Loops, Prediction, and Regression

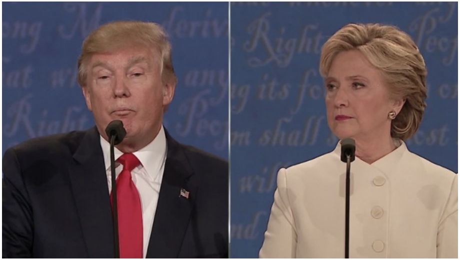
PSC7475: Week 5

Prof. Weldzius

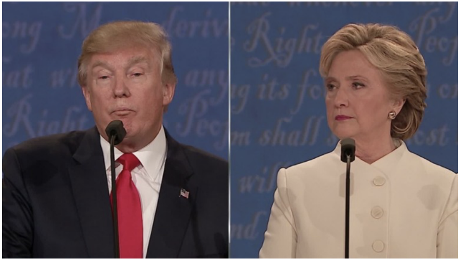
Villanova University

Slides Updated: 2025-02-19

2016 US Presidential Election

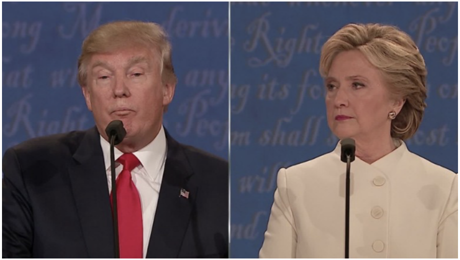


2016 US Presidential Election



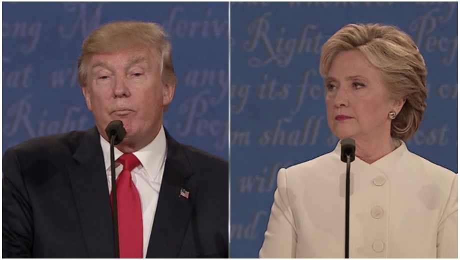
- 2016 election popular vote:

2016 US Presidential Election



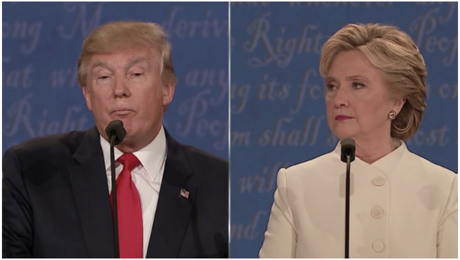
- 2016 election popular vote:
  - Clinton: 65,853,516 (48.2%)
  - Trump: 62,984,825 (46.1%)

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

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, PA)
  - 0.056% of the electorate (~136million)

Predicting US Presidential Elections

Predicting US Presidential Elections

- **Electoral college system**

Predicting US Presidential Elections

- **Electoral college system**
    - Must win an absolute majority of 538 electoral votes

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)

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

Predicting US Presidential Elections

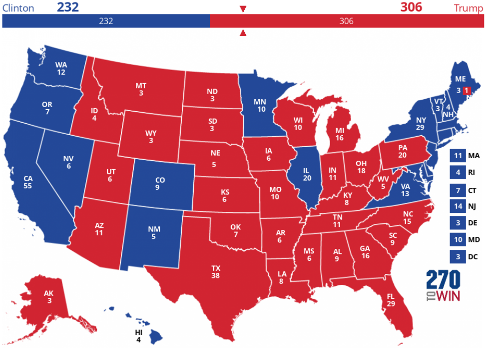
- **Electoral college system**
  - Must win an absolute majority of 538 electoral votes
  - $538 = 435$ (House of Representatives) $+ 100$ (Senators) $+ 3$ (DC)
  - Must win at least 270 votes
  - nobody wins an absolute majority $\rightsquigarrow$ House vote

Predicting US Presidential Elections

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

Predicting US Presidential Elections

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



Prediction strategy

Prediction strategy

- Predict state-level support for each candidate using polls

Prediction strategy

- Predict state-level support for each candidate using polls
- Allocate electoral college votes of that state to its predicted winner

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

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:

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

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

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

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

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

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

Multiplication

- Let's create a new variable that multiples each value in a vector by 2:

Multiplication

- Let's create a new variable that multiples each value in a vector by 2:
    - Easy in R: `values * 2`

Multiplication

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

Multiplication

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

Multiplication

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

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

Multiplication

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

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

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

Multiplication

```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
```

Multiplication

```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

results
```

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

Loops in R

Loops in R

- What if you had more values? Not efficient!

Loops in R

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

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:

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
}
```

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:

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:
  - `i`: counter (can use any name)

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:

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

- Elements of a loop:
  - `i`: counter (can use any name)
  - `X`: vector containing a set of ordered values the counter takes

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:
  - `i`: counter (can use any name)
  - `X`: vector containing a set of ordered values the counter takes
  - `expression`: a set of expressions that will be repeatedly evaluated.

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:
  - `i`: counter (can use any name)
  - `X`: vector containing a set of ordered values the counter takes
  - `expression`: a set of expressions that will be repeatedly evaluated.
  - `{ }`: curly braces to define beginning and end of the loop.

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:
  - `i`: counter (can use any name)
  - `X`: vector containing a set of ordered values the counter takes
  - `expression`: a set of expressions that will be repeatedly evaluated.
  - `{ }`: 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)
n <- length(values)
results <- rep(NA, times = n)

## begin loop
for (i in 1:n) {
  results[i] <- values[i] * 2
  print(str_c(values[i], " times 2 is equal to ", results[i]))
}
```

```
## [1] "2 times 2 is equal to 4"
## [1] "4 times 2 is equal to 8"
## [1] "6 times 2 is equal to 12"
```

2016 polling prediction

- Election data: `pres.csv`

Name	Description
`state_abb`	abbreviated name of state
`clinton`	Clinton's vote share (percentage)
`trump`	Trump's vote share (percentage)

- Polling data `polls16.csv`

Name	Description
`state`	abbreviated name of state in which polls was conducted
`middate`	middate of the period when polls 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

```
## download; don't forget to setwd()
pres16 <- read_csv("../data/pres2016.csv")
polls16 <- read_csv("../data/polls2016.csv")

## calculate Trump's margin of victory
polls16 <- polls16 %>%
  mutate(margin = Trump - Clinton)
pres16 <- pres16 %>%
  mutate(margin = Trump - Clinton)
```

What does the data look like?

```r
head(polls16)
```

```
## # A tibble: 6 x 8
##      id state Clinton Trump days_to_election electoral_votes
##   <dbl> <chr>   <dbl> <dbl>             <dbl>           <dbl>
## 1 26255 TX         38    41                24              38
## 2 26253 WI         48    44                23              10
## 3 26252 VA         54    41                23              13
## 4 26251 NV         47    40                19               6
## 5 26250 TX         46    48                23              38
## 6 26249 NH         50    43                23               4
## # i 2 more variables: population <chr>, margin <dbl>
```

Poll prediction for each state

```r
## place holder
poll.pred <- rep(NA, 51)

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

## add labels to place holder
names(poll.pred) <- state_names
```

Poll prediction for each state

```r
for (i in seq_along(state_names)) {
```

Poll prediction for each state

```
for (i in seq_along(state_names)) {
  ## subset the ith state
  state.data <- polls16 %>%
    filter(state == state_names[i])
```

Poll prediction for each state

```
for (i in seq_along(state_names)) {
  ## subset the ith state
  state.data <- polls16 %>%
    filter(state == state_names[i])

  ## pull out the closest date (minimum days to election)
  min_days <- min(state.data$days_to_election)
```

Poll prediction for each state

```r
for (i in seq_along(state_names)) {
  ## subset the ith state
  state.data <- polls16 %>%
    filter(state == state_names[i])

  ## pull out the closest date (minimum days to election)
  min_days <- min(state.data$days_to_election)

  ## subset only the latest polls within the state
  state.data <- state.data %>%
    filter(days_to_election == min_days)
```

Poll prediction for each state

```
for (i in seq_along(state_names)) {
  ## subset the ith state
  state.data <- polls16 %>%
    filter(state == state_names[i])

  ## pull out the closest date (minimum days to election)
  min_days <- min(state.data$days_to_election)

  ## subset only the latest polls within the state
  state.data <- state.data %>%
    filter(days_to_election == min_days)

  ## compute the mean of the latest polls and store it
  poll.pred[i] <- mean(state.data$margin)
}
head(poll.pred)
```

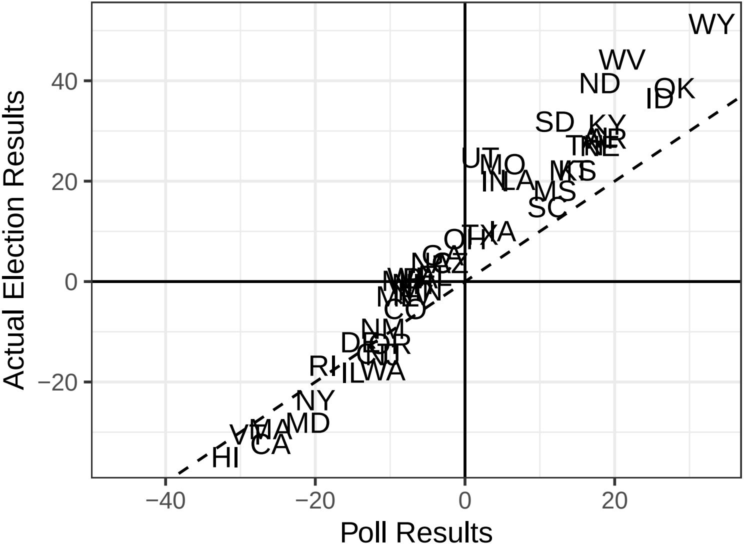
```
##  TX  WI  VA  NV  NH  PA
##   2  -8 -15  -7  -7  -6
```

Poll prediction for each state (my way)

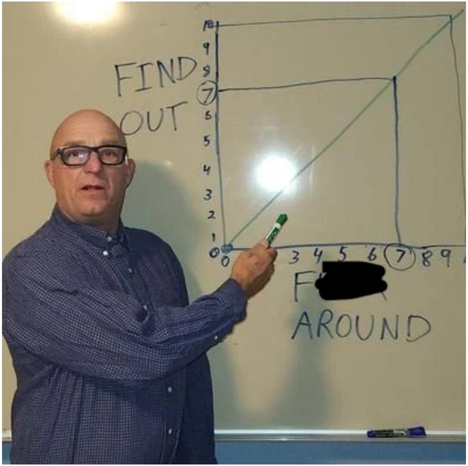
```
poll.list <- list()
state_names <- unique(polls16$state)

for (i in seq_along(state_names)) {
  state.data <- polls16 %>%
    filter(state == state_names[i]) %>%
    filter(days_to_election == min(days_to_election)) %>%
    mutate(margin_poll = mean(margin)) %>%
    select(state, margin_poll)
  poll.list[[i]] <- state.data
  print(i)
}
PollPred <- do.call(rbind,poll.list)
head(PollPred)
```

Comparing polls to outcomes



Let’s talk about regression



Can betting markets help us predict elections?

Can betting markets help us predict elections?

- Data from an online betting company Intrade

Can betting markets help us predict elections?

- Data from an online betting company Intrade
- People trade contracts such as “Obama to win the electoral votes in Florida' '

Can betting markets help us predict elections?

- Data from an online betting company Intrade
- People trade contracts such as "Obama to win the electoral votes in Florida' '
- Market prices of each contract fluctuate based on its sales

Can betting markets help us predict elections?

- Data from an online betting company Intrade
- People trade contracts such as "Obama to win the electoral votes in Florida' '
- Market prices of each contract fluctuate based on its sales
- Why might we expect betting markets like Intrade to accurately predict outcomes of elections?

Linear Regression: Prediction using bivariate relationships

Linear Regression: Prediction using bivariate relationships

- Goal: what's our best guess about Y_i if we know what X_i is?

Linear Regression: Prediction using bivariate relationships

- Goal: what's our best guess about Y_i if we know what X_i is?
  - What's our best guess about election margins if we know the market's margins?

Linear Regression: Prediction using bivariate relationships

- Goal: what's our best guess about Y_i if we know what X_i is?
  - What's our best guess about election margins if we know the market's margins?
- Terminology:
  - **Dependent/outcome variable**: what we want to predict (election margin).

Linear Regression: Prediction using bivariate relationships

- Goal: what's our best guess about Y_i if we know what X_i is?
  - What's our best guess about election margins if we know the market's margins?
- Terminology:
  - **Dependent/outcome variable**: what we want to predict (election margin).
  - **Independent/explanatory variable**: what we're using to predict (market margin).

We'll use two datasets: `intrade08.csv` & `pres08.csv`

Name	Description
`day`	Date of the session
`statename`	Full name of each state (including DC in 2008)
`state`	Abbreviation of each state (including DC in 2008)
`PriceD`	Closing price (predicted vote share) of Democratic Nominee's market
`PriceR`	Closing price (predicted vote share) of Republican Nominee's market
`VolumeD`	Total session trades of Democratic Party Nominee's market
`VolumeR`	Total session trades of Republican Party Nominee's market

- `intrade08.csv`: Each row represents daily trading information about the contracts for either the Democratic or Republican Party nominee's victory in a particular state.

Presidential voting data from 2008

Name	Description
`state.name`	Full name of state (only in `pres2008`)
`state`	Two letter state abbreviation
`Obama`	Vote percentage for Obama
`McCain`	Vote percentage for McCain
`EV`	Number of electoral college votes for this state

Predicting Elections Using Betting Markets and Linear Models

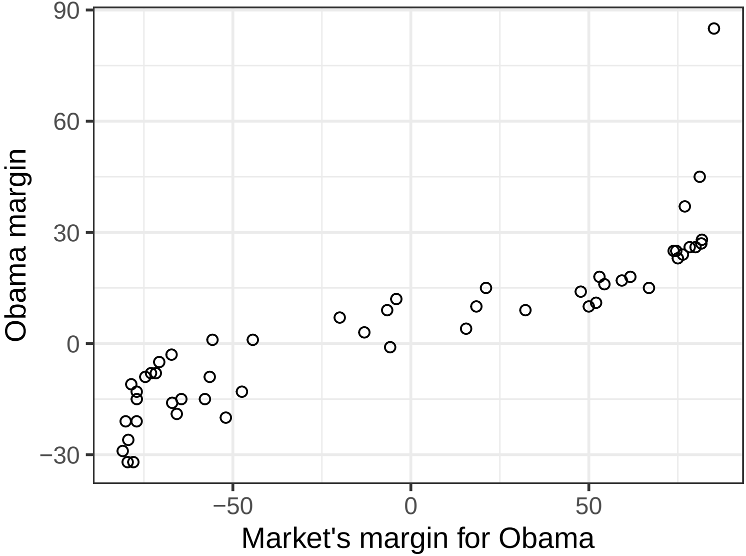
Predicting Elections Using Betting Markets and Linear Models

- Load the data

```
library(tidyverse)
intrade08 <- read.csv("../data/intrade08.csv")
pres08 <- read.csv("../data/pres08.csv")

## merge datasets and calculate margins for DV and IV
intresults08 <- inner_join(intrade08,pres08) %>%
  mutate(obama.intmarg = PriceD - PriceR,
         obama.actmarg = Obama - McCain)
```

Plot bivariate relationship



Using a line to predict

Using a line to predict

- Prediction: for any value of X , what's the best guess about Y?

Using a line to predict

- Prediction: 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.

Using a line to predict

- Prediction: 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 $f(x)$

Using a line to predict

- Prediction: 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 $f(x)$
- Simplest possible way to relate two variables: a line.

Using a line to predict

- Prediction: 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 $f(x)$
- Simplest possible way to relate two variables: a line.

$$y = mx + b$$

Using a line to predict

- Prediction: 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 $f(x)$
- 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.

Using a line to predict

- Prediction: 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 $f(x)$
- 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.

Using a line to predict

- Prediction: 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 $f(x)$
- 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

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

Linear Regression Model

- Model for the line of best fit

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **Coefficients/parameters** (α, β): true unknown intercept/slope of the line of best fit

Linear Regression Model

- Model for the line of best fit

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **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.

Linear Regression Model

- Model for the line of best fit

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **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

Linear Regression Model

- Model for the line of best fit

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **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"

Interpretting the Regression Line

$$Y_i = \alpha + \beta \times X_i + \epsilon_i$$

Interpretting the Regression Line

$$Y_i = \alpha + \beta \times X_i + \epsilon_i$$

- **Intercept** *alpha*: average value of Y when X is 0
  - Average Obama margin when market's margin is 0.

Interpretting the Regression Line

$$Y_i = \alpha + \beta \times X_i + \epsilon_i$$

- **Intercept** *alpha*: average value of Y when X is 0
  - Average Obama margin when market's margin is 0.
- **Slope** β: average change in Y when X increases by one unit
  - Average increase in Obama margin for each additional margin increase by the market.

Interpretting the Regression Line

$$Y_i = \alpha + \beta \times X_i + \epsilon_i$$

- **Intercept** *alpha*: average value of Y when X is 0
  - Average Obama margin when market's margin is 0.
- **Slope** β: average change in Y when X increases by one unit
  - Average increase in Obama margin for each additional margin increase by the market.
- But we don't know α or β. How can we estimate them? Next time...

Interpretting the Regression Line

$$Y_i = \alpha + \beta \times X_i + \epsilon_i$$

- **Intercept** *alpha*: average value of Y when X is 0
    - Average Obama margin when market's margin is 0.
- **Slope** β: average change in Y when X increases by one unit
    - Average increase in Obama margin for each additional margin increase by the market.
- But we don't know α or β. How can we estimate them? Next time...
    - Or now if we still have time!

Linear Regression Model (skip if same day)

- Model for the line of best fit

$$Y_i = \underbrace{\alpha}_{\text{intercept}} + \underbrace{\beta}_{\text{slope}} \times X_i + \underbrace{\epsilon_i}_{\text{error term}}$$

- **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

Estimate coefficients

Estimate coefficients

- Parameters: α, β
  - Unknown features of the data-generating process.
  - Chance error makes these impossible to observe directly.

Estimate 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.

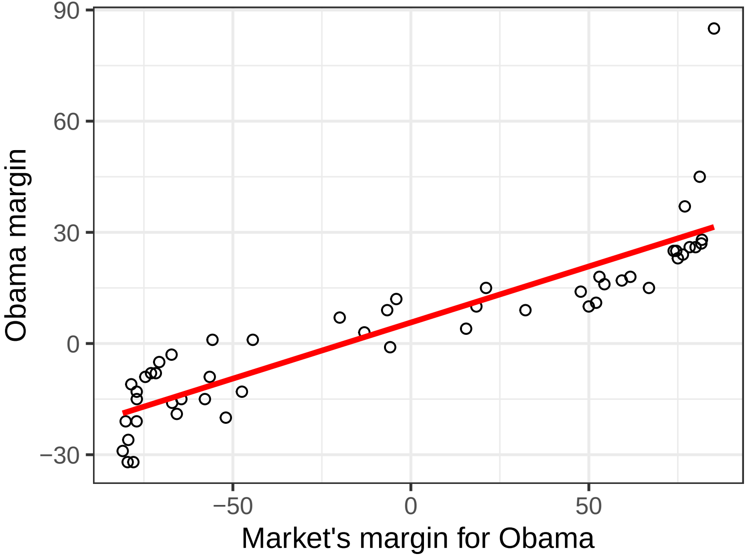
Estimate coefficients

- 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:

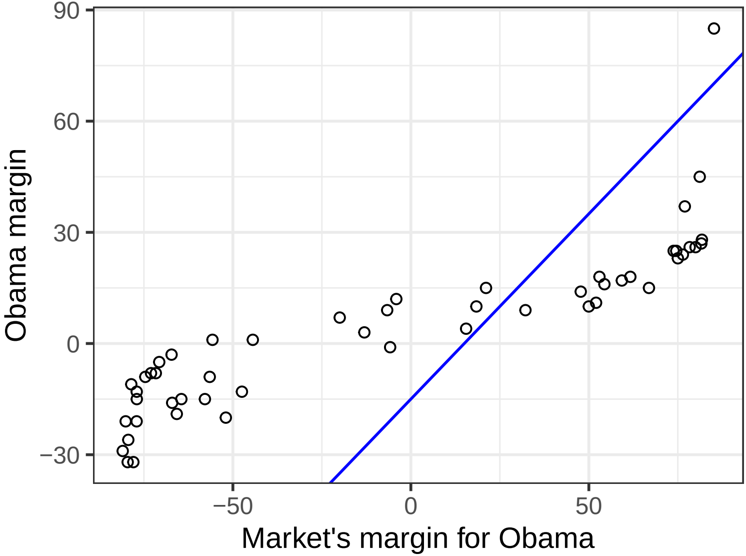
$$\hat{Y} = \hat{\alpha} + \hat{\beta} * x$$

- Average value of Y when X is x
- Represents the best guess or **predicted value** of the outcome at x.

Line of best fit



Why not this line?



Least squares

Least squares

- How do we figure out the best line to draw?

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$
  - **Residual/prediction error**: $\hat{\epsilon}_i = Y_i - \hat{Y}$

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$
  - **Residual/prediction error**: $\hat{\epsilon}_i = Y_i - \hat{Y}$
- Get these estimates by the **least squares method**

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$
  - **Residual/prediction error**: $\hat{\epsilon}_i = Y_i - \hat{Y}$
- Get these estimates by the **least squares method**
- Minimize the **sum of the squared residuals** (SSR):

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$
  - **Residual/prediction error**: $\hat{\epsilon}_i = Y_i - \hat{Y}$
- Get these estimates by the **least squares method**
- Minimize the **sum of the squared residuals** (SSR):

$$SSR = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta} X_i)^2$$

Least squares

- How do we figure out the best line to draw?
  - **Fitted/predicted value** for each observation: $\hat{Y} = \hat{\alpha} + b\hat{eta} \times X_i$
  - **Residual/prediction error**: $\hat{\epsilon}_i = Y_i - \hat{Y}$
- Get these estimates by the **least squares method**
- Minimize the **sum of the squared residuals** (SSR):

$$SSR = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (Y_i - \hat{\alpha} - \hat{\beta} X_i)^2$$

- Finds the line that minimizes the magnitude of the prediction errors!

Linear Regression in R

Linear Regression in R

- R will calculate least squares line for a data set using `lm()`

Linear Regression in R

- R will calculate least squares line for a data set using `lm()`
    - Syntax: `lm(y ~ x, data = mydata)`

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

Linear Regression in R

```
fit <- lm(obama.actmarg ~ obama.intmarg, data = intresults08)
fit
```

```
##
## Call:
## lm(formula = obama.actmarg ~ obama.intmarg, data = intresults08)
##
## Coefficients:
##   (Intercept)  obama.intmarg
##        5.5681         0.2799
```

Coefficients and fitted values

- Use coef() to extract estimated coefficients:

```
coef(fit)
```

```
##   (Intercept) obama.intmarg
##     5.5681423     0.2799326
```

- R can show you each of the fitted values as well:

```
head(fitted(fit))
```

```
##        1        2        3        4        5        6
## 5.568142 5.568142 5.568142 5.568142 5.568142 5.568142
```

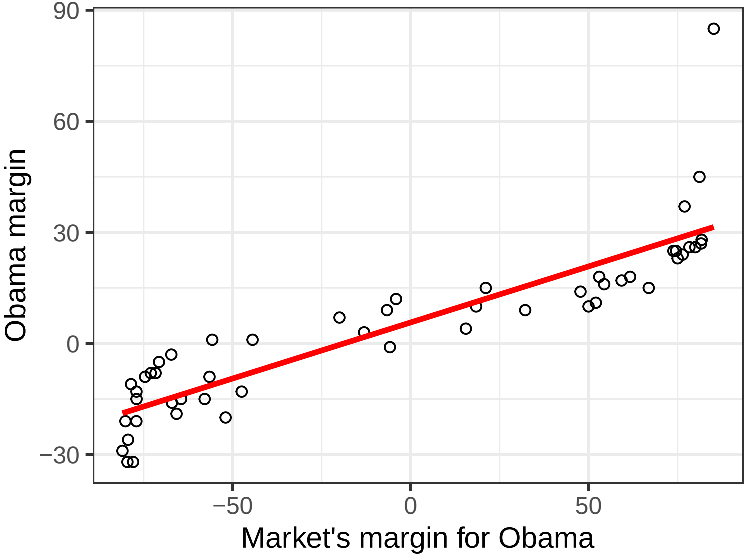
Properties of least squares

- Least squares line always goes through $(\bar{X}, \bar{Y})$
- Estimated slope is related to correlation:

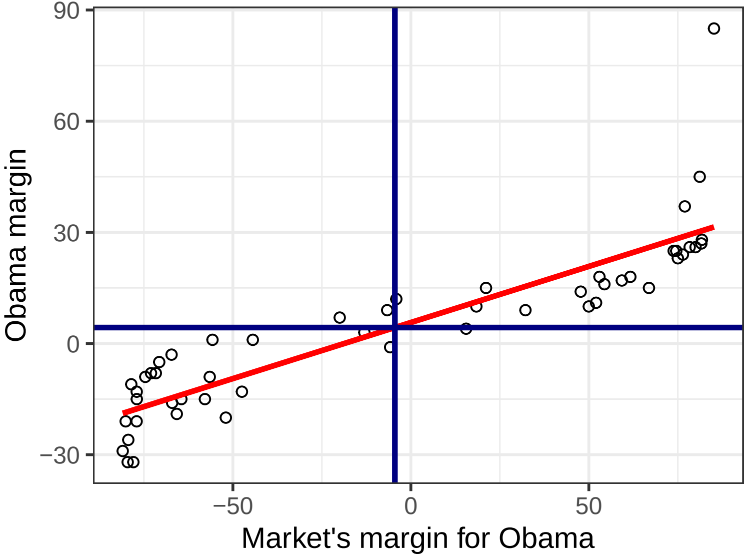
$$\hat{\beta} = (\text{correlation of } X \text{and } Y) \times \frac{\text{SD of } Y}{\text{SD of } X}$$

- Mean of residuals is always 0

Visual components of least squares



Visual components of least squares



Visual components of least squares