = n)

## multiply each value by 2
results[1] <- values[1] * 2
results[2] <- values[2] * 2
results[3] <- values[3] * 2
```

# Manually changing values

```r
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, times = n)

## multiply each value by 2
results[1] <- values[1] * 2
results[2] <- values[2] * 2
results[3] <- values[3] * 2

## print results
results
```

```
## [1]  4  8 12
```

# Loops in R

- What if you had more values? Not efficient!
    - **for loop**: a way to iteratively execute the same code multiple times.
- Basic structure:

```
for (i in X) {
  expression1
  expression2
  ...
  expression3
}
```

- Elements of a loop:
    1. `i`: counter (can use any name)
    2. `X`: vector containing a set of ordered values the counter takes.
    3. `expression`: a set of expressions that will be repeatedly evaluated.
    4. `{ }`: curly braces to define beginning and end of the loop.
- Indentation is important for readability of the code.

# Loop example

```
values <- c(2, 4, 6)

## number of values
n <- length(values)

## create container to hold results
results <- rep(NA, n)

## begin loop
for (i in 1:n) {
  results[i] <- values[i] * 2

  ## use cat() to display output
  cat(values[i], "times 2 is equal to ", results[i], "\n")
}
```

```
## 2 times 2 is equal to  4
## 4 times 2 is equal to  8
## 6 times 2 is equal to  12
```

# 2016 polling prediction

- Election data: `pres16.csv`

| Name | Description |
|---|---|
| `state` | abbreviated name of state |
| `state.name` | unabbreviated name of state |
| `clinton` | Clinton's vote share (percentage) |
| `trump` | Trump's vote share (percentage) |
| `ev` | number of electoral college votes for the state |

- Polling data `polls16.csv`

| Name | Description |
|---|---|
| `state` | abbreviated name of state in which poll was conducted |
| `middate` | middate of the period when poll was conducted |
| `daysleft` | number of days between middate and election day |
| `pollster` | name of organization conducting poll |
| `clinton` | predicted support for Clinton (percentage) |
| `trump` | predicted support for Trump (percentage) |

# Some preprocessing

```
# election results by state
pres16 <- read.csv("data/pres16.csv")

# polling data
polls16 <- read.csv("data/polls16.csv")

# calculate Trump's margin of victory
polls16$margin <- polls16$trump - polls16$clinton
pres16$margin <- pres16$trump - pres16$clinton
```

# What does the data look like?

```
head(polls16)
```

```
##   state  middate daysleft                  pollster
## 1    AK  8/11/16       89  Lake Research Partners
## 2    AK  8/20/16       80              SurveyMonkey
## 3    AK 10/20/16       19                   YouGov
## 4    AK 10/26/16       13 Google Consumer Surveys
## 5    AK  9/30/16       39 Google Consumer Surveys
## 6    AK 10/12/16       27 Google Consumer Surveys
##   clinton trump margin
## 1    30.0  38.0   8.00
## 2    31.0  38.0   7.00
## 3    37.4  37.7   0.30
## 4    38.0  39.0   1.00
## 5    47.5  36.7 -10.76
## 6    34.6  30.0  -4.62
```

# Poll prediction for each state

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

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)
```

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

for (i in 1:51) {
```

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

for (i in 1:51) {
  state_data <- subset(polls16, subset = (state == state_names[i]))
```

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

for (i in 1:51) {
  state_data <- subset(polls16, subset = (state == state_names[i]))

  latest <- state_data$daysleft == min(state_data$daysleft)
```

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

for (i in 1:51) {
  state_data <- subset(polls16, subset = (state == state_names[i]))

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

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

# Poll prediction for each state

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

# get list of unique state names to iterate over
state_names <- unique(polls16$state)

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

for (i in 1:51) {
  state_data <- subset(polls16, subset = (state == state_names[i]))

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

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

head(poll_pred)
```

```
##     AK     AL     AR     AZ     CA     CO
##  14.73  29.72  20.02   2.50 -23.00  -7.05
```

# Comparing polls to outcome

```
plot(poll_pred, pres16$margin, type = "n", main = "",
     xlim = c(-45, 50), ylim = c(-45, 50),
     xlab = "Poll Results",
     ylab = "Actual Election Results")

abline(a = 0, b = 1, lty = "dashed") ## 45-degree line
abline(v = 0, col = "grey50")
abline(h = 0, col = "grey50")

text(poll_pred, pres16$margin, pres16$state,
     col = "dodgerblue")
```

# Comparing polls to outcome