Harvard Data Science Professional - Data Science Inference and Modeling

# Learning Objectives

* The concepts necessary to define estimates and margins of errors of populations, parameters, estimates, and standard errors in order to make predictions about data
* How to use models to aggregate data from different sources
* The very basics of Bayesian statistics and predictive modeling

## Course Overview

### Section 1: Parameters and Estimates

You will learn how to estimate population parameters.

### Section 2: The Central Limit Theorem in Practice

You will apply the central limit theorem to assess how close a sample estimate is to the population parameter of interest.

### Section 3: Confidence Intervals and p-Values

You will learn how to calculate confidence intervals and learn about the relationship between confidence intervals and p-values.

### Section 4: Statistical Models

You will learn about statistical models in the context of election forecasting.

### Section 5: Bayesian Statistics

You will learn about Bayesian statistics through looking at examples from rare disease diagnosis and baseball.

### Section 6: Election Forecasting

You will learn about election forecasting, building on what you’ve learned in the previous sections about statistical modeling and Bayesian statistics.

### Section 7: Association Tests

You will learn how to use association and chi-squared tests to perform inference for binary, categorical, and ordinal data through an example looking at research funding rates.

## Introduction to Inference

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#polls).

In this course, we will learn:

* *statistical inference*, the process of deducing characteristics of a population using data from a random sample
* the statistical concepts necessary to define *estimates* and *margins of errors*
* how to *forecast future results* and estimate the precision of our forecast
* how to calculate and interpret *confidence intervals and p-values*

**Key points**

* Information gathered from a small random sample can be used to infer characteristics of the entire population.
* Opinion polls are useful when asking everyone in the population is impossible.
* A common use for opinion polls is determining voter preferences in political elections for the purposes of forecasting election results.
* The *spread* of a poll is the estimated difference between support two candidates or options.

# Section 1 Overview

Section 1 introduces you to parameters and estimates.

After completing Section 1, you will be able to:

* Understand how to use a sampling model to perform a poll.
* Explain the terms **population**, **parameter**, and **sample** as they relate to statistical inference.
* Use a sample to estimate the population proportion from the sample average.
* Calculate the expected value and standard error of the sample average.

## Sampling Model Parameters and Estimates

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#the-sampling-model-for-polls) and [here; first part](https://rafalab.github.io/dsbook/inference.html#populations-samples-parameters-and-estimates).

**Key points**

* The task of statistical inference is to estimate an unknown population parameter using observed data from a sample.
* In a sampling model, the collection of elements in the urn is called the *population*.
* A *parameter* is a number that summarizes data for an entire population.
* A *sample* is observed data from a subset of the population.
* An *estimate* is a summary of the observed data about a parameter that we believe is informative. It is a data-driven guess of the population parameter.
* We want to predict the proportion of the blue beads in the urn, the parameter . The proportion of red beads in the urn is and the *spread* is .
* The sample proportion is a random variable. Sampling gives random results drawn from the population distribution.

*Code: Function for taking a random draw from a specific urn*

The **dslabs** package includes a function for taking a random draw of size from the urn:

if(!require(tidyverse)) install.packages("tidyverse")

## Loading required package: tidyverse

## Warning: package 'tidyverse' was built under R version 3.6.3

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.5.0

## Warning: package 'tibble' was built under R version 3.6.3

## Warning: package 'tidyr' was built under R version 3.6.3

## Warning: package 'readr' was built under R version 3.6.3

## Warning: package 'purrr' was built under R version 3.6.3

## Warning: package 'dplyr' was built under R version 3.6.3

## Warning: package 'forcats' was built under R version 3.6.3

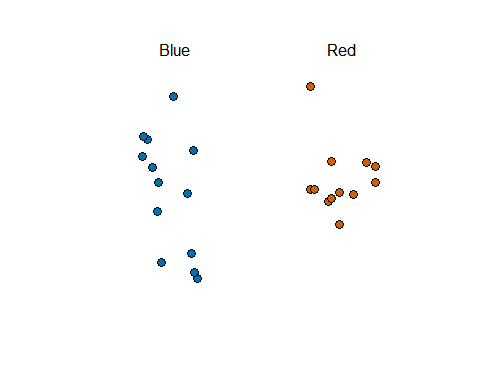
## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

if(!require(dslabs)) install.packages("dslabs")

## Loading required package: dslabs

## Warning: package 'dslabs' was built under R version 3.6.3

library(tidyverse)  
library(dslabs)  
take\_poll(25) # draw 25 beads



## The Sample Average

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#the-sample-average) and [here](https://rafalab.github.io/dsbook/inference.html#parameters).

**Key points**

* Many common data science tasks can be framed as estimating a parameter from a sample.
* We illustrate statistical inference by walking through the process to estimate . From the estimate of , we can easily calculate an estimate of the spread, .
* Consider the random variable that is 1 if a blue bead is chosen and 0 if a red bead is chosen. The proportion of blue beads in draws is the average of the draws .
* is the *sample average*. In statistics, a bar on top of a symbol denotes the average. is a random variable because it is the average of random draws - each time we take a sample, is different.
* The number of blue beads drawn in N draws, , is times the proportion of values in the urn. However, we do not know the true proportion: we are trying to estimate this parameter .

## Polling versus Forecasting

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#polling-versus-forecasting).

**Key points**

* A poll taken in advance of an election estimates for that moment, not for election day.
* In order to predict election results, forecasters try to use early estimates of to predict on election day. We discuss some approaches in later sections.

## Properties of Our Estimate

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#properties-of-our-estimate-expected-value-and-standard-error).

**Key points**

* When interpreting values of , it is important to remember that is a random variable with an expected value and standard error that represents the sample proportion of positive events.
* The expected value of is the parameter of interest . This follows from the fact that is the sum of independent draws of a random variable times a constant .
* As the number of draws increases, the standard error of our estimate decreases. The standard error of the average of over draws is:
* In theory, we can get more accurate estimates of by increasing . In practice, there are limits on the size of due to costs, as well as other factors we discuss later.
* We can also use other random variable equations to determine the expected value of the sum of draws and standard error of the sum of draws .

## Assessment - Parameters and Estimates

1. Suppose you poll a population in which a proportion of voters are Democrats and are Republicans.

Your sample size is . Consider the random variable , which is the **total** number of Democrats in your sample.

What is the expected value of this random variable ?

* ☐ A.
* ☒ B.
* ☐ C.
* ☐ D.

1. Again, consider the random variable , which is the **total** number of Democrats in your sample of 25 voters.

The variable describes the proportion of Democrats in the sample, whereas describes the proportion of Republicans.

What is the standard error of ?

* ☐ A.
* ☐ B.
* ☐ C.
* ☒ D.

1. Consider the random variable , which is equivalent to the sample average that we have been denoting as .

The variable represents the sample size and is the proportion of Democrats in the population.

What is the expected value of ?

* ☒ A.
* ☐ B.
* ☐ C.
* ☐ D.

1. What is the standard error of the sample average, ?

The variable represents the sample size and is the proportion of Democrats in the population.

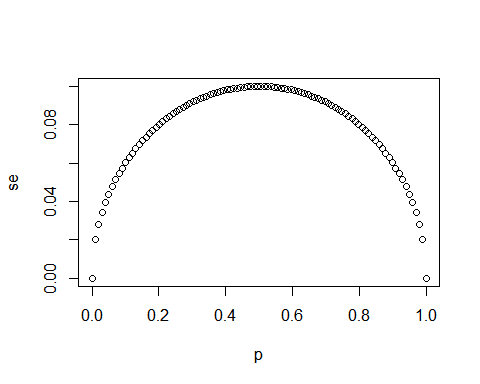
* ☐ A.
* ☒ B.
* ☐ C.
* ☐ D.

1. Write a line of code that calculates the standard error se of a sample average when you poll 25 people in the population.

Generate a sequence of 100 proportions of Democrats p that vary from 0 (no Democrats) to 1 (all Democrats).

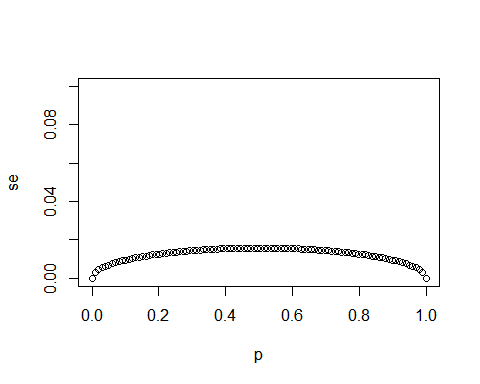
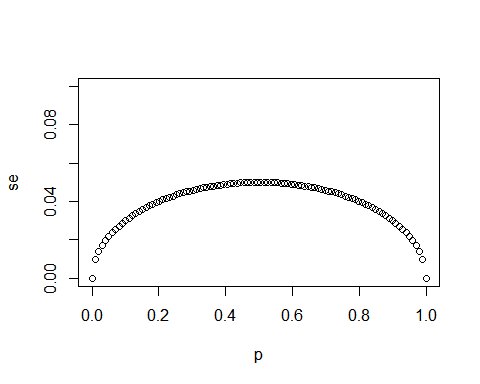
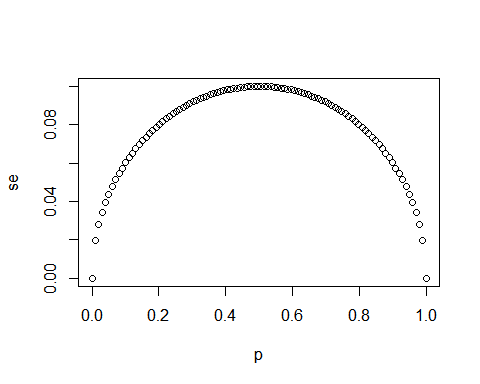
Plot se versus p for the 100 different proportions.

# `N` represents the number of people polled  
N <- 25  
  
# Create a variable `p` that contains 100 proportions ranging from 0 to 1 using the `seq` function  
p <- seq(0, 1, length.out = 100)  
  
# Create a variable `se` that contains the standard error of each sample average  
se <- sqrt(p \* (1 - p)/N)  
  
# Plot `p` on the x-axis and `se` on the y-axis  
plot(p,se)



1. Using the same code as in the previous exercise, create a for-loop that generates three plots of p versus se when the sample sizes equal , , and .

# The vector `p` contains 100 proportions of Democrats ranging from 0 to 1 using the `seq` function  
p <- seq(0, 1, length = 100)  
  
# The vector `sample\_sizes` contains the three sample sizes  
sample\_sizes <- c(25, 100, 1000)  
  
# Write a for-loop that calculates the standard error `se` for every value of `p` for each of the three samples sizes `N` in the vector `sample\_sizes`. Plot the three graphs, using the `ylim` argument to standardize the y-axis across all three plots.  
for (N in sample\_sizes)  
{  
se <- sqrt(p \* (1 - p)/N)  
plot(p,se,ylim = c(0,0.1))  
}



1. Our estimate for the difference in proportions of Democrats and Republicans is .

Which derivation correctly uses the rules we learned about sums of random variables and scaled random variables to derive the expected value of

* ☐ A.
* ☐ B.
* ☐ C.
* ☒ D.

1. Our estimate for the difference in proportions of Democrats and Republicans is .

Which derivation correctly uses the rules we learned about sums of random variables and scaled random variables to derive the standard error of ?

* ☐ A.
* ☐ B.
* ☒ C.
* ☐ D.

1. Say the actual proportion of Democratic voters is .

In this case, the Republican party is winning by a relatively large margin of , or a 10% margin of victory. What is the standard error of the spread in this case?

# `N` represents the number of people polled  
N <- 25  
  
# `p` represents the proportion of Democratic voters  
p <- 0.45  
  
# Calculate the standard error of the spread. Print this value to the console.  
2\*sqrt((p\*(1-p)/N))

## [1] 0.1989975

1. So far we have said that the difference between the proportion of Democratic voters and Republican voters is about 10% and that the standard error of this spread is about 0.2 when .

Select the statement that explains why this sample size is sufficient or not.

* ☐ A. This sample size is sufficient because the expected value of our estimate is so our prediction will be right on.
* ☒ B. This sample size is too small because the standard error is larger than the spread.
* ☐ C. This sample size is sufficient because the standard error of about 0.2 is much smaller than the spread of 10%.
* ☐ D. Without knowing p, we have no way of knowing that increasing our sample size would actually improve our standard error.

# Section 2 Overview

In Section 2, you will look at the Central Limit Theorem in practice.

After completing Section 2, you will be able to:

* Use the Central Limit Theorem to calculate the probability that a sample estimate is close to the population proportion .
* Run a Monte Carlo simulation to corroborate theoretical results built using probability theory.
* Estimate the spread based on estimates of and .
* Understand why bias can mean that larger sample sizes aren’t necessarily better.

## The Central Limit Theorem in Practice

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#clt).

**Key points**

* Because is the sum of random draws divided by a constant, the distribution of is approximately normal.
* We can convert to a standard normal random variable :
* The probability that is within .01 of the actual value of is:
* The Central Limit Theorem (CLT) still works if is used in place of . This is called a *plug-in estimate*. Hats over values denote estimates. Therefore:

Using the CLT, the probability that is within .01 of the actual value of is:

*Code: Computing the probability of being within .01 of*

X\_hat <- 0.48  
se <- sqrt(X\_hat\*(1-X\_hat)/25)  
pnorm(0.01/se) - pnorm(-0.01/se)

## [1] 0.07971926

## Margin of Error

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#clt).

**Key points**

* The *margin of error* is defined as 2 times the standard error of the estimate .
* There is about a 95% chance that will be within two standard errors of the actual parameter .

## A Monte Carlo Simulation for the CLT

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#a-monte-carlo-simulation).

**Key points**

* We can run Monte Carlo simulations to compare with theoretical results assuming a value of .
* In practice, is unknown. We can corroborate theoretical results by running Monte Carlo simulations with one or several values of .
* One practical choice for when modeling is , the observed value of in a sample.

*Code: Monte Carlo simulation using a set value of p*

p <- 0.45 # unknown p to estimate  
N <- 1000  
  
# simulate one poll of size N and determine x\_hat  
x <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))  
x\_hat <- mean(x)  
  
# simulate B polls of size N and determine average x\_hat  
B <- 10000 # number of replicates  
N <- 1000 # sample size per replicate  
x\_hat <- replicate(B, {  
 x <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))  
 mean(x)  
})

*Code: Histogram and QQ-plot of Monte Carlo results*

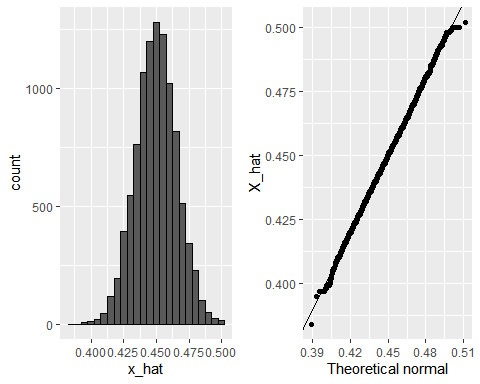
if(!require(gridExtra)) install.packages("gridExtra")

## Loading required package: gridExtra

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(gridExtra)  
p1 <- data.frame(x\_hat = x\_hat) %>%  
 ggplot(aes(x\_hat)) +  
 geom\_histogram(binwidth = 0.005, color = "black")  
p2 <- data.frame(x\_hat = x\_hat) %>%  
 ggplot(aes(sample = x\_hat)) +  
 stat\_qq(dparams = list(mean = mean(x\_hat), sd = sd(x\_hat))) +  
 geom\_abline() +  
 ylab("X\_hat") +  
 xlab("Theoretical normal")  
grid.arrange(p1, p2, nrow=1)



## The Spread

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#the-spread).

**Key points**

* The spread between two outcomes with probabilities and is .
* The expected value of the spread is .
* The standard error of the spread is .
* The margin of error of the spread is 2 times the margin of error of .

## Bias: Why Not Run a Very Large Poll?

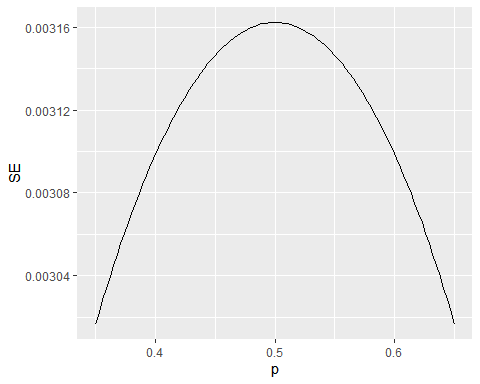
The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#bias-why-not-run-a-very-large-poll).

**Key points**

* An extremely large poll would theoretically be able to predict election results almost perfectly.
* These sample sizes are not practical. In addition to cost concerns, polling doesn’t reach everyone in the population (eventual voters) with equal probability, and it also may include data from outside our population (people who will not end up voting).
* These systematic errors in polling are called *bias*. We will learn more about bias in the future.

*Code: Plotting margin of error in an extremely large poll over a range of values of p*

N <- 100000  
p <- seq(0.35, 0.65, length = 100)  
SE <- sapply(p, function(x) 2\*sqrt(x\*(1-x)/N))  
data.frame(p = p, SE = SE) %>%  
 ggplot(aes(p, SE)) +  
 geom\_line()



## Assessment - Introduction to Inference

1. Write function called take\_sample that takes the proportion of Democrats and the sample size as arguments and returns the sample average of Democrats (1) and Republicans (0).

Calculate the sample average if the proportion of Democrats equals 0.45 and the sample size is 100.

# Write a function called `take\_sample` that takes `p` and `N` as arguements and returns the average value of a randomly sampled population.  
take\_sample <- function(p, N) {  
 x <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))  
 return(mean(x))  
}  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# Call the `take\_sample` function to determine the sample average of `N` randomly selected people from a population containing a proportion of Democrats equal to `p`. Print this value to the console.  
take\_sample(p, N)

## [1] 0.46

1. Assume the proportion of Democrats in the population equals 0.45 and that your sample size is 100 polled voters.

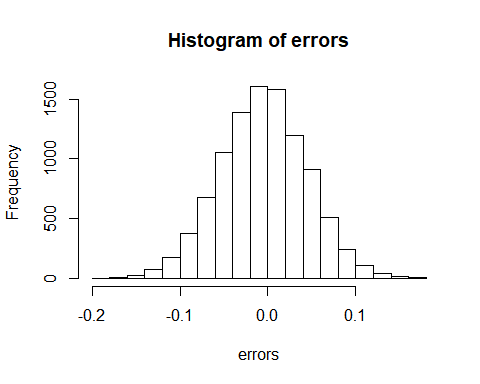
The take\_sample function you defined previously generates our estimate, .

Replicate the random sampling 10,000 times and calculate for each random sample. Save these differences as a vector called errors. Find the average of errors and plot a histogram of the distribution.

# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# The variable `B` specifies the number of times we want the sample to be replicated  
B <- 10000  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Create an objected called `errors` that replicates subtracting the result of the `take\_sample` function from `p` for `B` replications  
errors <- replicate(B, p - take\_sample(p, N))  
  
# Calculate the mean of the errors. Print this value to the console.  
mean(errors)

## [1] -4.9e-05

hist(errors)



1. In the last exercise, you made a vector of differences between the actual value for and an estimate, .

We called these differences between the actual and estimated values errors.

The errors object has already been loaded for you. Use the hist function to plot a histogram of the values contained in the vector errors. Which statement best describes the distribution of the errors?

* ☐ A. The errors are all about 0.05.
* ☐ B. The errors are all about -0.05.
* ☒ C. The errors are symmetrically distributed around 0.
* ☐ D. The errors range from -1 to 1.

1. The error is a random variable.

In practice, the error is not observed because we do not know the actual proportion of Democratic voters, . However, we can describe the size of the error by constructing a simulation.

What is the average size of the error if we define the size by taking the absolute value ?

# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# The variable `B` specifies the number of times we want the sample to be replicated  
B <- 10000  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# We generated `errors` by subtracting the estimate from the actual proportion of Democratic voters  
errors <- replicate(B, p - take\_sample(p, N))  
  
# Calculate the mean of the absolute value of each simulated error. Print this value to the console.  
mean(abs(errors))

## [1] 0.039267

1. The standard error is related to the typical **size** of the error we make when predicting.

We say **size** because, as we just saw, the errors are centered around 0. In that sense, the typical error is 0. For mathematical reasons related to the central limit theorem, we actually use the standard deviation of errors rather than the average of the absolute values.

As we have discussed, the standard error is the square root of the average squared distance . The standard deviation is defined as the square root of the distance squared.

Calculate the standard deviation of the spread.

# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# The variable `B` specifies the number of times we want the sample to be replicated  
B <- 10000  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# We generated `errors` by subtracting the estimate from the actual proportion of Democratic voters  
errors <- replicate(B, p - take\_sample(p, N))  
  
# Calculate the standard deviation of `errors`  
sqrt(mean(errors^2))

## [1] 0.04949939

1. The theory we just learned tells us what this standard deviation is going to be because it is the standard error of .

Estimate the standard error given an expected value of 0.45 and a sample size of 100.

# Define `p` as the expected value equal to 0.45  
p <- 0.45  
  
# Define `N` as the sample size  
N <- 100  
  
# Calculate the standard error  
sqrt(p\*(1-p)/N)

## [1] 0.04974937

1. In practice, we don’t know , so we construct an estimate of the theoretical prediction based by plugging in for . Calculate the standard error of the estimate:

# Define `p` as a proportion of Democratic voters to simulate  
p <- 0.45  
  
# Define `N` as the sample size  
N <- 100  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Define `X` as a random sample of `N` voters with a probability of picking a Democrat ('1') equal to `p`  
X <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))  
  
# Define `X\_bar` as the average sampled proportion  
X\_bar <- mean(X)  
  
# Calculate the standard error of the estimate. Print the result to the console.  
se <- sqrt((X\_bar\*(1-X\_bar)/N))  
se

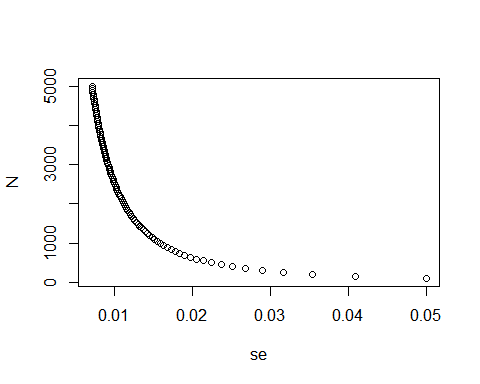
## [1] 0.04983974

1. The standard error estimates obtained from the Monte Carlo simulation, the theoretical prediction, and the estimate of the theoretical prediction are all very close, which tells us that the theory is working.

This gives us a practical approach to knowing the typical error we will make if we predict with . The theoretical result gives us an idea of how large a sample size is required to obtain the precision we need. Earlier we learned that the largest standard errors occur for .

Create a plot of the largest standard error for ranging from 100 to 5,000.

N <- seq(100, 5000, len = 100)  
p <- 0.5  
se <- sqrt(p\*(1-p)/N)  
plot(se, N)



Based on this plot, how large does the sample size have to be to have a standard error of about 1%?

* ☐ A. 100
* ☐ B. 500
* ☒ C. 2,500
* ☐ D. 4,000

1. For , the central limit theorem tells us that the distribution of is…

* ☐ A. practically equal to .
* ☒ B. approximately normal with expected value and standard error .
* ☐ C. approximately normal with expected value and standard error .
* ☐ D. not a random variable.

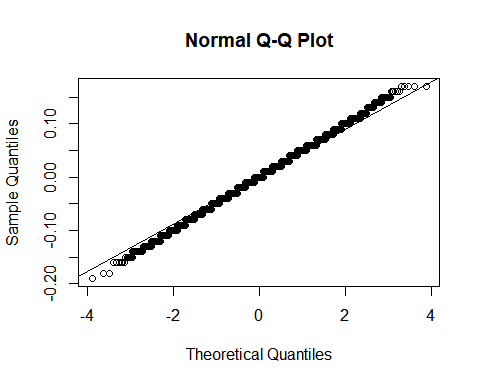
1. We calculated a vector errors that contained, for each simulated sample, the difference between the actual value p and our estimate .

The errors are:

* ☐ A. practically equal to 0.
* ☒ B. approximately normal with expected value 0 and standard error .
* ☐ C. approximately normal with expected value p and standard error .
* ☐ D. not a random variable.

1. Make a qq-plot of the errors you generated previously to see if they follow a normal distribution.

# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# The variable `B` specifies the number of times we want the sample to be replicated  
B <- 10000  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Generate `errors` by subtracting the estimate from the actual proportion of Democratic voters  
errors <- replicate(B, p - take\_sample(p, N))  
  
# Generate a qq-plot of `errors` with a qq-line showing a normal distribution  
qqnorm(errors)  
qqline(errors)



1. If and , use the central limit theorem to estimate the probability that .

# Define `p` as the proportion of Democrats in the population being polled  
p <- 0.45  
  
# Define `N` as the number of people polled  
N <- 100  
  
# Calculate the probability that the estimated proportion of Democrats in the population is greater than 0.5. Print this value to the console.  
  
1-pnorm(0.5, p, sqrt(p\*(1-p)/N))

## [1] 0.1574393

1. Assume you are in a practical situation and you don’t know .

Take a sample of size and obtain a sample average of .

What is the CLT approximation for the probability that your error size is equal or larger than 0.01?

# Define `N` as the number of people polled  
N <-100  
  
# Define `X\_hat` as the sample average  
X\_hat <- 0.51  
  
# Define `se\_hat` as the standard error of the sample average  
se\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)  
  
# Calculate the probability that the error is 0.01 or larger  
1-pnorm(0.01,0,se\_hat) + pnorm(-0.01,0,se\_hat)

## [1] 0.8414493

# Section 3 Overview

In Section 3, you will look at confidence intervals and p-values.

After completing Section 3, you will be able to:

* Calculate confidence intervals of difference sizes around an estimate.
* Understand that a confidence interval is a random interval with the given probability of falling on top of the parameter.
* Explain the concept of “power” as it relates to inference.
* Understand the relationship between p-values and confidence intervals and explain why reporting confidence intervals is often preferable.

## Confidence Intervals

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#confidence-intervals).

**Key points**

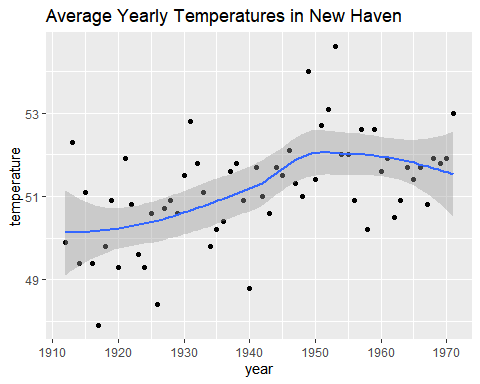
* We can use statistical theory to compute the probability that a given interval contains the true parameter .
* 95% confidence intervals are intervals constructed to have a 95% chance of including . The margin of error is approximately a 95% confidence interval.
* The start and end of these confidence intervals are random variables.
* To calculate any size confidence interval, we need to calculate the value for which equals the desired confidence. For example, a 99% confidence interval requires calculating for .
* For a confidence interval of size , we solve for .
* To determine a 95% confidence interval, use z <- qnorm(0.975). This value is slightly smaller than 2 times the standard error.

*Code: geom\_smooth confidence interval example*

The shaded area around the curve is related to the concept of confidence intervals.

data("nhtemp")  
data.frame(year = as.numeric(time(nhtemp)), temperature = as.numeric(nhtemp)) %>%  
 ggplot(aes(year, temperature)) +  
 geom\_point() +  
 geom\_smooth() +  
 ggtitle("Average Yearly Temperatures in New Haven")

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



*Code: Monte Carlo simulation of confidence intervals*

Note that to compute the exact 95% confidence interval, we would use qnorm(.975)\*SE\_hat instead of 2\*SE\_hat.

p <- 0.45  
N <- 1000  
X <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p)) # generate N observations  
X\_hat <- mean(X) # calculate X\_hat  
SE\_hat <- sqrt(X\_hat\*(1-X\_hat)/N) # calculate SE\_hat, SE of the mean of N observations  
c(X\_hat - 2\*SE\_hat, X\_hat + 2\*SE\_hat) # build interval of 2\*SE above and below mean

## [1] 0.4135691 0.4764309

*Code: Solving for with qnorm*

z <- qnorm(0.995) # calculate z to solve for 99% confidence interval  
pnorm(qnorm(0.995)) # demonstrating that qnorm gives the z value for a given probability

## [1] 0.995

pnorm(qnorm(1-0.995)) # demonstrating symmetry of 1-qnorm

## [1] 0.005

pnorm(z) - pnorm(-z) # demonstrating that this z value gives correct probability for interval

## [1] 0.99

## A Monte Carlo Simulation for Confidence Intervals

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#a-monte-carlo-simulation-1).

**Key points**

* We can run a Monte Carlo simulation to confirm that a 95% confidence interval contains the true value of 95% of the time.
* A plot of confidence intervals from this simulation demonstrates that most intervals include , but roughly 5% of intervals miss the true value of .

*Code: Monte Carlo simulation*

Note that to compute the exact 95% confidence interval, we would use qnorm(.975)\*SE\_hat instead of 2\*SE\_hat.

B <- 10000  
inside <- replicate(B, {  
 X <- sample(c(0,1), size = N, replace = TRUE, prob = c(1-p, p))  
 X\_hat <- mean(X)  
 SE\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)  
 between(p, X\_hat - 2\*SE\_hat, X\_hat + 2\*SE\_hat) # TRUE if p in confidence interval  
})  
mean(inside)

## [1] 0.9566

## The Correct Language

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#the-correct-language).

**Key points**

* The 95% confidence intervals are random, but is not random.
* 95% refers to the probability that the random interval falls on top of .
* It is technically incorrect to state that has a 95% chance of being in between two values because that implies is random.

## Power

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#power).

**Key points**

* If we are trying to predict the result of an election, then a confidence interval that includes a spread of 0 (a tie) is not helpful.
* A confidence interval that includes a spread of 0 does not imply a close election, it means the sample size is too small.
* Power is the probability of detecting an effect when there is a true effect to find. Power increases as sample size increases, because larger sample size means smaller standard error.

*Code: Confidence interval for the spread with sample size of 25*

Note that to compute the exact 95% confidence interval, we would use c(-qnorm(.975), qnorm(.975)) instead of 1.96.

N <- 25  
X\_hat <- 0.48  
(2\*X\_hat - 1) + c(-2, 2)\*2\*sqrt(X\_hat\*(1-X\_hat)/N)

## [1] -0.4396799 0.3596799

## p-Values

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#p-values).

**Key points**

* The null hypothesis is the hypothesis that there is no effect. In this case, the null hypothesis is that the spread is 0, or .
* The p-value is the probability of detecting an effect of a certain size or larger when the null hypothesis is true.
* We can convert the probability of seeing an observed value under the null hypothesis into a standard normal random variable. We compute the value of that corresponds to the observed result, and then use that to compute the p-value.
* If a 95% confidence interval does not include our observed value, then the p-value must be smaller than 0.05.
* It is preferable to report confidence intervals instead of p-values, as confidence intervals give information about the size of the estimate and p-values do not.

*Code: Computing a p-value for observed spread of 0.02*

N <- 100 # sample size  
z <- sqrt(N) \* 0.02/0.5 # spread of 0.02  
1 - (pnorm(z) - pnorm(-z))

## [1] 0.6891565

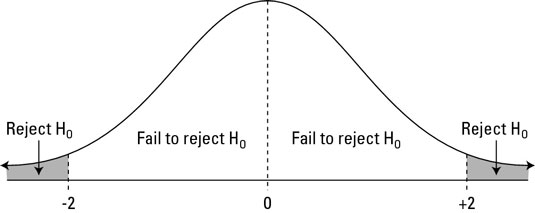
## Another Explanation of p-Values

The p-value is the probability of observing a value as extreme or more extreme than the result given that the null hypothesis is true.

In the context of the normal distribution, this refers to the probability of observing a Z-score whose absolute value is as high or higher than the Z-score of interest.

Suppose we want to find the p-value of an observation 2 standard deviations larger than the mean. This means we are looking for anything with .

Graphically, the p-value gives the probability of an observation that’s at least as far away from the mean or further. This plot shows a standard normal distribution (centered at ) with a standard deviation of 1). The shaded tails are the region of the graph that are 2 standard deviations or more away from the mean.



Standard normal distribution (centered at z=0 with a standard deviation of 1

The p-value is the proportion of area under a normal curve that has z-scores as extreme or more extreme than the given value - the tails on this plot of a normal distribution are shaded to show the region corresponding to the p-value.

The right tail can be found with 1-pnorm(2). We want to have both tails, though, because we want to find the probability of any observation as far away from the mean or farther, in either direction. (This is what’s meant by a two-tailed p-value.) Because the distribution is symmetrical, the right and left tails are the same size and we know that our desired value is just 2\*(1-pnorm(2)).

Recall that, by default, pnorm() gives the CDF for a normal distribution with a mean of and standard deviation of . To find p-values for a given z-score z in a normal distribution with mean mu and standard deviation sigma, use 2\*(1-pnorm(z, mu, sigma)) instead.

## Assessment - Confidence Intervals and p-Values

1. For the following exercises, we will use actual poll data from the 2016 election.

The exercises will contain pre-loaded data from the dslabs package.

library(dslabs)  
data("polls\_us\_election\_2016")

We will use all the national polls that ended within a few weeks before the election.

Assume there are only two candidates and construct a 95% confidence interval for the election night proportion .

# Load the data  
data(polls\_us\_election\_2016)  
  
# Generate an object `polls` that contains data filtered for polls that ended on or after October 31, 2016 in the United States  
polls <- filter(polls\_us\_election\_2016, enddate >= "2016-10-31" & state == "U.S.")  
  
# How many rows does `polls` contain? Print this value to the console.  
nrow(polls)

## [1] 70

# Assign the sample size of the first poll in `polls` to a variable called `N`. Print this value to the console.  
N <- head(polls$samplesize,1)  
N

## [1] 2220

# For the first poll in `polls`, assign the estimated percentage of Clinton voters to a variable called `X\_hat`. Print this value to the console.  
X\_hat <- (head(polls$rawpoll\_clinton,1)/100)  
X\_hat

## [1] 0.47

# Calculate the standard error of `X\_hat` and save it to a variable called `se\_hat`. Print this value to the console.  
se\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)  
se\_hat

## [1] 0.01059279

# Use `qnorm` to calculate the 95% confidence interval for the proportion of Clinton voters. Save the lower and then the upper confidence interval to a variable called `ci`.  
ci <- c(X\_hat - qnorm(0.975)\*se\_hat, X\_hat + qnorm(0.975)\*se\_hat)  
ci

## [1] 0.4492385 0.4907615

1. Create a new object called pollster\_results that contains the pollster’s name, the end date of the poll, the proportion of voters who declared a vote for Clinton, the standard error of this estimate, and the lower and upper bounds of the confidence interval for the estimate.

# The `polls` object that filtered all the data by date and nation has already been loaded. Examine it using the `head` function.  
head(polls)

## state startdate enddate  
## 1 U.S. 2016-11-03 2016-11-06  
## 2 U.S. 2016-11-01 2016-11-07  
## 3 U.S. 2016-11-02 2016-11-06  
## 4 U.S. 2016-11-04 2016-11-07  
## 5 U.S. 2016-11-03 2016-11-06  
## 6 U.S. 2016-11-03 2016-11-06  
## pollster grade samplesize  
## 1 ABC News/Washington Post A+ 2220  
## 2 Google Consumer Surveys B 26574  
## 3 Ipsos A- 2195  
## 4 YouGov B 3677  
## 5 Gravis Marketing B- 16639  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research A 1295  
## population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson rawpoll\_mcmullin  
## 1 lv 47.00 43.00 4.00 NA  
## 2 lv 38.03 35.69 5.46 NA  
## 3 lv 42.00 39.00 6.00 NA  
## 4 lv 45.00 41.00 5.00 NA  
## 5 rv 47.00 43.00 3.00 NA  
## 6 lv 48.00 44.00 3.00 NA  
## adjpoll\_clinton adjpoll\_trump adjpoll\_johnson adjpoll\_mcmullin  
## 1 45.20163 41.72430 4.626221 NA  
## 2 43.34557 41.21439 5.175792 NA  
## 3 42.02638 38.81620 6.844734 NA  
## 4 45.65676 40.92004 6.069454 NA  
## 5 46.84089 42.33184 3.726098 NA  
## 6 49.02208 43.95631 3.057876 NA

# Create a new object called `pollster\_results` that contains columns for pollster name, end date, X\_hat, se\_hat, lower confidence interval, and upper confidence interval for each poll.  
polls <- mutate(polls, X\_hat = polls$rawpoll\_clinton/100, se\_hat = sqrt(X\_hat\*(1-X\_hat)/polls$samplesize), lower = X\_hat - qnorm(0.975)\*se\_hat, upper = X\_hat + qnorm(0.975)\*se\_hat)  
pollster\_results <- select(polls, pollster, enddate, X\_hat, se\_hat, lower, upper)  
pollster\_results

## pollster enddate X\_hat  
## 1 ABC News/Washington Post 2016-11-06 0.4700  
## 2 Google Consumer Surveys 2016-11-07 0.3803  
## 3 Ipsos 2016-11-06 0.4200  
## 4 YouGov 2016-11-07 0.4500  
## 5 Gravis Marketing 2016-11-06 0.4700  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-06 0.4800  
## 7 CBS News/New York Times 2016-11-06 0.4500  
## 8 NBC News/Wall Street Journal 2016-11-05 0.4400  
## 9 IBD/TIPP 2016-11-07 0.4120  
## 10 Selzer & Company 2016-11-06 0.4400  
## 11 Angus Reid Global 2016-11-04 0.4800  
## 12 Monmouth University 2016-11-06 0.5000  
## 13 Marist College 2016-11-03 0.4400  
## 14 The Times-Picayune/Lucid 2016-11-07 0.4500  
## 15 USC Dornsife/LA Times 2016-11-07 0.4361  
## 16 RKM Research and Communications, Inc. 2016-11-05 0.4760  
## 17 CVOTER International 2016-11-06 0.4891  
## 18 Morning Consult 2016-11-05 0.4500  
## 19 SurveyMonkey 2016-11-06 0.4700  
## 20 Rasmussen Reports/Pulse Opinion Research 2016-11-06 0.4500  
## 21 Insights West 2016-11-07 0.4900  
## 22 RAND (American Life Panel) 2016-11-01 0.4370  
## 23 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-03 0.4550  
## 24 CBS News/New York Times 2016-11-01 0.4500  
## 25 ABC News/Washington Post 2016-11-05 0.4700  
## 26 Ipsos 2016-11-04 0.4300  
## 27 ABC News/Washington Post 2016-11-04 0.4800  
## 28 YouGov 2016-11-06 0.4290  
## 29 IBD/TIPP 2016-11-06 0.4070  
## 30 ABC News/Washington Post 2016-11-03 0.4700  
## 31 IBD/TIPP 2016-11-03 0.4440  
## 32 IBD/TIPP 2016-11-05 0.4300  
## 33 ABC News/Washington Post 2016-11-02 0.4700  
## 34 ABC News/Washington Post 2016-11-01 0.4700  
## 35 ABC News/Washington Post 2016-10-31 0.4600  
## 36 Ipsos 2016-11-03 0.4320  
## 37 IBD/TIPP 2016-11-04 0.4420  
## 38 YouGov 2016-11-01 0.4600  
## 39 IBD/TIPP 2016-10-31 0.4460  
## 40 Ipsos 2016-11-02 0.4550  
## 41 Rasmussen Reports/Pulse Opinion Research 2016-11-03 0.4400  
## 42 The Times-Picayune/Lucid 2016-11-06 0.4500  
## 43 Ipsos 2016-11-01 0.4470  
## 44 IBD/TIPP 2016-11-02 0.4400  
## 45 IBD/TIPP 2016-11-01 0.4400  
## 46 Rasmussen Reports/Pulse Opinion Research 2016-11-02 0.4200  
## 47 Ipsos 2016-10-31 0.4400  
## 48 The Times-Picayune/Lucid 2016-11-05 0.4500  
## 49 Rasmussen Reports/Pulse Opinion Research 2016-10-31 0.4400  
## 50 Google Consumer Surveys 2016-10-31 0.3769  
## 51 CVOTER International 2016-11-05 0.4925  
## 52 Rasmussen Reports/Pulse Opinion Research 2016-11-01 0.4400  
## 53 CVOTER International 2016-11-04 0.4906  
## 54 The Times-Picayune/Lucid 2016-11-04 0.4500  
## 55 USC Dornsife/LA Times 2016-11-06 0.4323  
## 56 CVOTER International 2016-11-03 0.4853  
## 57 The Times-Picayune/Lucid 2016-11-03 0.4400  
## 58 USC Dornsife/LA Times 2016-11-05 0.4263  
## 59 CVOTER International 2016-11-02 0.4878  
## 60 USC Dornsife/LA Times 2016-11-04 0.4256  
## 61 CVOTER International 2016-11-01 0.4881  
## 62 The Times-Picayune/Lucid 2016-11-02 0.4400  
## 63 Gravis Marketing 2016-10-31 0.4600  
## 64 USC Dornsife/LA Times 2016-11-03 0.4338  
## 65 The Times-Picayune/Lucid 2016-11-01 0.4300  
## 66 USC Dornsife/LA Times 2016-11-02 0.4247  
## 67 Gravis Marketing 2016-11-02 0.4700  
## 68 USC Dornsife/LA Times 2016-11-01 0.4236  
## 69 The Times-Picayune/Lucid 2016-10-31 0.4200  
## 70 USC Dornsife/LA Times 2016-10-31 0.4328  
## se\_hat lower upper  
## 1 0.010592790 0.4492385 0.4907615  
## 2 0.002978005 0.3744632 0.3861368  
## 3 0.010534681 0.3993524 0.4406476  
## 4 0.008204286 0.4339199 0.4660801  
## 5 0.003869218 0.4624165 0.4775835  
## 6 0.013883131 0.4527896 0.5072104  
## 7 0.013174309 0.4241788 0.4758212  
## 8 0.013863610 0.4128278 0.4671722  
## 9 0.014793245 0.3830058 0.4409942  
## 10 0.017560908 0.4055813 0.4744187  
## 11 0.014725994 0.4511376 0.5088624  
## 12 0.018281811 0.4641683 0.5358317  
## 13 0.016190357 0.4082675 0.4717325  
## 14 0.009908346 0.4305800 0.4694200  
## 15 0.009096403 0.4182714 0.4539286  
## 16 0.015722570 0.4451843 0.5068157  
## 17 0.012400526 0.4647954 0.5134046  
## 18 0.012923005 0.4246714 0.4753286  
## 19 0.001883809 0.4663078 0.4736922  
## 20 0.012845233 0.4248238 0.4751762  
## 21 0.016304940 0.4580429 0.5219571  
## 22 0.010413043 0.4165908 0.4574092  
## 23 0.014966841 0.4256655 0.4843345  
## 24 0.016339838 0.4179745 0.4820255  
## 25 0.011340235 0.4477735 0.4922265  
## 26 0.010451057 0.4095163 0.4504837  
## 27 0.012170890 0.4561455 0.5038545  
## 28 0.001704722 0.4256588 0.4323412  
## 29 0.015337369 0.3769393 0.4370607  
## 30 0.013249383 0.4440317 0.4959683  
## 31 0.016580236 0.4115033 0.4764967  
## 32 0.016475089 0.3977094 0.4622906  
## 33 0.014711237 0.4411665 0.4988335  
## 34 0.014610041 0.4413648 0.4986352  
## 35 0.014496630 0.4315871 0.4884129  
## 36 0.011018764 0.4104036 0.4535964  
## 37 0.017514599 0.4076720 0.4763280  
## 38 0.014193655 0.4321809 0.4878191  
## 39 0.015579317 0.4154651 0.4765349  
## 40 0.011358664 0.4327374 0.4772626  
## 41 0.012816656 0.4148798 0.4651202  
## 42 0.009786814 0.4308182 0.4691818  
## 43 0.011810940 0.4238510 0.4701490  
## 44 0.016858185 0.4069586 0.4730414  
## 45 0.016907006 0.4068629 0.4731371  
## 46 0.012743626 0.3950230 0.4449770  
## 47 0.012973281 0.4145728 0.4654272  
## 48 0.009898535 0.4305992 0.4694008  
## 49 0.012816656 0.4148798 0.4651202  
## 50 0.003107749 0.3708089 0.3829911  
## 51 0.012609413 0.4677860 0.5172140  
## 52 0.012816656 0.4148798 0.4651202  
## 53 0.012998976 0.4651225 0.5160775  
## 54 0.009830663 0.4307323 0.4692677  
## 55 0.009144248 0.4143776 0.4502224  
## 56 0.013381202 0.4590733 0.5115267  
## 57 0.009684792 0.4210182 0.4589818  
## 58 0.009047108 0.4085680 0.4440320  
## 59 0.013711286 0.4609264 0.5146736  
## 60 0.009046705 0.4078688 0.4433312  
## 61 0.013441133 0.4617559 0.5144441  
## 62 0.009697721 0.4209928 0.4590072  
## 63 0.006807590 0.4466574 0.4733426  
## 64 0.009106200 0.4159522 0.4516478  
## 65 0.009677647 0.4110322 0.4489678  
## 66 0.009119319 0.4068265 0.4425735  
## 67 0.010114336 0.4501763 0.4898237  
## 68 0.009015504 0.4059299 0.4412701  
## 69 0.009679479 0.4010286 0.4389714  
## 70 0.008834895 0.4154839 0.4501161

1. The final tally for the popular vote was Clinton 48.2% and Trump 46.1%. Add a column called hit to pollster\_results that states if the confidence interval included the true proportion or not. What proportion of confidence intervals included ?

# The `pollster\_results` object has already been loaded. Examine it using the `head` function.  
head(pollster\_results)

## pollster enddate X\_hat  
## 1 ABC News/Washington Post 2016-11-06 0.4700  
## 2 Google Consumer Surveys 2016-11-07 0.3803  
## 3 Ipsos 2016-11-06 0.4200  
## 4 YouGov 2016-11-07 0.4500  
## 5 Gravis Marketing 2016-11-06 0.4700  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-06 0.4800  
## se\_hat lower upper  
## 1 0.010592790 0.4492385 0.4907615  
## 2 0.002978005 0.3744632 0.3861368  
## 3 0.010534681 0.3993524 0.4406476  
## 4 0.008204286 0.4339199 0.4660801  
## 5 0.003869218 0.4624165 0.4775835  
## 6 0.013883131 0.4527896 0.5072104

# Add a logical variable called `hit` that indicates whether the actual value exists within the confidence interval of each poll. Summarize the average `hit` result to determine the proportion of polls with confidence intervals include the actual value. Save the result as an object called `avg\_hit`.  
avg\_hit <- pollster\_results %>% mutate(hit=(lower<0.482 & upper>0.482)) %>% summarize(mean(hit))  
avg\_hit

## mean(hit)  
## 1 0.3142857

1. If these confidence intervals are constructed correctly, and the theory holds up, what proportion of confidence intervals should include ?

* ☐ A. 0.05
* ☐ B. 0.31
* ☐ C. 0.50
* ☒ D. 0.95

1. A much smaller proportion of the polls than expected produce confidence intervals containing .

Notice that most polls that fail to include are underestimating. The rationale for this is that undecided voters historically divide evenly between the two main candidates on election day.

In this case, it is more informative to estimate the spread or the difference between the proportion of two candidates , or for this election.

Assume that there are only two parties and that . Construct a 95% confidence interval for difference in proportions on election night.

# Add a statement to this line of code that will add a new column named `d\_hat` to `polls`. The new column should contain the difference in the proportion of voters.  
polls <- polls\_us\_election\_2016 %>% filter(enddate >= "2016-10-31" & state == "U.S.") %>%  
 mutate(d\_hat = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
# Assign the sample size of the first poll in `polls` to a variable called `N`. Print this value to the console.  
N <- polls$samplesize[1]  
N

## [1] 2220

# Assign the difference `d\_hat` of the first poll in `polls` to a variable called `d\_hat`. Print this value to the console.  
d\_hat <- polls$d\_hat[1]  
d\_hat

## [1] 0.04

# Assign proportion of votes for Clinton to the variable `X\_hat`.  
X\_hat <- (d\_hat+1)/2  
X\_hat

## [1] 0.52

# Calculate the standard error of the spread and save it to a variable called `se\_hat`. Print this value to the console.  
se\_hat <- 2\*sqrt(X\_hat\*(1-X\_hat)/N)  
se\_hat

## [1] 0.02120683

# Use `qnorm` to calculate the 95% confidence interval for the difference in the proportions of voters. Save the lower and then the upper confidence interval to a variable called `ci`.  
ci <- c(d\_hat - qnorm(0.975)\*se\_hat, d\_hat + qnorm(0.975)\*se\_hat)  
ci

## [1] -0.001564627 0.081564627

1. Create a new object called pollster\_results that contains the pollster’s name, the end date of the poll, the difference in the proportion of voters who declared a vote either, and the lower and upper bounds of the confidence interval for the estimate.

# The subset `polls` data with 'd\_hat' already calculated has been loaded. Examine it using the `head` function.  
head(polls)

## state startdate enddate  
## 1 U.S. 2016-11-03 2016-11-06  
## 2 U.S. 2016-11-01 2016-11-07  
## 3 U.S. 2016-11-02 2016-11-06  
## 4 U.S. 2016-11-04 2016-11-07  
## 5 U.S. 2016-11-03 2016-11-06  
## 6 U.S. 2016-11-03 2016-11-06  
## pollster grade samplesize  
## 1 ABC News/Washington Post A+ 2220  
## 2 Google Consumer Surveys B 26574  
## 3 Ipsos A- 2195  
## 4 YouGov B 3677  
## 5 Gravis Marketing B- 16639  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research A 1295  
## population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson rawpoll\_mcmullin  
## 1 lv 47.00 43.00 4.00 NA  
## 2 lv 38.03 35.69 5.46 NA  
## 3 lv 42.00 39.00 6.00 NA  
## 4 lv 45.00 41.00 5.00 NA  
## 5 rv 47.00 43.00 3.00 NA  
## 6 lv 48.00 44.00 3.00 NA  
## adjpoll\_clinton adjpoll\_trump adjpoll\_johnson adjpoll\_mcmullin d\_hat  
## 1 45.20163 41.72430 4.626221 NA 0.0400  
## 2 43.34557 41.21439 5.175792 NA 0.0234  
## 3 42.02638 38.81620 6.844734 NA 0.0300  
## 4 45.65676 40.92004 6.069454 NA 0.0400  
## 5 46.84089 42.33184 3.726098 NA 0.0400  
## 6 49.02208 43.95631 3.057876 NA 0.0400

# Create a new object called `pollster\_results` that contains columns for pollster name, end date, d\_hat, lower confidence interval of d\_hat, and upper confidence interval of d\_hat for each poll.  
d\_hat = polls$rawpoll\_clinton/100 - polls$rawpoll\_trump/100  
X\_hat = (d\_hat + 1) / 2  
polls <- mutate(polls, X\_hat, se\_hat = 2 \* sqrt(X\_hat \* (1 - X\_hat) / samplesize), lower = d\_hat - qnorm(0.975)\*se\_hat, upper = d\_hat + qnorm(0.975)\*se\_hat)  
pollster\_results <- select(polls, pollster, enddate, d\_hat, lower, upper)  
pollster\_results

## pollster enddate  
## 1 ABC News/Washington Post 2016-11-06  
## 2 Google Consumer Surveys 2016-11-07  
## 3 Ipsos 2016-11-06  
## 4 YouGov 2016-11-07  
## 5 Gravis Marketing 2016-11-06  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-06  
## 7 CBS News/New York Times 2016-11-06  
## 8 NBC News/Wall Street Journal 2016-11-05  
## 9 IBD/TIPP 2016-11-07  
## 10 Selzer & Company 2016-11-06  
## 11 Angus Reid Global 2016-11-04  
## 12 Monmouth University 2016-11-06  
## 13 Marist College 2016-11-03  
## 14 The Times-Picayune/Lucid 2016-11-07  
## 15 USC Dornsife/LA Times 2016-11-07  
## 16 RKM Research and Communications, Inc. 2016-11-05  
## 17 CVOTER International 2016-11-06  
## 18 Morning Consult 2016-11-05  
## 19 SurveyMonkey 2016-11-06  
## 20 Rasmussen Reports/Pulse Opinion Research 2016-11-06  
## 21 Insights West 2016-11-07  
## 22 RAND (American Life Panel) 2016-11-01  
## 23 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-03  
## 24 CBS News/New York Times 2016-11-01  
## 25 ABC News/Washington Post 2016-11-05  
## 26 Ipsos 2016-11-04  
## 27 ABC News/Washington Post 2016-11-04  
## 28 YouGov 2016-11-06  
## 29 IBD/TIPP 2016-11-06  
## 30 ABC News/Washington Post 2016-11-03  
## 31 IBD/TIPP 2016-11-03  
## 32 IBD/TIPP 2016-11-05  
## 33 ABC News/Washington Post 2016-11-02  
## 34 ABC News/Washington Post 2016-11-01  
## 35 ABC News/Washington Post 2016-10-31  
## 36 Ipsos 2016-11-03  
## 37 IBD/TIPP 2016-11-04  
## 38 YouGov 2016-11-01  
## 39 IBD/TIPP 2016-10-31  
## 40 Ipsos 2016-11-02  
## 41 Rasmussen Reports/Pulse Opinion Research 2016-11-03  
## 42 The Times-Picayune/Lucid 2016-11-06  
## 43 Ipsos 2016-11-01  
## 44 IBD/TIPP 2016-11-02  
## 45 IBD/TIPP 2016-11-01  
## 46 Rasmussen Reports/Pulse Opinion Research 2016-11-02  
## 47 Ipsos 2016-10-31  
## 48 The Times-Picayune/Lucid 2016-11-05  
## 49 Rasmussen Reports/Pulse Opinion Research 2016-10-31  
## 50 Google Consumer Surveys 2016-10-31  
## 51 CVOTER International 2016-11-05  
## 52 Rasmussen Reports/Pulse Opinion Research 2016-11-01  
## 53 CVOTER International 2016-11-04  
## 54 The Times-Picayune/Lucid 2016-11-04  
## 55 USC Dornsife/LA Times 2016-11-06  
## 56 CVOTER International 2016-11-03  
## 57 The Times-Picayune/Lucid 2016-11-03  
## 58 USC Dornsife/LA Times 2016-11-05  
## 59 CVOTER International 2016-11-02  
## 60 USC Dornsife/LA Times 2016-11-04  
## 61 CVOTER International 2016-11-01  
## 62 The Times-Picayune/Lucid 2016-11-02  
## 63 Gravis Marketing 2016-10-31  
## 64 USC Dornsife/LA Times 2016-11-03  
## 65 The Times-Picayune/Lucid 2016-11-01  
## 66 USC Dornsife/LA Times 2016-11-02  
## 67 Gravis Marketing 2016-11-02  
## 68 USC Dornsife/LA Times 2016-11-01  
## 69 The Times-Picayune/Lucid 2016-10-31  
## 70 USC Dornsife/LA Times 2016-10-31  
## d\_hat lower upper  
## 1 0.0400 -0.001564627 0.0815646272  
## 2 0.0234 0.011380104 0.0354198955  
## 3 0.0300 -0.011815309 0.0718153088  
## 4 0.0400 0.007703641 0.0722963589  
## 5 0.0400 0.024817728 0.0551822719  
## 6 0.0400 -0.014420872 0.0944208716  
## 7 0.0400 -0.011860967 0.0918609675  
## 8 0.0400 -0.014696100 0.0946961005  
## 9 -0.0150 -0.073901373 0.0439013728  
## 10 0.0300 -0.039307332 0.0993073320  
## 11 0.0400 -0.017724837 0.0977248370  
## 12 0.0600 -0.011534270 0.1315342703  
## 13 0.0100 -0.053923780 0.0739237800  
## 14 0.0500 0.011013152 0.0889868476  
## 15 -0.0323 -0.068233293 0.0036332933  
## 16 0.0320 -0.029670864 0.0936708643  
## 17 0.0278 -0.020801931 0.0764019309  
## 18 0.0300 -0.020889533 0.0808895334  
## 19 0.0600 0.052615604 0.0673843956  
## 20 0.0200 -0.030595930 0.0705959303  
## 21 0.0400 -0.023875814 0.1038758144  
## 22 0.0910 0.050024415 0.1319755848  
## 23 0.0150 -0.043901373 0.0739013728  
## 24 0.0300 -0.034344689 0.0943446886  
## 25 0.0400 -0.004497495 0.0844974954  
## 26 0.0400 -0.001341759 0.0813417590  
## 27 0.0500 0.002312496 0.0976875039  
## 28 0.0390 0.032254341 0.0457456589  
## 29 -0.0240 -0.085171524 0.0371715236  
## 30 0.0400 -0.011988727 0.0919887265  
## 31 0.0050 -0.060404028 0.0704040277  
## 32 -0.0100 -0.075220256 0.0552202561  
## 33 0.0300 -0.027745070 0.0877450695  
## 34 0.0200 -0.037362198 0.0773621983  
## 35 0.0000 -0.057008466 0.0570084657  
## 36 0.0490 0.005454535 0.0925454654  
## 37 0.0050 -0.064121736 0.0741217361  
## 38 0.0300 -0.025791885 0.0857918847  
## 39 0.0090 -0.052426619 0.0704266191  
## 40 0.0820 0.037443982 0.1265560180  
## 41 0.0000 -0.050606052 0.0506060525  
## 42 0.0500 0.011491350 0.0885086495  
## 43 0.0730 0.026563876 0.1194361238  
## 44 -0.0010 -0.067563833 0.0655638334  
## 45 -0.0040 -0.070756104 0.0627561042  
## 46 -0.0300 -0.080583275 0.0205832746  
## 47 0.0680 0.016894089 0.1191059111  
## 48 0.0500 0.011051757 0.0889482429  
## 49 0.0000 -0.050606052 0.0506060525  
## 50 0.0262 0.013635277 0.0387647229  
## 51 0.0333 -0.016106137 0.0827061365  
## 52 0.0000 -0.050606052 0.0506060525  
## 53 0.0124 -0.038560140 0.0633601401  
## 54 0.0600 0.021340150 0.0986598497  
## 55 -0.0475 -0.083637121 -0.0113628793  
## 56 0.0009 -0.051576011 0.0533760106  
## 57 0.0500 0.011807815 0.0881921851  
## 58 -0.0553 -0.091100799 -0.0194992008  
## 59 0.0056 -0.048162418 0.0593624177  
## 60 -0.0540 -0.089809343 -0.0181906570  
## 61 0.0067 -0.046002019 0.0594020190  
## 62 0.0500 0.011756829 0.0882431712  
## 63 0.0100 -0.016769729 0.0367697294  
## 64 -0.0351 -0.071090499 0.0008904993  
## 65 0.0300 -0.008295761 0.0682957614  
## 66 -0.0503 -0.086413709 -0.0141862908  
## 67 0.0200 -0.019711083 0.0597110831  
## 68 -0.0547 -0.090406512 -0.0189934879  
## 69 0.0200 -0.018430368 0.0584303678  
## 70 -0.0362 -0.071126335 -0.0012736645

1. What proportion of confidence intervals for the difference between the proportion of voters included , the actual difference in election day?

# The `pollster\_results` object has already been loaded. Examine it using the `head` function.  
head(pollster\_results)

## pollster enddate d\_hat  
## 1 ABC News/Washington Post 2016-11-06 0.0400  
## 2 Google Consumer Surveys 2016-11-07 0.0234  
## 3 Ipsos 2016-11-06 0.0300  
## 4 YouGov 2016-11-07 0.0400  
## 5 Gravis Marketing 2016-11-06 0.0400  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research 2016-11-06 0.0400  
## lower upper  
## 1 -0.001564627 0.08156463  
## 2 0.011380104 0.03541990  
## 3 -0.011815309 0.07181531  
## 4 0.007703641 0.07229636  
## 5 0.024817728 0.05518227  
## 6 -0.014420872 0.09442087

# Add a logical variable called `hit` that indicates whether the actual value (0.021) exists within the confidence interval of each poll. Summarize the average `hit` result to determine the proportion of polls with confidence intervals include the actual value. Save the result as an object called `avg\_hit`.  
avg\_hit <- pollster\_results %>% mutate(hit=(lower<0.021 & upper>0.021)) %>% summarize(mean(hit))  
avg\_hit

## mean(hit)  
## 1 0.7714286

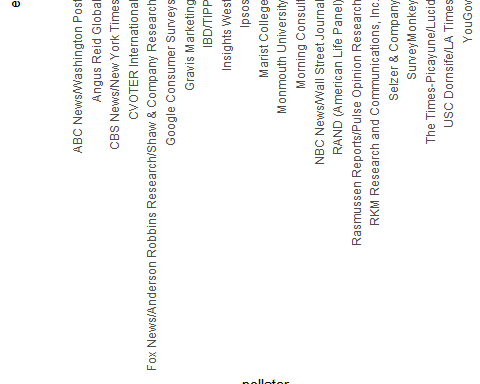
1. Although the proportion of confidence intervals that include the actual difference between the proportion of voters increases substantially, it is still lower that 0.95.

In the next chapter, we learn the reason for this. To motivate our next exercises, calculate the difference between each poll’s estimate and the actual . Stratify this difference, or error, by pollster in a plot.

# The `polls` object has already been loaded. Examine it using the `head` function.  
head(polls)

## state startdate enddate  
## 1 U.S. 2016-11-03 2016-11-06  
## 2 U.S. 2016-11-01 2016-11-07  
## 3 U.S. 2016-11-02 2016-11-06  
## 4 U.S. 2016-11-04 2016-11-07  
## 5 U.S. 2016-11-03 2016-11-06  
## 6 U.S. 2016-11-03 2016-11-06  
## pollster grade samplesize  
## 1 ABC News/Washington Post A+ 2220  
## 2 Google Consumer Surveys B 26574  
## 3 Ipsos A- 2195  
## 4 YouGov B 3677  
## 5 Gravis Marketing B- 16639  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research A 1295  
## population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson rawpoll\_mcmullin  
## 1 lv 47.00 43.00 4.00 NA  
## 2 lv 38.03 35.69 5.46 NA  
## 3 lv 42.00 39.00 6.00 NA  
## 4 lv 45.00 41.00 5.00 NA  
## 5 rv 47.00 43.00 3.00 NA  
## 6 lv 48.00 44.00 3.00 NA  
## adjpoll\_clinton adjpoll\_trump adjpoll\_johnson adjpoll\_mcmullin d\_hat X\_hat  
## 1 45.20163 41.72430 4.626221 NA 0.0400 0.5200  
## 2 43.34557 41.21439 5.175792 NA 0.0234 0.5117  
## 3 42.02638 38.81620 6.844734 NA 0.0300 0.5150  
## 4 45.65676 40.92004 6.069454 NA 0.0400 0.5200  
## 5 46.84089 42.33184 3.726098 NA 0.0400 0.5200  
## 6 49.02208 43.95631 3.057876 NA 0.0400 0.5200  
## se\_hat lower upper  
## 1 0.021206832 -0.001564627 0.08156463  
## 2 0.006132712 0.011380104 0.03541990  
## 3 0.021334733 -0.011815309 0.07181531  
## 4 0.016478037 0.007703641 0.07229636  
## 5 0.007746199 0.024817728 0.05518227  
## 6 0.027766261 -0.014420872 0.09442087

# Add variable called `error` to the object `polls` that contains the difference between d\_hat and the actual difference on election day. Then make a plot of the error stratified by pollster.  
error <- polls$d\_hat - 0.021  
polls <- mutate(polls, error)  
polls %>% ggplot(aes(x = pollster, y = error)) +  
 geom\_point() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



1. Remake the plot you made for the previous exercise, but only for pollsters that took five or more polls.

You can use dplyr tools group\_by and n to group data by a variable of interest and then count the number of observations in the groups. The function filter filters data piped into it by your specified condition.

For example:

data %>% group\_by(variable\_for\_grouping)   
 %>% filter(n() >= 5)

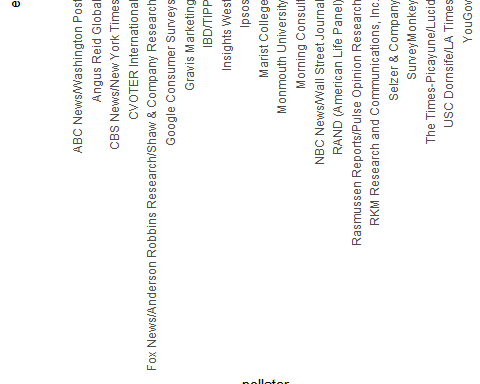
# The `polls` object has already been loaded. Examine it using the `head` function.  
head(polls)

## state startdate enddate  
## 1 U.S. 2016-11-03 2016-11-06  
## 2 U.S. 2016-11-01 2016-11-07  
## 3 U.S. 2016-11-02 2016-11-06  
## 4 U.S. 2016-11-04 2016-11-07  
## 5 U.S. 2016-11-03 2016-11-06  
## 6 U.S. 2016-11-03 2016-11-06  
## pollster grade samplesize  
## 1 ABC News/Washington Post A+ 2220  
## 2 Google Consumer Surveys B 26574  
## 3 Ipsos A- 2195  
## 4 YouGov B 3677  
## 5 Gravis Marketing B- 16639  
## 6 Fox News/Anderson Robbins Research/Shaw & Company Research A 1295  
## population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson rawpoll\_mcmullin  
## 1 lv 47.00 43.00 4.00 NA  
## 2 lv 38.03 35.69 5.46 NA  
## 3 lv 42.00 39.00 6.00 NA  
## 4 lv 45.00 41.00 5.00 NA  
## 5 rv 47.00 43.00 3.00 NA  
## 6 lv 48.00 44.00 3.00 NA  
## adjpoll\_clinton adjpoll\_trump adjpoll\_johnson adjpoll\_mcmullin d\_hat X\_hat  
## 1 45.20163 41.72430 4.626221 NA 0.0400 0.5200  
## 2 43.34557 41.21439 5.175792 NA 0.0234 0.5117  
## 3 42.02638 38.81620 6.844734 NA 0.0300 0.5150  
## 4 45.65676 40.92004 6.069454 NA 0.0400 0.5200  
## 5 46.84089 42.33184 3.726098 NA 0.0400 0.5200  
## 6 49.02208 43.95631 3.057876 NA 0.0400 0.5200  
## se\_hat lower upper error  
## 1 0.021206832 -0.001564627 0.08156463 0.0190  
## 2 0.006132712 0.011380104 0.03541990 0.0024  
## 3 0.021334733 -0.011815309 0.07181531 0.0090  
## 4 0.016478037 0.007703641 0.07229636 0.0190  
## 5 0.007746199 0.024817728 0.05518227 0.0190  
## 6 0.027766261 -0.014420872 0.09442087 0.0190

# Add variable called `error` to the object `polls` that contains the difference between d\_hat and the actual difference on election day. Then make a plot of the error stratified by pollster, but only for pollsters who took 5 or more polls.  
error <- polls$d\_hat - 0.021  
polls <- mutate(polls, error)  
polls %>% group\_by(pollster) %>% filter(n() >= 5)

## # A tibble: 48 x 21  
## # Groups: pollster [7]  
## state startdate enddate pollster grade samplesize population  
## <fct> <date> <date> <fct> <fct> <int> <chr>   
## 1 U.S. 2016-11-03 2016-11-06 ABC News/Washington ~ A+ 2220 lv   
## 2 U.S. 2016-11-02 2016-11-06 Ipsos A- 2195 lv   
## 3 U.S. 2016-11-04 2016-11-07 IBD/TIPP A- 1107 lv   
## 4 U.S. 2016-11-05 2016-11-07 The Times-Picayune/L~ <NA> 2521 lv   
## 5 U.S. 2016-11-01 2016-11-07 USC Dornsife/LA Times <NA> 2972 lv   
## 6 U.S. 2016-10-31 2016-11-06 CVOTER International C+ 1625 lv   
## 7 U.S. 2016-11-02 2016-11-06 Rasmussen Reports/Pu~ C+ 1500 lv   
## 8 U.S. 2016-11-02 2016-11-05 ABC News/Washington ~ A+ 1937 lv   
## 9 U.S. 2016-10-31 2016-11-04 Ipsos A- 2244 lv   
## 10 U.S. 2016-11-01 2016-11-04 ABC News/Washington ~ A+ 1685 lv   
## # ... with 38 more rows, and 14 more variables: rawpoll\_clinton <dbl>,  
## # rawpoll\_trump <dbl>, rawpoll\_johnson <dbl>, rawpoll\_mcmullin <dbl>,  
## # adjpoll\_clinton <dbl>, adjpoll\_trump <dbl>, adjpoll\_johnson <dbl>,  
## # adjpoll\_mcmullin <dbl>, d\_hat <dbl>, X\_hat <dbl>, se\_hat <dbl>,  
## # lower <dbl>, upper <dbl>, error <dbl>

polls %>% ggplot(aes(x = pollster, y = error)) +  
 geom\_point() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



# Section 4 Overview

In Section 4, you will look at statistical models in the context of election polling and forecasting.

After completing Section 4, you will be able to:

* Understand how aggregating data from different sources, as poll aggregators do for poll data, can improve the precision of a prediction.
* Understand how to fit a multilevel model to the data to forecast, for example, election results.
* Explain why a simple aggregation of data is insufficient to combine results because of factors such as pollster bias.
* Use a data-driven model to account for additional types of sampling variability such as pollster-to-pollster variability.

## Poll Aggregators

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html) and [here](https://rafalab.github.io/dsbook/models.html#poll-aggregators).

**Key points**

* Poll aggregators combine the results of many polls to simulate polls with a large sample size and therefore generate more precise estimates than individual polls.
* Polls can be simulated with a Monte Carlo simulation and used to construct an estimate of the spread and confidence intervals.
* The actual data science exercise of forecasting elections involves more complex statistical modeling, but these underlying ideas still apply.

*Code: Simulating polls*

Note that to compute the exact 95% confidence interval, we would use qnorm(.975)\*SE\_hat instead of 2\*SE\_hat.

d <- 0.039  
Ns <- c(1298, 533, 1342, 897, 774, 254, 812, 324, 1291, 1056, 2172, 516)  
p <- (d+1)/2  
  
# calculate confidence intervals of the spread  
confidence\_intervals <- sapply(Ns, function(N){  
 X <- sample(c(0,1), size=N, replace=TRUE, prob = c(1-p, p))  
 X\_hat <- mean(X)  
 SE\_hat <- sqrt(X\_hat\*(1-X\_hat)/N)  
 2\*c(X\_hat, X\_hat - 2\*SE\_hat, X\_hat + 2\*SE\_hat) - 1  
})  
  
# generate a data frame storing results  
polls <- data.frame(poll = 1:ncol(confidence\_intervals),  
 t(confidence\_intervals), sample\_size = Ns)  
names(polls) <- c("poll", "estimate", "low", "high", "sample\_size")  
polls

## poll estimate low high sample\_size  
## 1 1 0.020030817 -0.0354707836 0.07553242 1298  
## 2 2 0.009380863 -0.0772449415 0.09600667 533  
## 3 3 0.005961252 -0.0486328871 0.06055539 1342  
## 4 4 0.030100334 -0.0366474636 0.09684813 897  
## 5 5 0.085271318 0.0136446370 0.15689800 774  
## 6 6 0.070866142 -0.0543095137 0.19604180 254  
## 7 7 0.029556650 -0.0405989265 0.09971223 812  
## 8 8 0.086419753 -0.0242756708 0.19711518 324  
## 9 9 0.027110767 -0.0285318074 0.08275334 1291  
## 10 10 0.022727273 -0.0388025756 0.08425712 1056  
## 11 11 0.042357274 -0.0005183189 0.08523287 2172  
## 12 12 0.112403101 0.0249159791 0.19989022 516

*Code: Calculating the spread of combined polls*

Note that to compute the exact 95% confidence interval, we would use qnorm(.975) instead of 1.96.

d\_hat <- polls %>%  
 summarize(avg = sum(estimate\*sample\_size) / sum(sample\_size)) %>%  
 .$avg  
  
p\_hat <- (1+d\_hat)/2  
moe <- 2\*1.96\*sqrt(p\_hat\*(1-p\_hat)/sum(polls$sample\_size))   
round(d\_hat\*100,1)

## [1] 3.6

round(moe\*100, 1)

## [1] 1.8

## Pollsters and Multilevel Models

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#poll-aggregators).

**Key points**

* Different poll aggregators generate different models of election results from the same poll data. This is because they use different statistical models.
* We will use actual polling data about the popular vote from the 2016 US presidential election to learn the principles of statistical modeling.

## Poll Data and Pollster Bias

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#poll-data) and [here](https://rafalab.github.io/dsbook/models.html#pollster-bias).

**Key points**

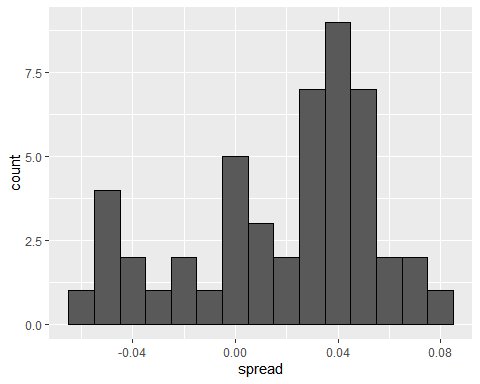
* We analyze real 2016 US polling data organized by FiveThirtyEight. We start by using reliable national polls taken within the week before the election to generate an urn model.
* Consider the proportion voting for Clinton and the proportion voting for Trump. We are interested in the spread .
* Poll results are a random normal variable with expected value of the spread and standard error .
* Our initial estimate of the spread did not include the actual spread. Part of the reason is that different pollsters have different numbers of polls in our dataset, and each pollster has a bias.
* *Pollster bias* reflects the fact that repeated polls by a given pollster have an expected value different from the actual spread and different from other pollsters. Each pollster has a different bias.
* The urn model does not account for pollster bias. We will develop a more flexible data-driven model that can account for effects like bias.

*Code: Generating simulated poll data*

names(polls\_us\_election\_2016)

## [1] "state" "startdate" "enddate" "pollster"   
## [5] "grade" "samplesize" "population" "rawpoll\_clinton"   
## [9] "rawpoll\_trump" "rawpoll\_johnson" "rawpoll\_mcmullin" "adjpoll\_clinton"   
## [13] "adjpoll\_trump" "adjpoll\_johnson" "adjpoll\_mcmullin"

# keep only national polls from week before election with a grade considered reliable  
polls <- polls\_us\_election\_2016 %>%  
 filter(state == "U.S." & enddate >= "2016-10-31" &  
 (grade %in% c("A+", "A", "A-", "B+") | is.na(grade)))  
  
# add spread estimate  
polls <- polls %>%  
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
# compute estimated spread for combined polls  
d\_hat <- polls %>%  
 summarize(d\_hat = sum(spread \* samplesize) / sum(samplesize)) %>%  
 .$d\_hat  
  
# compute margin of error  
p\_hat <- (d\_hat+1)/2  
moe <- 1.96 \* 2 \* sqrt(p\_hat\*(1-p\_hat)/sum(polls$samplesize))  
  
# histogram of the spread  
polls %>%  
 ggplot(aes(spread)) +  
 geom\_histogram(color="black", binwidth = .01)

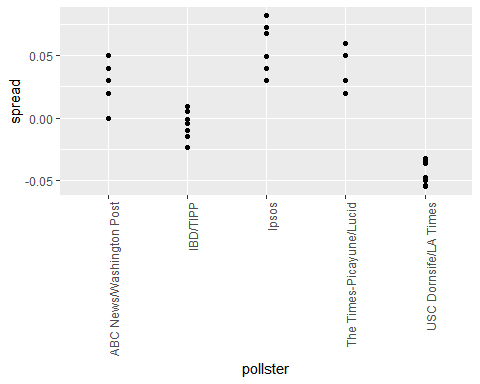


*Code: Investigating poll data and pollster bias*

# number of polls per pollster in week before election  
polls %>% group\_by(pollster) %>% summarize(n())

## # A tibble: 15 x 2  
## pollster `n()`  
## <fct> <int>  
## 1 ABC News/Washington Post 7  
## 2 Angus Reid Global 1  
## 3 CBS News/New York Times 2  
## 4 Fox News/Anderson Robbins Research/Shaw & Company Research 2  
## 5 IBD/TIPP 8  
## 6 Insights West 1  
## 7 Ipsos 6  
## 8 Marist College 1  
## 9 Monmouth University 1  
## 10 Morning Consult 1  
## 11 NBC News/Wall Street Journal 1  
## 12 RKM Research and Communications, Inc. 1  
## 13 Selzer & Company 1  
## 14 The Times-Picayune/Lucid 8  
## 15 USC Dornsife/LA Times 8

# plot results by pollsters with at least 6 polls  
polls %>% group\_by(pollster) %>%  
 filter(n() >= 6) %>%  
 ggplot(aes(pollster, spread)) +  
 geom\_point() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



# standard errors within each pollster  
polls %>% group\_by(pollster) %>%  
 filter(n() >= 6) %>%  
 summarize(se = 2 \* sqrt(p\_hat \* (1-p\_hat) / median(samplesize)))

## # A tibble: 5 x 2  
## pollster se  
## <fct> <dbl>  
## 1 ABC News/Washington Post 0.0265  
## 2 IBD/TIPP 0.0333  
## 3 Ipsos 0.0225  
## 4 The Times-Picayune/Lucid 0.0196  
## 5 USC Dornsife/LA Times 0.0183

## Data-Driven Models

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#data-driven-model).

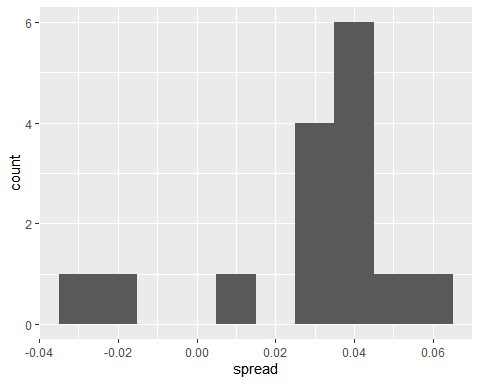
**Key points**

* Instead of using an urn model where each poll is a random draw from the same distribution of voters, we instead define a model using an urn that contains poll results from all possible pollsters.
* We assume the expected value of this model is the actual spread .
* Our new standard error now factors in pollster-to-pollster variability. It can no longer be calculated from or and is an unknown parameter.
* The central limit theorem still works to estimate the sample average of many polls because the average of the sum of many random variables is a normally distributed random variable with expected value and standard error .
* We can estimate the unobserved as the sample standard deviation, which is calculated with the sd function.

*Code*

Note that to compute the exact 95% confidence interval, we would use qnorm(.975) instead of 1.96.

# collect last result before the election for each pollster  
one\_poll\_per\_pollster <- polls %>% group\_by(pollster) %>%  
 filter(enddate == max(enddate)) %>% # keep latest poll  
 ungroup()  
  
# histogram of spread estimates  
one\_poll\_per\_pollster %>%  
 ggplot(aes(spread)) + geom\_histogram(binwidth = 0.01)



# construct 95% confidence interval  
results <- one\_poll\_per\_pollster %>%  
 summarize(avg = mean(spread), se = sd(spread)/sqrt(length(spread))) %>%  
 mutate(start = avg - 1.96\*se, end = avg + 1.96\*se)  
round(results\*100, 1)

## avg se start end  
## 1 2.9 0.6 1.7 4.1

## Assessment - Statistical Models

1. We have been using *urn models* to motivate the use of probability models.

However, most data science applications are not related to data obtained from urns. More common are data that come from individuals. Probability plays a role because the data come from a random sample. The random sample is taken from a population and the urn serves as an analogy for the population.

Let’s revisit the heights dataset. For now, consider x to be the heights of all males in the data set. Mathematically speaking, x is our population. Using the urn analogy, we have an urn with the values of x in it.

What are the population average and standard deviation of our population?

data(heights)  
  
# Make a vector of heights from all males in the population  
x <- heights %>% filter(sex == "Male") %>%  
 .$height  
  
# Calculate the population average. Print this value to the console.  
gemiddelde\_lengte <- mean(x)  
gemiddelde\_lengte

## [1] 69.31475

# Calculate the population standard deviation. Print this value to the console.  
standaard\_deviatie <- sd(x)  
standaard\_deviatie

## [1] 3.611024

1. Call the population average computed above and the standard deviation . Now take a sample of size 50, with replacement, and construct an estimate for and .

# The vector of all male heights in our population `x` has already been loaded for you. You can examine the first six elements using `head`.  
head(x)

## [1] 75 70 68 74 61 67

# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Define `N` as the number of people measured  
N <- 50  
  
# Define `X` as a random sample from our population `x`  
X <- sample(x, N, replace = TRUE)  
X

## [1] 71.00000 69.00000 70.00000 66.92913 68.50000 79.05000 72.00000 72.00000  
## [9] 69.00000 66.00000 69.00000 70.86614 72.00000 66.00000 70.00000 73.00000  
## [17] 69.00000 76.00000 72.44000 72.44000 74.00000 74.00000 74.00000 63.00000  
## [25] 69.00000 70.86614 68.50394 72.00000 70.00000 68.00000 70.00000 72.00000  
## [33] 68.00000 67.00000 76.00000 70.00000 73.00000 68.00000 69.00000 77.16540  
## [41] 72.00000 66.00000 67.50000 68.00000 72.00000 63.38583 73.00000 78.00000  
## [49] 68.00000 68.00000

# Calculate the sample average. Print this value to the console.  
mu <- mean(X)  
mu

## [1] 70.47293

# Calculate the sample standard deviation. Print this value to the console.  
sigma <- sd(X)  
sigma

## [1] 3.426742

1. What does the central limit theory tell us about the sample average and how it is related to , the population average?

* ☐ A. It is identical to .
* ☒ B. It is a random variable with expected value and standard error .
* ☐ C. It is a random variable with expected value and standard error .
* ☐ D. It underestimates .

1. We will use as our estimate of the heights in the population from our sample size . We know from previous exercises that the standard estimate of our error is .

Construct a 95% confidence interval for .

# The vector of all male heights in our population `x` has already been loaded for you. You can examine the first six elements using `head`.  
head(x)

## [1] 75 70 68 74 61 67

# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Define `N` as the number of people measured  
N <- 50  
  
# Define `X` as a random sample from our population `x`  
X <- sample(x, N, replace = TRUE)  
  
# Define `se` as the standard error of the estimate. Print this value to the console.  
X\_hat <- mean(X)  
se <- sd(X)/sqrt(N)  
se

## [1] 0.4846145

# Construct a 95% confidence interval for the population average based on our sample. Save the lower and then the upper confidence interval to a variable called `ci`.  
# Construct a 95% confidence interval for the population average based on our sample. Save the lower and then the upper confidence interval to a variable called `ci`.  
ci <- c(qnorm(0.025, X\_hat, se), qnorm(0.975, X\_hat, se))  
ci

## [1] 69.52310 71.42276

1. Now run a Monte Carlo simulation in which you compute 10,000 confidence intervals as you have just done.

What proportion of these intervals include ?

# Define `mu` as the population average  
mu <- mean(x)  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Define `N` as the number of people measured  
N <- 50  
  
# Define `B` as the number of times to run the model  
B <- 10000  
  
# Define an object `res` that contains a logical vector for simulated intervals that contain mu  
res <- replicate(B, {  
 X <- sample(x, N, replace = TRUE)  
 se <- sd(X) / sqrt(N)  
 interval <- c(qnorm(0.025, mean(X), se) , qnorm(0.975, mean(X), se))  
 between(mu, interval[1], interval[2])  
})  
  
# Calculate the proportion of results in `res` that include mu. Print this value to the console.  
mean(res)

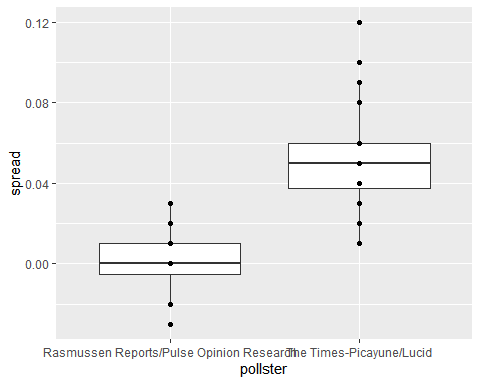
## [1] 0.9479

1. In this section, we used visualization to motivate the presence of pollster bias in election polls.

Here we will examine that bias more rigorously. Lets consider two pollsters that conducted daily polls and look at national polls for the month before the election.

Is there a poll bias? Make a plot of the spreads for each poll.

# These lines of code filter for the polls we want and calculate the spreads  
polls <- polls\_us\_election\_2016 %>%   
 filter(pollster %in% c("Rasmussen Reports/Pulse Opinion Research","The Times-Picayune/Lucid") &  
 enddate >= "2016-10-15" &  
 state == "U.S.") %>%   
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)   
  
# Make a boxplot with points of the spread for each pollster  
polls %>% ggplot(aes(pollster, spread)) + geom\_boxplot() + geom\_point()



1. The data do seem to suggest there is a difference between the pollsters.

However, these data are subject to variability. Perhaps the differences we observe are due to chance. Under the urn model, both pollsters should have the same expected value: the election day difference, .

We will model the observed data in the following way:

with indexing the two pollsters, the bias for pollster , and poll to poll chance variability. We assume the are independent from each other, have expected value 0 and standard deviation regardless of .

Which of the following statements best reflects what we need to know to determine if our data fit the urn model?

* ☐ A. Is ?
* ☐ B. How close are to ?
* ☒ C. Is ?
* ☐ D. Are and ?

1. We modelled the observed data as:

On the right side of this model, only is a random variable. The other two values are constants.

What is the expected value of ?

* ☒ A.
* ☐ B.
* ☐ C.
* ☐ D.

1. Suppose we define as the average of poll results from the first poll and as the standard deviation of the first poll.

What is the expected value and standard error of ?

* ☐ A. The expected value is and the standard error is
* ☐ B. The expected value is and the standard error is
* ☒ C. The expected value is and the standard error is
* ☐ D. The expected value is and the standard error is

1. Now we define as the average of poll results from the second poll.

What is the expected value and standard error of ?

* ☐ A. The expected value is and the standard error is
* ☐ B. The expected value is and the standard error is
* ☒ C. The expected value is and the standard error is
* ☐ D. The expected value is and the standard error is

1. Using what we learned by answering the previous questions, what is the expected value of ?

* ☐ A.
* ☐ B.
* ☐ C.
* ☒ D.

1. Standard Error of the Difference Between Polls

Using what we learned by answering the questions above, what is the standard error of ?

* ☒ A.
* ☐ B.
* ☐ C.
* ☐ D.

1. The answer to the previous question depends on and , which we don’t know.

We learned that we can estimate these values using the sample standard deviation.

Compute the estimates of and .

# The `polls` data have already been loaded for you. Use the `head` function to examine them.  
head(polls)

## state startdate enddate pollster grade  
## 1 U.S. 2016-11-05 2016-11-07 The Times-Picayune/Lucid <NA>  
## 2 U.S. 2016-11-02 2016-11-06 Rasmussen Reports/Pulse Opinion Research C+  
## 3 U.S. 2016-11-01 2016-11-03 Rasmussen Reports/Pulse Opinion Research C+  
## 4 U.S. 2016-11-04 2016-11-06 The Times-Picayune/Lucid <NA>  
## 5 U.S. 2016-10-31 2016-11-02 Rasmussen Reports/Pulse Opinion Research C+  
## 6 U.S. 2016-11-03 2016-11-05 The Times-Picayune/Lucid <NA>  
## samplesize population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson  
## 1 2521 lv 45 40 5  
## 2 1500 lv 45 43 4  
## 3 1500 lv 44 44 4  
## 4 2584 lv 45 40 5  
## 5 1500 lv 42 45 4  
## 6 2526 lv 45 40 5  
## rawpoll\_mcmullin adjpoll\_clinton adjpoll\_trump adjpoll\_johnson  
## 1 NA 45.13966 42.26495 3.679914  
## 2 NA 45.56041 43.13745 4.418502  
## 3 NA 44.66353 44.28981 4.331246  
## 4 NA 45.22830 42.30433 3.770880  
## 5 NA 42.67626 45.41689 4.239500  
## 6 NA 45.31041 42.34422 3.779955  
## adjpoll\_mcmullin spread  
## 1 NA 0.05  
## 2 NA 0.02  
## 3 NA 0.00  
## 4 NA 0.05  
## 5 NA -0.03  
## 6 NA 0.05

# Create an object called `sigma` that contains a column for `pollster` and a column for `s`, the standard deviation of the spread  
sigma <- polls %>% group\_by(pollster) %>% summarize(s = sd(spread))  
  
# Print the contents of sigma to the console  
sigma

## # A tibble: 2 x 2  
## pollster s  
## <fct> <dbl>  
## 1 Rasmussen Reports/Pulse Opinion Research 0.0177  
## 2 The Times-Picayune/Lucid 0.0268

1. What does the central limit theorem tell us about the distribution of the differences between the pollster averages, ?

* ☐ A. The central limit theorem cannot tell us anything because this difference is not the average of a sample.
* ☐ B. Because are approximately normal, the averages are normal too.
* ☒ C. If we assume and are large enough, and , and their difference, are approximately normal.
* ☐ D. These data do not contain vectors of 0 and 1, so the central limit theorem does not apply.

1. We have constructed a random variable that has expected value , the pollster bias difference.

If our model holds, then this random variable has an approximately normal distribution. The standard error of this random variable depends on and , but we can use the sample standard deviations we computed earlier. We have everything we need to answer our initial question: is different from 0?

Construct a 95% confidence interval for the difference and . Does this interval contain zero?

# The `polls` data have already been loaded for you. Use the `head` function to examine them.  
head(polls)

## state startdate enddate pollster grade  
## 1 U.S. 2016-11-05 2016-11-07 The Times-Picayune/Lucid <NA>  
## 2 U.S. 2016-11-02 2016-11-06 Rasmussen Reports/Pulse Opinion Research C+  
## 3 U.S. 2016-11-01 2016-11-03 Rasmussen Reports/Pulse Opinion Research C+  
## 4 U.S. 2016-11-04 2016-11-06 The Times-Picayune/Lucid <NA>  
## 5 U.S. 2016-10-31 2016-11-02 Rasmussen Reports/Pulse Opinion Research C+  
## 6 U.S. 2016-11-03 2016-11-05 The Times-Picayune/Lucid <NA>  
## samplesize population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson  
## 1 2521 lv 45 40 5  
## 2 1500 lv 45 43 4  
## 3 1500 lv 44 44 4  
## 4 2584 lv 45 40 5  
## 5 1500 lv 42 45 4  
## 6 2526 lv 45 40 5  
## rawpoll\_mcmullin adjpoll\_clinton adjpoll\_trump adjpoll\_johnson  
## 1 NA 45.13966 42.26495 3.679914  
## 2 NA 45.56041 43.13745 4.418502  
## 3 NA 44.66353 44.28981 4.331246  
## 4 NA 45.22830 42.30433 3.770880  
## 5 NA 42.67626 45.41689 4.239500  
## 6 NA 45.31041 42.34422 3.779955  
## adjpoll\_mcmullin spread  
## 1 NA 0.05  
## 2 NA 0.02  
## 3 NA 0.00  
## 4 NA 0.05  
## 5 NA -0.03  
## 6 NA 0.05

# Create an object called `res` that summarizes the average, standard deviation, and number of polls for the two pollsters.  
res <- polls %>% group\_by(pollster) %>% summarise(avg = mean(spread), s = sd(spread), N=n())  
  
# Store the difference between the larger average and the smaller in a variable called `estimate`. Print this value to the console.  
estimate <- max(res$avg) - min(res$avg)  
estimate

## [1] 0.05229167

# Store the standard error of the estimates as a variable called `se\_hat`. Print this value to the console.  
se\_hat <- sqrt(res$s[2]^2/res$N[2] + res$s[1]^2/res$N[1])  
se\_hat

## [1] 0.007031433

# Calculate the 95% confidence interval of the spreads. Save the lower and then the upper confidence interval to a variable called `ci`.  
ci <- c(estimate - qnorm(0.975)\*se\_hat, estimate + qnorm(0.975)\*se\_hat)  
ci

## [1] 0.03851031 0.06607302

1. The confidence interval tells us there is relatively strong pollster effect resulting in a difference of about 5%.

Random variability does not seem to explain it.

Compute a p-value to relay the fact that chance does not explain the observed pollster effect.

# We made an object `res` to summarize the average, standard deviation, and number of polls for the two pollsters.  
res <- polls %>% group\_by(pollster) %>%   
 summarize(avg = mean(spread), s = sd(spread), N = n())   
  
# The variables `estimate` and `se\_hat` contain the spread estimates and standard error, respectively.  
estimate <- res$avg[2] - res$avg[1]  
se\_hat <- sqrt(res$s[2]^2/res$N[2] + res$s[1]^2/res$N[1])  
  
# Calculate the p-value  
2 \* (1 - pnorm(estimate / se\_hat, 0, 1))

## [1] 1.030287e-13

1. We compute statistic called the *t-statistic* by dividing our estimate of by its estimated standard error:

Later we learn will learn of another approximation for the distribution of this statistic for values of and that aren’t large enough for the CLT.

Note that our data has more than two pollsters. We can also test for pollster effect using all pollsters, not just two. The idea is to compare the variability across polls to variability within polls. We can construct statistics to test for effects and approximate their distribution. The area of statistics that does this is called Analysis of Variance or ANOVA. We do not cover it here, but ANOVA provides a very useful set of tools to answer questions such as: is there a pollster effect?

Compute the average and standard deviation for each pollster and examine the variability across the averages and how it compares to the variability within the pollsters, summarized by the standard deviation.

# Execute the following lines of code to filter the polling data and calculate the spread  
polls <- polls\_us\_election\_2016 %>%   
 filter(enddate >= "2016-10-15" &  
 state == "U.S.") %>%  
 group\_by(pollster) %>%  
 filter(n() >= 5) %>%   
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100) %>%  
 ungroup()  
  
# Create an object called `var` that contains columns for the pollster, mean spread, and standard deviation. Print the contents of this object to the console.  
var <- polls %>% group\_by(pollster) %>% summarise(avg = mean(spread), s = sd(spread))  
var

## # A tibble: 11 x 3  
## pollster avg s  
## <fct> <dbl> <dbl>  
## 1 ABC News/Washington Post 0.0373 0.0339   
## 2 CVOTER International 0.0279 0.0180   
## 3 Google Consumer Surveys 0.0303 0.0185   
## 4 Gravis Marketing 0.0160 0.0152   
## 5 IBD/TIPP 0.00105 0.0168   
## 6 Ipsos 0.0553 0.0195   
## 7 Morning Consult 0.0414 0.0146   
## 8 Rasmussen Reports/Pulse Opinion Research 0.000625 0.0177   
## 9 The Times-Picayune/Lucid 0.0529 0.0268   
## 10 USC Dornsife/LA Times -0.0213 0.0207   
## 11 YouGov 0.0398 0.00709

# Section 5 Overview

In Section 5, you will learn about Bayesian statistics through looking at examples from rare disease diagnosis and baseball.

After completing Section 5, you will be able to:

* Apply Bayes’ theorem to calculate the probability of A given B.
* Understand how to use hierarchical models to make better predictions by considering multiple levels of variability.
* Compute a posterior probability using an empirical Bayesian approach.
* Calculate a 95% credible interval from a posterior probability.

## Bayesian Statistics

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#bayesian-statistics).

**Key points**

* In the urn model, it does not make sense to talk about the probability of being greater than a certain value because is a fixed value.
* With Bayesian statistics, we assume that is in fact random, which allows us to calculate probabilities related to .
* Hierarchical models describe variability at different levels and incorporate all these levels into a model for estimating .

## Bayes’ Theorem

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#bayes-theorem).

**Key points**

* Bayes’ Theorem states that the probability of event A happening given event B is equal to the probability of both A and B divided by the probability of event B:
* Bayes’ Theorem shows that a test for a very rare disease will have a high percentage of false positives even if the accuracy of the test is high.

*Equations: Cystic fibrosis test probabilities*

In these probabilities, + represents a positive test, - represents a negative test, indicates no disease, and indicates the disease is present.

Probability of having the disease given a positive test:

99% test accuracy when disease is present:

99% test accuracy when disease is absent:

Rate of cystic fibrosis:

Bayes’ theorem can be applied like this:

Substituting known values, we obtain:

*Code: Monte Carlo simulation*

prev <- 0.00025 # disease prevalence  
N <- 100000 # number of tests  
outcome <- sample(c("Disease", "Healthy"), N, replace = TRUE, prob = c(prev, 1-prev))  
  
N\_D <- sum(outcome == "Disease") # number with disease  
N\_H <- sum(outcome == "Healthy") # number healthy  
  
# for each person, randomly determine if test is + or -  
accuracy <- 0.99  
test <- vector("character", N)  
test[outcome == "Disease"] <- sample(c("+", "-"), N\_D, replace=TRUE, prob = c(accuracy, 1-accuracy))  
test[outcome == "Healthy"] <- sample(c("-", "+"), N\_H, replace=TRUE, prob = c(accuracy, 1-accuracy))  
  
table(outcome, test)

## test  
## outcome - +  
## Disease 0 27  
## Healthy 99013 960

## Bayes in Practice

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#bayes-in-practice).

**Key points**

* The techniques we have used up until now are referred to as *frequentist statistics* as they consider only the frequency of outcomes in a dataset and do not include any outside information. Frequentist statistics allow us to compute confidence intervals and p-values.
* Frequentist statistics can have problems when sample sizes are small and when the data are extreme compared to historical results.
* *Bayesian statistics* allows prior knowledge to modify observed results, which alters our conclusions about event probabilities.

## The Hierarchical Model

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#hierarchical-models).

**Key points**

* Hierarchical models use multiple levels of variability to model results. They are hierarchical because values in the lower levels of the model are computed using values from higher levels of the model.
* We model baseball player batting average using a hierarchical model with two levels of variability:
  + describes player-to-player variability in natural ability to hit, which has a mean and standard deviation .
  + describes a player’s observed batting average given their ability , which has a mean and standard deviation . This represents variability due to luck.
  + In Bayesian hierarchical models, the first level is called the *prior distribution* and the second level is called the *sampling distribution*.
* The *posterior distribution* allows us to compute the probability distribution of given that we have observed data .
* By the continuous version of Bayes’ rule, the *expected value of the posterior distribution* given is a weighted average between the prior mean and the observed data :

where

* The *standard error of the posterior distribution* is . Note that you will need to take the square root of both sides to solve for the standard error.
* This Bayesian approach is also known as *shrinking*. When is large, is close to 1 and our prediction of shrinks towards the mean (). When is small, is close to 0 and our prediction of is more weighted towards the observed data .

## Assessment - Bayesian Statistics

1. In 1999 in England [Sally Clark](https://en.wikipedia.org/wiki/Sally_Clark) was found guilty of the murder of two of her sons.

Both infants were found dead in the morning, one in 1996 and another in 1998, and she claimed the cause of death was sudden infant death syndrome (SIDS). No evidence of physical harm was found on the two infants so the main piece of evidence against her was the testimony of Professor Sir Roy Meadow, who testified that the chances of two infants dying of SIDS was 1 in 73 million. He arrived at this figure by finding that the rate of SIDS was 1 in 8,500 and then calculating that the chance of two SIDS cases was .

Based on what we’ve learned throughout this course, which statement best describes a potential flaw in Sir Meadow’s reasoning?

* ☒ A. Sir Meadow assumed the second death was independent of the first son being affected, thereby ignoring possible genetic causes.
* ☐ B. There is no flaw. The multiplicative rule always applies in this way:
* ☐ C. Sir Meadow should have added the probabilities:
* ☐ D. The rate of SIDS is too low to perform these types of statistics.

1. Let’s assume that there is in fact a genetic component to SIDS and the the probability of , is much higher than 1 in 8,500.

What is the probability of both of Sally Clark’s sons dying of SIDS?

# Define `Pr\_1` as the probability of the first son dying of SIDS  
Pr\_1 <- 1/8500  
  
# Define `Pr\_2` as the probability of the second son dying of SIDS  
Pr\_2 <- 1/100  
  
# Calculate the probability of both sons dying of SIDS. Print this value to the console.  
Pr\_1 \* Pr\_2

## [1] 1.176471e-06

1. Many press reports stated that the expert claimed the probability of Sally Clark being innocent as 1 in 73 million.

Perhaps the jury and judge also interpreted the testimony this way. This probability can be written like this:

Bayes’ rule tells us this probability is equal to:

* ☐ A.
* ☐ B.
* ☒ C.
* ☐ D. 1/8500

1. Assume that the probability of a murderer finding a way to kill her two children without leaving evidence of physical harm is:

Assume that the murder rate among mothers is 1 in 1,000,000.

According to Bayes’ rule, what is the probability of:

# Define `Pr\_1` as the probability of the first son dying of SIDS  
Pr\_1 <- 1/8500  
  
# Define `Pr\_2` as the probability of the second son dying of SIDS  
Pr\_2 <- 1/100  
  
# Define `Pr\_B` as the probability of both sons dying of SIDS  
Pr\_B <- Pr\_1\*Pr\_2  
  
# Define Pr\_A as the rate of mothers that are murderers  
Pr\_A <- 1/1000000  
  
# Define Pr\_BA as the probability that two children die without evidence of harm, given that their mother is a murderer  
Pr\_BA <- 0.50  
  
# Define Pr\_AB as the probability that a mother is a murderer, given that her two children died with no evidence of physical harm. Print this value to the console.  
Pr\_AB <- (Pr\_BA \* Pr\_A)/Pr\_B  
Pr\_AB

## [1] 0.425

1. After Sally Clark was found guilty, the Royal Statistical Society issued a statement saying that there was “no statistical basis” for the expert’s claim.

They expressed concern at the “misuse of statistics in the courts”. Eventually, Sally Clark was acquitted in June 2003.

In addition to misusing the multiplicative rule as we saw earlier, what else did Sir Meadow miss?

* ☐ A. He made an arithmetic error in forgetting to divide by the rate of SIDS in siblings.
* ☒ B. He did not take into account how rare it is for a mother to murder her children.
* ☐ C. He mixed up the numerator and denominator of Bayes’ rule.
* ☐ D. He did not take into account murder rates in the population.

1. Florida is one of the most closely watched states in the U.S. election because it has many electoral votes and the election is generally close.

Create a table with the poll spread results from Florida taken during the last days before the election using the sample code.

The CLT tells us that the average of these spreads is approximately normal. Calculate a spread average and provide an estimate of the standard error.

# Create an object `polls` that contains the spread of predictions for each candidate in Florida during the last polling days  
polls <- polls\_us\_election\_2016 %>%   
 filter(state == "Florida" & enddate >= "2016-11-04" ) %>%   
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
# Examine the `polls` object using the `head` function  
head(polls)

## state startdate enddate pollster grade  
## 1 Florida 2016-11-03 2016-11-06 Quinnipiac University A-  
## 2 Florida 2016-11-02 2016-11-04 YouGov B  
## 3 Florida 2016-11-01 2016-11-07 SurveyMonkey C-  
## 4 Florida 2016-11-06 2016-11-06 Trafalgar Group C  
## 5 Florida 2016-11-05 2016-11-06 Opinion Savvy/InsiderAdvantage C-  
## 6 Florida 2016-11-02 2016-11-06 Rasmussen Reports/Pulse Opinion Research C+  
## samplesize population rawpoll\_clinton rawpoll\_trump rawpoll\_johnson  
## 1 884 lv 46.00 45.00 2.00  
## 2 1188 rv 45.00 45.00 NA  
## 3 4092 lv 47.00 45.00 4.00  
## 4 1100 lv 46.13 49.72 2.43  
## 5 843 lv 48.40 46.40 3.00  
## 6 525 lv 47.00 45.00 2.00  
## rawpoll\_mcmullin adjpoll\_clinton adjpoll\_trump adjpoll\_johnson  
## 1 NA 46.44315 43.93999 2.098310  
## 2 NA 47.07455 46.99468 NA  
## 3 NA 45.59190 44.32744 1.692430  
## 4 NA 45.75904 46.82230 3.495849  
## 5 NA 47.37976 45.68637 2.721857  
## 6 NA 47.55885 45.13673 2.418502  
## adjpoll\_mcmullin spread  
## 1 NA 0.0100  
## 2 NA 0.0000  
## 3 NA 0.0200  
## 4 NA -0.0359  
## 5 NA 0.0200  
## 6 NA 0.0200

# Create an object called `results` that has two columns containing the average spread (`avg`) and the standard error (`se`). Print the results to the console.  
results <- polls %>% summarize(avg = mean(spread), se = sd(spread)/sqrt(n()))  
results

## avg se  
## 1 0.004154545 0.007218692

1. Assume a Bayesian model sets the prior distribution for Florida’s election night spread to be normal with expected value and standard deviation .

What are the interpretations of and ?

* ☐ A. and are arbitrary numbers that let us make probability statements about .
* ☒ B. and summarize what we would predict for Florida before seeing any polls.
* ☐ C. and summarize what we want to be true. We therefore set at 0.10 and at 0.01.
* ☐ D. The choice of prior has no effect on the Bayesian analysis.

1. The CLT tells us that our estimate of the spread has a normal distribution with expected value and standard deviation , which we calculated in a previous exercise.

Use the formulas for the posterior distribution to calculate the expected value of the posterior distribution if we set and .

# The results` object has already been loaded. Examine the values stored: `avg` and `se` of the spread  
results

## avg se  
## 1 0.004154545 0.007218692

# Define `mu` and `tau`  
mu <- 0  
tau <- 0.01  
  
# Define a variable called `sigma` that contains the standard error in the object `results`  
sigma <- results$se  
  
# Define a variable called `Y` that contains the average in the object `results`  
Y <- results$avg  
  
# Define a variable `B` using `sigma` and `tau`. Print this value to the console.  
B <- sigma^2/(sigma^2+tau^2)  
  
# Calculate the expected value of the posterior distribution  
EV <- B\*mu + (1-B)\*Y  
EV

## [1] 0.002731286

1. Compute the standard error of the posterior distribution.

# Here are the variables we have defined  
mu <- 0  
tau <- 0.01  
sigma <- results$se  
Y <- results$avg  
B <- sigma^2 / (sigma^2 + tau^2)  
  
# Compute the standard error of the posterior distribution. Print this value to the console.  
SE <- sqrt(1/((1/sigma^2)+(1/tau^2)))  
SE

## [1] 0.005853024

1. Using the fact that the posterior distribution is normal, create an interval that has a 95% of occurring centered at the posterior expected value.

Note that we call these credible intervals.

# Here are the variables we have defined in previous exercises  
mu <- 0  
tau <- 0.01  
sigma <- results$se  
Y <- results$avg  
B <- sigma^2 / (sigma^2 + tau^2)  
se <- sqrt( 1/ (1/sigma^2 + 1/tau^2))  
  
# Construct the 95% credible interval. Save the lower and then the upper confidence interval to a variable called `ci`.  
ev <- B\*mu + (1-B)\*Y  
ci <- c(ev - qnorm(0.975) \* se, ev + qnorm(0.975) \* se)  
ci

## [1] -0.008740432 0.014203003

1. According to this analysis, what was the probability that Trump wins Florida?

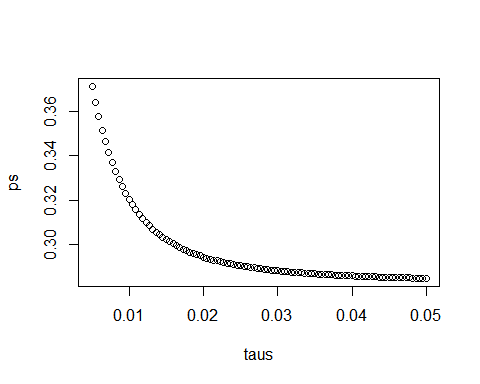
# Assign the expected value of the posterior distribution to the variable `exp\_value`  
exp\_value <- B\*mu + (1-B)\*Y   
  
# Assign the standard error of the posterior distribution to the variable `se`  
se <- sqrt( 1/ (1/sigma^2 + 1/tau^2))  
  
# Using the `pnorm` function, calculate the probability that the actual spread was less than 0 (in Trump's favor). Print this value to the console.  
pnorm(0, exp\_value, se)

## [1] 0.3203769

1. We had set the prior variance to 0.01, reflecting that these races are often close.

Change the prior variance to include values ranging from 0.005 to 0.05 and observe how the probability of Trump winning Florida changes by making a plot.

# Define the variables from previous exercises  
mu <- 0  
sigma <- results$se  
Y <- results$avg  
  
# Define a variable `taus` as different values of tau  
taus <- seq(0.005, 0.05, len = 100)  
  
# Create a function called `p\_calc` that generates `B` and calculates the probability of the spread being less than 0  
p\_calc <- function(tau) {  
 B <- sigma^2 / (sigma^2 + tau^2)  
 ev <- B\*mu + (1-B)\*Y  
 se <- sqrt(1 / (1/sigma^2 + 1/tau^2))  
 pnorm(0,ev,se)  
}  
  
# Create a vector called `ps` by applying the function `p\_calc` across values in `taus`  
ps <- p\_calc(taus)  
  
# Plot `taus` on the x-axis and `ps` on the y-axis  
plot(taus, ps)



# Section 6 Overview

In Section 6, you will learn about election forecasting, building on what you’ve learned in the previous sections about statistical modeling and Bayesian statistics.

After completing Section 6, you will be able to:

* Understand how pollsters use hierarchical models to forecast the results of elections.
* Incorporate multiple sources of variability into a mathematical model to make predictions.
* Construct confidence intervals that better model deviations such as those seen in election data using the t-distribution.

## Election Forecasting

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#bayesian-approach).

**Key points**

* In our model:
  + The spread describes our best guess in the absence of polling data. We set and using historical data.
  + The average of observed data describes randomness due to sampling and the pollster effect.
* Because the posterior distribution is normal, we can report a 95% *credible interval* that has a 95% chance of overlapping the parameter using and .
* Given an estimate of and , we can use pnorm to compute the probability that .
* It is common to see a general bias that affects all pollsters in the same way. This bias cannot be predicted or measured before the election. We will include a term in later models to account for this variability.

*Code: Definition of results object*

This code defines the results object used for empirical Bayes election forecasting.

polls <- polls\_us\_election\_2016 %>%  
 filter(state == "U.S." & enddate >= "2016-10-31" &  
 (grade %in% c("A+", "A", "A-", "B+") | is.na(grade))) %>%  
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
one\_poll\_per\_pollster <- polls %>% group\_by(pollster) %>%  
 filter(enddate == max(enddate)) %>%  
 ungroup()  
  
results <- one\_poll\_per\_pollster %>%  
 summarize(avg = mean(spread), se = sd(spread)/sqrt(length(spread))) %>%  
 mutate(start = avg - 1.96\*se, end = avg + 1.96\*se)

*Code: Computing the posterior mean, standard error, credible interval and probability*

Note that to compute an exact 95% credible interval, we would use qnorm(.975) instead of 1.96.

mu <- 0  
tau <- 0.035  
sigma <- results$se  
Y <- results$avg  
B <- sigma^2 / (sigma^2 + tau^2)  
posterior\_mean <- B\*mu + (1-B)\*Y  
posterior\_se <- sqrt(1 / (1/sigma^2 + 1/tau^2))  
  
posterior\_mean

## [1] 0.02808534

posterior\_se

## [1] 0.006149604

# 95% credible interval  
posterior\_mean + c(-1.96, 1.96)\*posterior\_se

## [1] 0.01603212 0.04013857

# probability of d > 0  
1 - pnorm(0, posterior\_mean, posterior\_se)

## [1] 0.9999975

## Mathematical Representations of Models

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#mathematical-representations-of-models).

**Key points**

* If we collect several polls with measured spreads with a sample size of , these random variables have expected value and standard error .
* We represent each measurement as where:
  + The index represents the different pollsters
  + The index represents the different polls
  + is the th poll by the th pollster
  + is the actual spread of the election
  + is the general bias affecting all pollsters
  + represents the house effect for the th pollster
  + represents the random error associated with the th poll.
* The sample average is now with standard deviation .
* The standard error of the general bias does not get reduced by averaging multiple polls, which increases the variability of our final estimate.

*Code: Simulated data with*

J <- 6  
N <- 2000  
d <- .021  
p <- (d+1)/2  
X <- d + rnorm(J, 0, 2\*sqrt(p\*(1-p)/N))

*Code: Simulated data with*

I <- 5  
J <- 6  
N <- 2000  
d <- .021  
p <- (d+1)/2  
X <- sapply(1:I, function(i){  
 d + rnorm(J, 0, 2\*sqrt(p\*(1-p)/N))  
})

*Code: Simulated data with*

I <- 5  
J <- 6  
N <- 2000  
d <- .021  
p <- (d+1)/2  
h <- rnorm(I, 0, 0.025) # assume standard error of pollster-to-pollster variability is 0.025  
X <- sapply(1:I, function(i){  
 d + rnorm(J, 0, 2\*sqrt(p\*(1-p)/N))  
})

*Code: Calculating probability of with general bias*

Note that sigma now includes an estimate of the variability due to general bias .

mu <- 0  
tau <- 0.035  
sigma <- sqrt(results$se^2 + .025^2)  
Y <- results$avg  
B <- sigma^2 / (sigma^2 + tau^2)  
  
posterior\_mean <- B\*mu + (1-B)\*Y  
posterior\_se <- sqrt(1 / (1/sigma^2 + 1/tau^2))  
  
1 - pnorm(0, posterior\_mean, posterior\_se)

## [1] 0.8174373

## Predicting the Electoral College

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#predicting-the-electoral-college).

**Key points**

* In the US election, each state has a certain number of votes that are won all-or-nothing based on the popular vote result in that state (with minor exceptions not discussed here).
* We use the left\_join() function to combine the number of electoral votes with our poll results.
* For each state, we apply a Bayesian approach to generate an Election Day . We keep our prior simple by assuming an expected value of 0 and a standard deviation based on recent history of 0.02.
* We can run a Monte Carlo simulation that for each iteration simulates poll results in each state using that state’s average and standard deviation, awards electoral votes for each state to Clinton if the spread is greater than 0, then compares the number of electoral votes won to the number of votes required to win the election (over 269).
* If we run a Monte Carlo simulation for the electoral college without accounting for general bias, we overestimate Clinton’s chances of winning at over 99%.
* If we include a general bias term, the estimated probability of Clinton winning decreases significantly.

*Code: Top 5 states ranked by electoral votes*

The results\_us\_election\_2016 object is defined in the **dslabs** package:

head(results\_us\_election\_2016)

## state electoral\_votes clinton trump others  
## 1 California 55 61.7 31.6 6.7  
## 2 Texas 38 43.2 52.2 4.5  
## 3 Florida 29 47.8 49.0 3.2  
## 4 New York 29 59.0 36.5 4.5  
## 5 Illinois 20 55.8 38.8 5.4  
## 6 Pennsylvania 20 47.9 48.6 3.6

results\_us\_election\_2016 %>% arrange(desc(electoral\_votes)) %>% top\_n(5, electoral\_votes)

## state electoral\_votes clinton trump others  
## 1 California 55 61.7 31.6 6.7  
## 2 Texas 38 43.2 52.2 4.5  
## 3 Florida 29 47.8 49.0 3.2  
## 4 New York 29 59.0 36.5 4.5  
## 5 Illinois 20 55.8 38.8 5.4  
## 6 Pennsylvania 20 47.9 48.6 3.6

*Code: Computing the average and standard deviation for each state*

results <- polls\_us\_election\_2016 %>%  
 filter(state != "U.S." &  
 !grepl("CD", "state") &  
 enddate >= "2016-10-31" &  
 (grade %in% c("A+", "A", "A-", "B+") | is.na(grade))) %>%  
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100) %>%  
 group\_by(state) %>%  
 summarize(avg = mean(spread), sd = sd(spread), n = n()) %>%  
 mutate(state = as.character(state))  
  
# 10 closest races = battleground states  
results %>% arrange(abs(avg))

## # A tibble: 47 x 4  
## state avg sd n  
## <chr> <dbl> <dbl> <int>  
## 1 Florida 0.00356 0.0163 7  
## 2 North Carolina -0.00730 0.0306 9  
## 3 Ohio -0.0104 0.0252 6  
## 4 Nevada 0.0169 0.0441 7  
## 5 Iowa -0.0197 0.0437 3  
## 6 Michigan 0.0209 0.0203 6  
## 7 Arizona -0.0326 0.0270 9  
## 8 Pennsylvania 0.0353 0.0116 9  
## 9 New Mexico 0.0389 0.0226 6  
## 10 Georgia -0.0448 0.0238 4  
## # ... with 37 more rows

# joining electoral college votes and results  
results <- left\_join(results, results\_us\_election\_2016, by="state")  
  
# states with no polls: note Rhode Island and District of Columbia = Democrat  
results\_us\_election\_2016 %>% filter(!state %in% results$state)

## state electoral\_votes clinton trump others  
## 1 Rhode Island 4 54.4 38.9 6.7  
## 2 Alaska 3 36.6 51.3 12.2  
## 3 Wyoming 3 21.9 68.2 10.0  
## 4 District of Columbia 3 90.9 4.1 5.0

# assigns sd to states with just one poll as median of other sd values  
results <- results %>%  
 mutate(sd = ifelse(is.na(sd), median(results$sd, na.rm = TRUE), sd))

*Code: Calculating the posterior mean and posterior standard error*

mu <- 0  
tau <- 0.02  
results %>% mutate(sigma = sd/sqrt(n),  
 B = sigma^2/ (sigma^2 + tau^2),  
 posterior\_mean = B\*mu + (1-B)\*avg,  
 posterior\_se = sqrt( 1 / (1/sigma^2 + 1/tau^2))) %>%  
 arrange(abs(posterior\_mean))

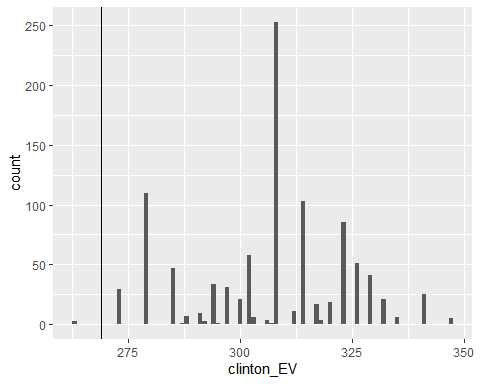
## # A tibble: 47 x 12  
## state avg sd n electoral\_votes clinton trump others sigma  
## <chr> <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 Florida 0.00356 0.0163 7 29 47.8 49 3.2 0.00618  
## 2 North Car~ -0.00730 0.0306 9 15 46.2 49.8 4 0.0102   
## 3 Iowa -0.0197 0.0437 3 6 41.7 51.1 7.1 0.0252   
## 4 Ohio -0.0104 0.0252 6 18 43.5 51.7 4.8 0.0103   
## 5 Nevada 0.0169 0.0441 7 6 47.9 45.5 6.6 0.0167   
## 6 Michigan 0.0209 0.0203 6 16 47.3 47.5 5.2 0.00827  
## 7 Arizona -0.0326 0.0270 9 11 45.1 48.7 6.2 0.00898  
## 8 New Mexico 0.0389 0.0226 6 5 48.3 40 11.7 0.00921  
## 9 Georgia -0.0448 0.0238 4 16 45.9 51 3.1 0.0119   
## 10 Pennsylva~ 0.0353 0.0116 9 20 47.9 48.6 3.6 0.00387  
## # ... with 37 more rows, and 3 more variables: B <dbl>, posterior\_mean <dbl>,  
## # posterior\_se <dbl>

*Code: Monte Carlo simulation of Election Night results (no general bias)*

mu <- 0  
tau <- 0.02  
clinton\_EV <- replicate(1000, {  
 results %>% mutate(sigma = sd/sqrt(n),  
 B = sigma^2/ (sigma^2 + tau^2),  
 posterior\_mean = B\*mu + (1-B)\*avg,  
 posterior\_se = sqrt( 1 / (1/sigma^2 + 1/tau^2)),  
 simulated\_result = rnorm(length(posterior\_mean), posterior\_mean, posterior\_se),  
 clinton = ifelse(simulated\_result > 0, electoral\_votes, 0)) %>% # award votes if Clinton wins state  
 summarize(clinton = sum(clinton)) %>% # total votes for Clinton  
 .$clinton + 7 # 7 votes for Rhode Island and DC  
})  
mean(clinton\_EV > 269) # over 269 votes wins election

## [1] 0.998

# histogram of outcomes  
data.frame(clinton\_EV) %>%  
 ggplot(aes(clinton\_EV)) +  
 geom\_histogram(binwidth = 1) +  
 geom\_vline(xintercept = 269)



*Code: Monte Carlo simulation including general bias*

mu <- 0  
tau <- 0.02  
bias\_sd <- 0.03  
clinton\_EV\_2 <- replicate(1000, {  
 results %>% mutate(sigma = sqrt(sd^2/(n) + bias\_sd^2), # added bias\_sd term  
 B = sigma^2/ (sigma^2 + tau^2),  
 posterior\_mean = B\*mu + (1-B)\*avg,  
 posterior\_se = sqrt( 1 / (1/sigma^2 + 1/tau^2)),  
 simulated\_result = rnorm(length(posterior\_mean), posterior\_mean, posterior\_se),  
 clinton = ifelse(simulated\_result > 0, electoral\_votes, 0)) %>% # award votes if Clinton wins state  
 summarize(clinton = sum(clinton)) %>% # total votes for Clinton  
 .$clinton + 7 # 7 votes for Rhode Island and DC  
})  
mean(clinton\_EV\_2 > 269) # over 269 votes wins election

## [1] 0.832

## Forecasting

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#forecasting).

**Key points**

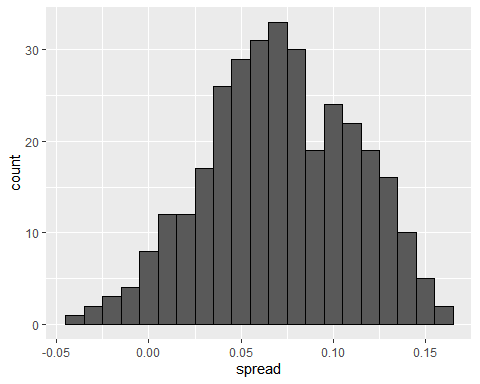
* In poll results, is not fixed over time. Variability within a single pollster comes from time variation.
* In order to forecast, our model must include a bias term to model the time effect.
* Pollsters also try to estimate , the trend of given time using a model like:
* Once we decide on a model, we can use historical data and current data to estimate the necessary parameters to make predictions.

*Code: Variability across one pollster*

# select all national polls by one pollster  
one\_pollster <- polls\_us\_election\_2016 %>%  
 filter(pollster == "Ipsos" & state == "U.S.") %>%  
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
# the observed standard error is higher than theory predicts  
se <- one\_pollster %>%  
 summarize(empirical = sd(spread),  
 theoretical = 2\*sqrt(mean(spread)\*(1-mean(spread))/min(samplesize)))  
se

## empirical theoretical  
## 1 0.04025194 0.03256719

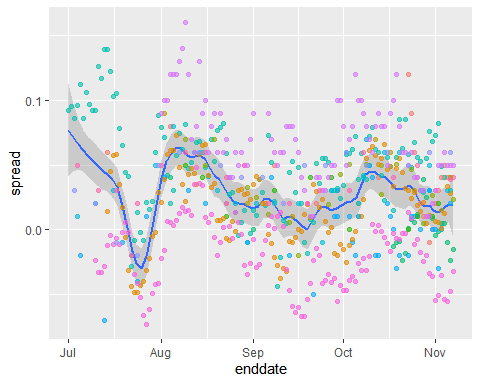
# the distribution of the data is not normal  
one\_pollster %>% ggplot(aes(spread)) +  
 geom\_histogram(binwidth = 0.01, color = "black")



*Code: Trend across time for several pollsters*

polls\_us\_election\_2016 %>%  
 filter(state == "U.S." & enddate >= "2016-07-01") %>%  
 group\_by(pollster) %>%  
 filter(n() >= 10) %>%  
 ungroup() %>%  
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100) %>%  
 ggplot(aes(enddate, spread)) +  
 geom\_smooth(method = "loess", span = 0.1) +  
 geom\_point(aes(color = pollster), show.legend = FALSE, alpha = 0.6)

## `geom\_smooth()` using formula 'y ~ x'



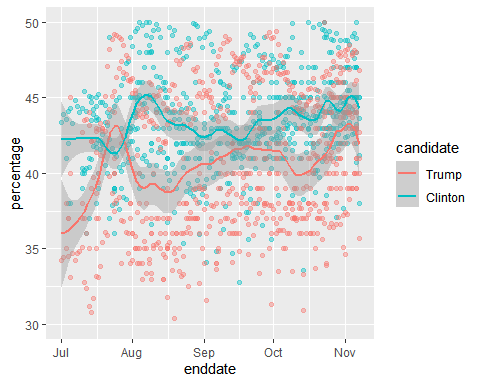
*Code: Plotting raw percentages across time*

polls\_us\_election\_2016 %>%  
 filter(state == "U.S." & enddate >= "2016-07-01") %>%  
 select(enddate, pollster, rawpoll\_clinton, rawpoll\_trump) %>%  
 rename(Clinton = rawpoll\_clinton, Trump = rawpoll\_trump) %>%  
 gather(candidate, percentage, -enddate, -pollster) %>%  
 mutate(candidate = factor(candidate, levels = c("Trump", "Clinton"))) %>%  
 group\_by(pollster) %>%  
 filter(n() >= 10) %>%  
 ungroup() %>%  
 ggplot(aes(enddate, percentage, color = candidate)) +  
 geom\_point(show.legend = FALSE, alpha = 0.4) +  
 geom\_smooth(method = "loess", span = 0.15) +  
 scale\_y\_continuous(limits = c(30, 50))

## `geom\_smooth()` using formula 'y ~ x'

## Warning: Removed 22 rows containing non-finite values (stat\_smooth).

## Warning: Removed 22 rows containing missing values (geom\_point).



## Assessment - Election Forecasting

1. For each poll in the polling data set, use the CLT to create a 95% confidence interval for the spread.

Create a new table called cis that contains columns for the lower and upper limits of the confidence intervals.

# Create a table called `polls` that filters by state, date, and reports the spread  
polls <- polls\_us\_election\_2016 %>%   
 filter(state != "U.S." & enddate >= "2016-10-31") %>%   
 mutate(spread = rawpoll\_clinton/100 - rawpoll\_trump/100)  
  
# Create an object called `cis` that has the columns indicated in the instructions  
cis <- polls %>% mutate(X\_hat = (spread+1)/2, se = 2\*sqrt(X\_hat\*(1-X\_hat)/samplesize), lower = spread - qnorm(0.975)\*se, upper = spread + qnorm(0.975)\*se) %>% select(state, startdate, enddate, pollster, grade, spread, lower, upper)

1. You can add the final result to the cis table you just created using the left\_join function as shown in the sample code.

Now determine how often the 95% confidence interval includes the actual result.

# Add the actual results to the `cis` data set  
add <- results\_us\_election\_2016 %>% mutate(actual\_spread = clinton/100 - trump/100) %>% select(state, actual\_spread)  
ci\_data <- cis %>% mutate(state = as.character(state)) %>% left\_join(add, by = "state")  
  
# Create an object called `p\_hits` that summarizes the proportion of confidence intervals that contain the actual value. Print this object to the console.  
p\_hits <- ci\_data %>% mutate(hit = lower <= actual\_spread & upper >= actual\_spread) %>% summarize(proportion\_hits = mean(hit))  
p\_hits

## proportion\_hits  
## 1 0.66133

1. Now find the proportion of hits for each pollster.

Show only pollsters with at least 5 polls and order them from best to worst. Show the number of polls conducted by each pollster and the FiveThirtyEight grade of each pollster.

# The `cis` data have already been loaded for you  
add <- results\_us\_election\_2016 %>% mutate(actual\_spread = clinton/100 - trump/100) %>% select(state, actual\_spread)  
ci\_data <- cis %>% mutate(state = as.character(state)) %>% left\_join(add, by = "state")  
  
# Create an object called `p\_hits` that summarizes the proportion of hits for each pollster that has at least 5 polls.  
p\_hits <- ci\_data %>% mutate(hit = lower <= actual\_spread & upper >= actual\_spread) %>% group\_by(pollster) %>% filter(n() >= 5) %>% summarize(proportion\_hits = mean(hit), n = n(), grade = grade[1]) %>% arrange(desc(proportion\_hits))  
p\_hits

## # A tibble: 13 x 4  
## pollster proportion\_hits n grade  
## <fct> <dbl> <int> <fct>  
## 1 Quinnipiac University 1 6 A-   
## 2 Emerson College 0.909 11 B   
## 3 Public Policy Polling 0.889 9 B+   
## 4 University of New Hampshire 0.857 7 B+   
## 5 Ipsos 0.807 119 A-   
## 6 Mitchell Research & Communications 0.8 5 D   
## 7 Gravis Marketing 0.783 23 B-   
## 8 Trafalgar Group 0.778 9 C   
## 9 Rasmussen Reports/Pulse Opinion Research 0.774 31 C+   
## 10 Remington 0.667 9 <NA>   
## 11 Google Consumer Surveys 0.588 102 B   
## 12 SurveyMonkey 0.577 357 C-   
## 13 YouGov 0.544 57 B

1. Repeat the previous exercise, but instead of pollster, stratify by state. Here we can’t show grades.

# The `cis` data have already been loaded for you  
add <- results\_us\_election\_2016 %>% mutate(actual\_spread = clinton/100 - trump/100) %>% select(state, actual\_spread)  
ci\_data <- cis %>% mutate(state = as.character(state)) %>% left\_join(add, by = "state")  
  
# Create an object called `p\_hits` that summarizes the proportion of hits for each state that has more than 5 polls.  
p\_hits <- ci\_data %>% mutate(hit = lower <= actual\_spread & upper >= actual\_spread) %>% group\_by(state) %>% filter(n() >= 5) %>% summarize(proportion\_hits = mean(hit), n = n()) %>% arrange(desc(proportion\_hits))  
p\_hits

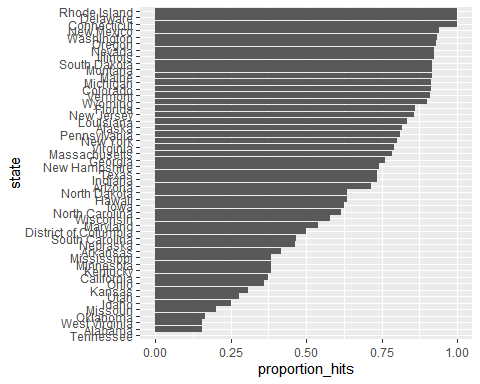
## # A tibble: 51 x 3  
## state proportion\_hits n  
## <chr> <dbl> <int>  
## 1 Connecticut 1 13  
## 2 Delaware 1 12  
## 3 Rhode Island 1 10  
## 4 New Mexico 0.941 17  
## 5 Washington 0.933 15  
## 6 Oregon 0.929 14  
## 7 Illinois 0.923 13  
## 8 Nevada 0.923 26  
## 9 Maine 0.917 12  
## 10 Montana 0.917 12  
## # ... with 41 more rows

1. Make a barplot based on the result from the previous exercise.

# The `p\_hits` data have already been loaded for you. Use the `head` function to examine it.  
head(p\_hits)

## # A tibble: 6 x 3  
## state proportion\_hits n  
## <chr> <dbl> <int>  
## 1 Connecticut 1 13  
## 2 Delaware 1 12  
## 3 Rhode Island 1 10  
## 4 New Mexico 0.941 17  
## 5 Washington 0.933 15  
## 6 Oregon 0.929 14

# Make a barplot of the proportion of hits for each state  
p\_hits %>% mutate(state = reorder(state, proportion\_hits)) %>%  
 ggplot(aes(state, proportion\_hits)) +   
 geom\_bar(stat = "identity") +  
 coord\_flip()



1. Even if a forecaster’s confidence interval is incorrect, the overall predictions will do better if they correctly called the right winner.

Add two columns to the cis table by computing, for each poll, the difference between the predicted spread and the actual spread, and define a column hit that is true if the signs are the same.

# The `cis` data have already been loaded. Examine it using the `head` function.  
head(cis)

## state startdate enddate pollster grade spread  
## 1 New Mexico 2016-11-06 2016-11-06 Zia Poll <NA> 0.02  
## 2 Virginia 2016-11-03 2016-11-04 Public Policy Polling B+ 0.05  
## 3 Iowa 2016-11-01 2016-11-04 Selzer & Company A+ -0.07  
## 4 Wisconsin 2016-10-26 2016-10-31 Marquette University A 0.06  
## 5 North Carolina 2016-11-04 2016-11-06 Siena College A 0.00  
## 6 Georgia 2016-11-06 2016-11-06 Landmark Communications B -0.03  
## lower upper  
## 1 -0.001331221 0.0413312213  
## 2 -0.005634504 0.1056345040  
## 3 -0.139125210 -0.0008747905  
## 4 0.004774064 0.1152259363  
## 5 -0.069295191 0.0692951912  
## 6 -0.086553820 0.0265538203

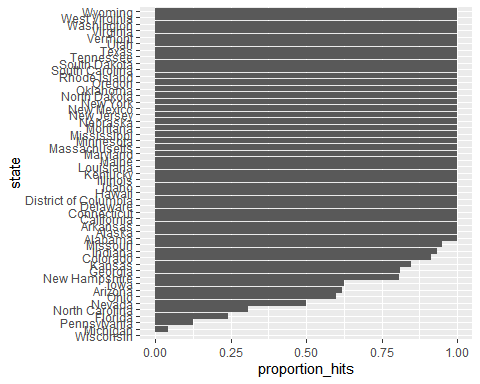
# Create an object called `errors` that calculates the difference between the predicted and actual spread and indicates if the correct winner was predicted  
errors <- cis %>% mutate(error = spread - ci\_data$actual\_spread, hit = sign(spread) == sign(ci\_data$actual\_spread))  
  
# Examine the last 6 rows of `errors`  
tail(errors)

## state startdate enddate pollster grade spread  
## 807 Utah 2016-10-04 2016-11-06 YouGov B -0.0910  
## 808 Utah 2016-10-25 2016-10-31 Google Consumer Surveys B -0.0121  
## 809 South Dakota 2016-10-28 2016-11-02 Ipsos A- -0.1875  
## 810 Washington 2016-10-21 2016-11-02 Ipsos A- 0.0838  
## 811 Utah 2016-11-01 2016-11-07 Google Consumer Surveys B -0.1372  
## 812 Oregon 2016-10-21 2016-11-02 Ipsos A- 0.0905  
## lower upper error hit  
## 807 -0.1660704570 -0.01592954 0.0890 TRUE  
## 808 -0.1373083389 0.11310834 0.1679 TRUE  
## 809 -0.3351563485 -0.03984365 0.1105 TRUE  
## 810 -0.0004028265 0.16800283 -0.0782 TRUE  
## 811 -0.2519991224 -0.02240088 0.0428 TRUE  
## 812 -0.0019261469 0.18292615 -0.0195 TRUE

1. Create an object called p\_hits that contains the proportion of instances when the sign of the actual spread matches the predicted spread for states with 5 or more polls.

Make a barplot based on the result from the previous exercise that shows the proportion of times the sign of the spread matched the actual result for the data in p\_hits.

# Create an object called `errors` that calculates the difference between the predicted and actual spread and indicates if the correct winner was predicted  
errors <- cis %>% mutate(error = spread - ci\_data$actual\_spread, hit = sign(spread) == sign(ci\_data$actual\_spread))  
  
# Create an object called `p\_hits` that summarizes the proportion of hits for each state that has 5 or more polls  
p\_hits <- errors %>% group\_by(state) %>% filter(n() >= 5) %>% summarize(proportion\_hits = mean(hit), n = n())  
  
# Make a barplot of the proportion of hits for each state  
p\_hits %>% mutate(state = reorder(state, proportion\_hits)) %>%  
 ggplot(aes(state, proportion\_hits)) +   
 geom\_bar(stat = "identity") +  
 coord\_flip()



1. In the previous graph, we see that most states’ polls predicted the correct winner 100% of the time.

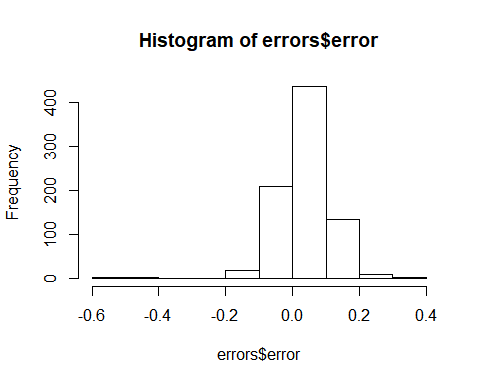
Only a few states polls’ were incorrect more than 25% of the time. Wisconsin got every single poll wrong. In Pennsylvania and Michigan, more than 90% of the polls had the signs wrong.

Make a histogram of the errors. What is the median of these errors?

# The `errors` data have already been loaded. Examine them using the `head` function.  
head(errors)

## state startdate enddate pollster grade spread  
## 1 New Mexico 2016-11-06 2016-11-06 Zia Poll <NA> 0.02  
## 2 Virginia 2016-11-03 2016-11-04 Public Policy Polling B+ 0.05  
## 3 Iowa 2016-11-01 2016-11-04 Selzer & Company A+ -0.07  
## 4 Wisconsin 2016-10-26 2016-10-31 Marquette University A 0.06  
## 5 North Carolina 2016-11-04 2016-11-06 Siena College A 0.00  
## 6 Georgia 2016-11-06 2016-11-06 Landmark Communications B -0.03  
## lower upper error hit  
## 1 -0.001331221 0.0413312213 -0.063 TRUE  
## 2 -0.005634504 0.1056345040 -0.004 TRUE  
## 3 -0.139125210 -0.0008747905 0.024 TRUE  
## 4 0.004774064 0.1152259363 0.067 FALSE  
## 5 -0.069295191 0.0692951912 0.036 FALSE  
## 6 -0.086553820 0.0265538203 0.021 TRUE

# Generate a histogram of the error  
hist(errors$error)



# Calculate the median of the errors. Print this value to the console.  
median(errors$error)

## [1] 0.037

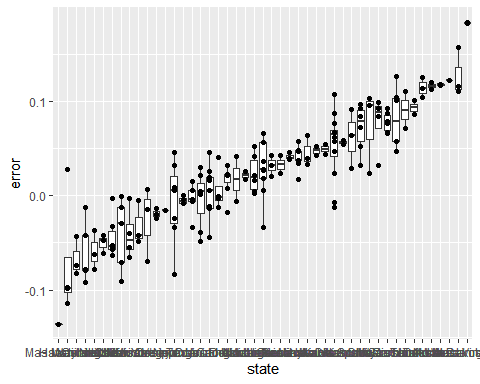
1. We see that, at the state level, the median error was slightly in favor of Clinton. The distribution is not centered at 0, but at 0.037. This value represents the general bias we described in an earlier section.

Create a boxplot to examine if the bias was general to all states or if it affected some states differently. Filter the data to include only pollsters with grades B+ or higher.

# The `errors` data have already been loaded. Examine them using the `head` function.  
head(errors)

## state startdate enddate pollster grade spread  
## 1 New Mexico 2016-11-06 2016-11-06 Zia Poll <NA> 0.02  
## 2 Virginia 2016-11-03 2016-11-04 Public Policy Polling B+ 0.05  
## 3 Iowa 2016-11-01 2016-11-04 Selzer & Company A+ -0.07  
## 4 Wisconsin 2016-10-26 2016-10-31 Marquette University A 0.06  
## 5 North Carolina 2016-11-04 2016-11-06 Siena College A 0.00  
## 6 Georgia 2016-11-06 2016-11-06 Landmark Communications B -0.03  
## lower upper error hit  
## 1 -0.001331221 0.0413312213 -0.063 TRUE  
## 2 -0.005634504 0.1056345040 -0.004 TRUE  
## 3 -0.139125210 -0.0008747905 0.024 TRUE  
## 4 0.004774064 0.1152259363 0.067 FALSE  
## 5 -0.069295191 0.0692951912 0.036 FALSE  
## 6 -0.086553820 0.0265538203 0.021 TRUE

# Create a boxplot showing the errors by state for polls with grades B+ or higher  
errors %>% filter(grade %in% c("A+","A","A-","B+") | is.na(grade)) %>% mutate(state = reorder(state, error)) %>% ggplot(aes(state, error)) + geom\_boxplot() + geom\_point()



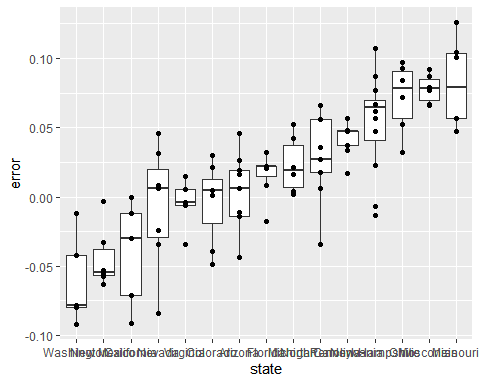
1. Some of these states only have a few polls.

Repeat the previous exercise to plot the errors for each state, but only include states with five good polls or more.

# The `errors` data have already been loaded. Examine them using the `head` function.  
head(errors)

## state startdate enddate pollster grade spread  
## 1 New Mexico 2016-11-06 2016-11-06 Zia Poll <NA> 0.02  
## 2 Virginia 2016-11-03 2016-11-04 Public Policy Polling B+ 0.05  
## 3 Iowa 2016-11-01 2016-11-04 Selzer & Company A+ -0.07  
## 4 Wisconsin 2016-10-26 2016-10-31 Marquette University A 0.06  
## 5 North Carolina 2016-11-04 2016-11-06 Siena College A 0.00  
## 6 Georgia 2016-11-06 2016-11-06 Landmark Communications B -0.03  
## lower upper error hit  
## 1 -0.001331221 0.0413312213 -0.063 TRUE  
## 2 -0.005634504 0.1056345040 -0.004 TRUE  
## 3 -0.139125210 -0.0008747905 0.024 TRUE  
## 4 0.004774064 0.1152259363 0.067 FALSE  
## 5 -0.069295191 0.0692951912 0.036 FALSE  
## 6 -0.086553820 0.0265538203 0.021 TRUE

# Create a boxplot showing the errors by state for states with at least 5 polls with grades B+ or higher  
errors %>% filter(grade %in% c("A+","A","A-","B+") | is.na(grade)) %>% group\_by(state) %>% filter(n() >= 5) %>% ungroup() %>%mutate(state = reorder(state, error)) %>% ggplot(aes(state, error)) + geom\_boxplot() + geom\_point()



## The t-Distribution

The textbook for this section is available [here](https://rafalab.github.io/dsbook/models.html#t-dist).

**Key points**

* In models where we must estimate two parameters, and , the Central Limit Theorem can result in overconfident confidence intervals for sample sizes smaller than approximately 30.
* If the population data are known to follow a normal distribution, theory tells us how much larger to make confidence intervals to account for estimation of .
* Given as an estimate of , then follows a t-distribution with degrees of freedom.
* Degrees of freedom determine the weight of the tails of the distribution. Small values of degrees of freedom lead to increased probabilities of extreme values.
* We can determine confidence intervals using the t-distribution instead of the normal distribution by calculating the desired quantile with the function qt().

*Code: Calculating 95% confidence intervals with the t-distribution*

z <- qt(0.975, nrow(one\_poll\_per\_pollster) - 1)  
one\_poll\_per\_pollster %>%  
 summarize(avg = mean(spread), moe = z\*sd(spread)/sqrt(length(spread))) %>%  
 mutate(start = avg - moe, end = avg + moe)

## # A tibble: 1 x 4  
## avg moe start end  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.0290 0.0134 0.0156 0.0424

# quantile from t-distribution versus normal distribution  
qt(0.975, 14) # 14 = nrow(one\_poll\_per\_pollster) - 1

## [1] 2.144787

qnorm(0.975)

## [1] 1.959964

## Assessment - The t-Distribution

1. We know that, with a normal distribution, only 5% of values are more than 2 standard deviations away from the mean.

Calculate the probability of seeing t-distributed random variables being more than 2 in absolute value when the degrees of freedom are 3.

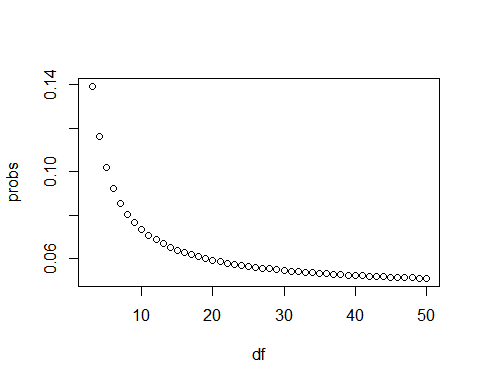
# Calculate the probability of seeing t-distributed random variables being more than 2 in absolute value when 'df = 3'.  
1 - pt(2, df = 3) + pt(-2, df = 3)

## [1] 0.139326

1. Now use sapply to compute the same probability for degrees of freedom from 3 to 50.

Make a plot and notice when this probability converges to the normal distribution’s 5%.

# Generate a vector 'df' that contains a sequence of numbers from 3 to 50  
df <- c(3:50)  
  
# Make a function called 'pt\_func' that calculates the probability that a value is more than |2| for any degrees of freedom   
pt\_function <- function(f) {  
1 - pt(2, f) + pt(-2, f)  
}  
  
# Generate a vector 'probs' that uses the `pt\_func` function to calculate the probabilities  
probs <- sapply(df, pt\_function)  
  
# Plot 'df' on the x-axis and 'probs' on the y-axis  
plot(df, probs)



1. In a previous section, we repeatedly took random samples of 50 heights from a distribution of heights.

We noticed that about 95% of the samples had confidence intervals spanning the true population mean.

Re-do this Monte Carlo simulation, but now instead of , use . Notice what happens to the proportion of hits.

# Use the sample code to generate 'x', a vector of male heights  
x <- heights %>% filter(sex == "Male") %>%  
 .$height  
  
# Create variables for the mean height 'mu', the sample size 'N', and the number of times the simulation should run 'B'  
mu <- mean(x)  
N <- 15  
B <- 10000  
  
# Use the `set.seed` function to make sure your answer matches the expected result after random sampling  
set.seed(1)  
  
# Generate a logical vector 'res' that contains the results of the simulations  
res <- replicate(B, {  
X <- sample(x, N, replace = TRUE)  
interval <- mean(X) + c(-1,1)\*qnorm(0.975)\*sd(X)/sqrt(N)  
between(mu, interval[1], interval[2])  
}  
)  
  
# Calculate the proportion of times the simulation produced values within the 95% confidence interval. Print this value to the console.  
p\_hit <- mean(res)  
p\_hit

## [1] 0.9331

1. is not that big. We know that heights are normally distributed, so the t-distribution should apply.

Repeat the previous Monte Carlo simulation using the t-distribution instead of using the normal distribution to construct the confidence intervals.

What are the proportion of 95% confidence intervals that span the actual mean height now?

# The vector of filtered heights 'x' has already been loaded for you. Calculate the mean.  
mu <- mean(x)  
  
# Use the same sampling parameters as in the previous exercise.  
set.seed(1)  
N <- 15  
B <- 10000  
  
# Generate a logical vector 'res' that contains the results of the simulations using the t-distribution  
res <- replicate(B, {  
X <- sample(x, N, replace = TRUE)  
interval <- mean(X) + c(-1,1)\*qt(0.975, N - 1) \* sd(X) / sqrt(N)  
between(mu, interval[1], interval[2])  
}  
)  
  
# Calculate the proportion of times the simulation produced values within the 95% confidence interval. Print this value to the console.  
p\_hit <- mean(res)  
p\_hit

## [1] 0.9512

1. Why did the t-distribution confidence intervals work so much better?

* ☒ A. The t-distribution takes the variability into account and generates larger confidence intervals.
* ☐ B. Because the t-distribution shifts the intervals in the direction towards the actual mean.
* ☐ C. This was just a chance occurrence. If we run it again, the CLT will work better.
* ☐ D. The t-distribution is always a better approximation than the normal distribution.

# Section 7 Overview

In Section 7, you will learn how to use association and chi-squared tests to perform inference for binary, categorical, and ordinal data through an example looking at research funding rates.

After completing Section 7, you will be able to:

* Use association and chi-squared tests to perform inference on binary, categorical, and ordinal data.
* Calculate an odds ratio to get an idea of the magnitude of an observed effect.

## Association Tests

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#association-tests) up to and including the [textbook section](https://rafalab.github.io/dsbook/inference.html#two-by-two-tables) on two-by-two tables.

**Key points**

* We learn how to determine the probability that an observation is due to random variability given categorical, binary or ordinal data.
* Fisher’s exact test determines the p-value as the probability of observing an outcome as extreme or more extreme than the observed outcome given the null distribution.
* Data from a binary experiment are often summarized in *two-by-two tables*.
* The p-value can be calculated from a two-by-two table using Fisher’s exact test with the function fisher.test().

*Code: Research funding rates example*

# load and inspect research funding rates object  
data(research\_funding\_rates)  
research\_funding\_rates

## discipline applications\_total applications\_men applications\_women  
## 1 Chemical sciences 122 83 39  
## 2 Physical sciences 174 135 39  
## 3 Physics 76 67 9  
## 4 Humanities 396 230 166  
## 5 Technical sciences 251 189 62  
## 6 Interdisciplinary 183 105 78  
## 7 Earth/life sciences 282 156 126  
## 8 Social sciences 834 425 409  
## 9 Medical sciences 505 245 260  
## awards\_total awards\_men awards\_women success\_rates\_total success\_rates\_men  
## 1 32 22 10 26.2 26.5  
## 2 35 26 9 20.1 19.3  
## 3 20 18 2 26.3 26.9  
## 4 65 33 32 16.4 14.3  
## 5 43 30 13 17.1 15.9  
## 6 29 12 17 15.8 11.4  
## 7 56 38 18 19.9 24.4  
## 8 112 65 47 13.4 15.3  
## 9 75 46 29 14.9 18.8  
## success\_rates\_women  
## 1 25.6  
## 2 23.1  
## 3 22.2  
## 4 19.3  
## 5 21.0  
## 6 21.8  
## 7 14.3  
## 8 11.5  
## 9 11.2

# compute totals that were successful or not successful  
totals <- research\_funding\_rates %>%  
 select(-discipline) %>%  
 summarize\_all(funs(sum)) %>%  
 summarize(yes\_men = awards\_men,  
 no\_men = applications\_men - awards\_men,  
 yes\_women = awards\_women,  
 no\_women = applications\_women - awards\_women)

## Warning: `funs()` was deprecated in dplyr 0.8.0.  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))

# compare percentage of men/women with awards  
totals %>% summarize(percent\_men = yes\_men/(yes\_men + no\_men),  
 percent\_women = yes\_women/(yes\_women + no\_women))

## percent\_men percent\_women  
## 1 0.17737 0.1489899

*Code: Two-by-two table and p-value for the Lady Tasting Tea problem*

tab <- matrix(c(3,1,1,3), 2, 2)  
rownames(tab) <- c("Poured Before", "Poured After")  
colnames(tab) <- c("Guessed Before", "Guessed After")  
tab

## Guessed Before Guessed After  
## Poured Before 3 1  
## Poured After 1 3

# p-value calculation with Fisher's Exact Test  
fisher.test(tab, alternative = "greater")

##   
## Fisher's Exact Test for Count Data  
##   
## data: tab  
## p-value = 0.2429  
## alternative hypothesis: true odds ratio is greater than 1  
## 95 percent confidence interval:  
## 0.3135693 Inf  
## sample estimates:  
## odds ratio   
## 6.408309

## Chi-Squared Tests

The textbook for this section is available [here](https://rafalab.github.io/dsbook/inference.html#chi-square-test) and [here](https://rafalab.github.io/dsbook/inference.html#odds-ratio).

**Key points**

* If the sums of the rows and the sums of the columns in the two-by-two table are fixed, then the hypergeometric distribution and Fisher’s exact test can be used. Otherwise, we must use the chi-squared test.
* The *chi-squared test* compares the observed two-by-two table to the two-by-two table expected by the null hypothesis and asks how likely it is that we see a deviation as large as observed or larger by chance.
* The function chisq.test() takes a two-by-two table and returns the p-value from the chi-squared test.
* The *odds ratio* states how many times larger the odds of an outcome are for one group relative to another group.
* A small p-value does not imply a large odds ratio. If a finding has a small p-value but also a small odds ratio, it may not be a practically significant or scientifically significant finding.
* Because the odds ratio is a ratio of ratios, there is no simple way to use the Central Limit Theorem to compute confidence intervals. There are advanced methods for computing confidence intervals for odds ratios that we do not discuss here.

*Code: Chi-squared test*

# compute overall funding rate  
funding\_rate <- totals %>%  
 summarize(percent\_total = (yes\_men + yes\_women) / (yes\_men + no\_men + yes\_women + no\_women)) %>%  
 .$percent\_total  
funding\_rate

## [1] 0.1654269

# construct two-by-two table for observed data  
two\_by\_two <- tibble(awarded = c("no", "yes"),  
 men = c(totals$no\_men, totals$yes\_men),  
 women = c(totals$no\_women, totals$yes\_women))  
two\_by\_two

## # A tibble: 2 x 3  
## awarded men women  
## <chr> <dbl> <dbl>  
## 1 no 1345 1011  
## 2 yes 290 177

# compute null hypothesis two-by-two table  
tibble(awarded = c("no", "yes"),  
 men = (totals$no\_men + totals$yes\_men) \* c(1-funding\_rate, funding\_rate),  
 women = (totals$no\_women + totals$yes\_women) \* c(1-funding\_rate, funding\_rate))

## # A tibble: 2 x 3  
## awarded men women  
## <chr> <dbl> <dbl>  
## 1 no 1365. 991.  
## 2 yes 270. 197.

# chi-squared test  
chisq\_test <- two\_by\_two %>%  
 select(-awarded) %>% chisq.test()  
chisq\_test$p.value

## [1] 0.05091372

*Code: Odds ratio*

# odds of getting funding for men  
odds\_men <- (two\_by\_two$men[2] / sum(two\_by\_two$men)) /  
 (two\_by\_two$men[1] / sum(two\_by\_two$men))  
  
# odds of getting funding for women  
odds\_women <- (two\_by\_two$women[2] / sum(two\_by\_two$women)) /  
 (two\_by\_two$women[1] / sum(two\_by\_two$women))  
  
# odds ratio - how many times larger odds are for men than women  
odds\_men/odds\_women

## [1] 1.231554

Code: p-value and odds ratio responses to increasing sample size

# multiplying all observations by 10 decreases p-value without changing odds ratio  
two\_by\_two %>%  
 select(-awarded) %>%  
 mutate(men = men\*10, women = women\*10) %>%  
 chisq.test()

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: .  
## X-squared = 39.935, df = 1, p-value = 2.625e-10

## Assessment - Association and Chi-Squared Tests

1. In a previous exercise, we determined whether or not each poll predicted the correct winner for their state in the 2016 U.S. presidential election.

Each poll was also assigned a grade by the poll aggregator. Now we’re going to determine if polls rated A- made better predictions than polls rated C-.

In this exercise, filter the errors data for just polls with grades A- and C-. Calculate the proportion of times each grade of poll predicted the correct winner.

# The 'errors' data have already been loaded. Examine them using the `head` function.  
head(errors)

## state startdate enddate pollster grade spread  
## 1 New Mexico 2016-11-06 2016-11-06 Zia Poll <NA> 0.02  
## 2 Virginia 2016-11-03 2016-11-04 Public Policy Polling B+ 0.05  
## 3 Iowa 2016-11-01 2016-11-04 Selzer & Company A+ -0.07  
## 4 Wisconsin 2016-10-26 2016-10-31 Marquette University A 0.06  
## 5 North Carolina 2016-11-04 2016-11-06 Siena College A 0.00  
## 6 Georgia 2016-11-06 2016-11-06 Landmark Communications B -0.03  
## lower upper error hit  
## 1 -0.001331221 0.0413312213 -0.063 TRUE  
## 2 -0.005634504 0.1056345040 -0.004 TRUE  
## 3 -0.139125210 -0.0008747905 0.024 TRUE  
## 4 0.004774064 0.1152259363 0.067 FALSE  
## 5 -0.069295191 0.0692951912 0.036 FALSE  
## 6 -0.086553820 0.0265538203 0.021 TRUE

# Generate an object called 'totals' that contains the numbers of good and bad predictions for polls rated A- and C-  
totals <- errors %>% filter(grade %in% c("A-", "C-")) %>%  
 group\_by(grade,hit) %>% summarize(num = n()) %>% spread(grade, num)

## `summarise()` has grouped output by 'grade'. You can override using the `.groups` argument.

totals

## # A tibble: 2 x 3  
## hit `C-` `A-`  
## <lgl> <int> <int>  
## 1 FALSE 50 26  
## 2 TRUE 311 106

# Print the proportion of hits for grade A- polls to the console  
p\_hitA <- totals[[2,3]]/sum(totals[[3]])  
p\_hitA

## [1] 0.8030303

# Print the proportion of hits for grade C- polls to the console  
p\_hitC <- totals[[2,2]]/sum(totals[[2]])  
p\_hitC

## [1] 0.8614958

1. We found that the A- polls predicted the correct winner about 80% of the time in their states and C- polls predicted the correct winner about 86% of the time.

Use a chi-squared test to determine if these proportions are different.

# The 'totals' data have already been loaded. Examine them using the `head` function.  
head(totals)

## # A tibble: 2 x 3  
## hit `C-` `A-`  
## <lgl> <int> <int>  
## 1 FALSE 50 26  
## 2 TRUE 311 106

# Perform a chi-squared test on the hit data. Save the results as an object called 'chisq\_test'.  
chisq\_test <- totals %>%  
 select(-hit) %>% chisq.test()  
  
# Print the p-value of the chi-squared test to the console  
chisq\_test$p.value

## [1] 0.1467902

1. It doesn’t look like the grade A- polls performed significantly differently than the grade C- polls in their states.

Calculate the odds ratio to determine the magnitude of the difference in performance between these two grades of polls.

# The 'totals' data have already been loaded. Examine them using the `head` function.  
head(totals)

## # A tibble: 2 x 3  
## hit `C-` `A-`  
## <lgl> <int> <int>  
## 1 FALSE 50 26  
## 2 TRUE 311 106

# Generate a variable called `odds\_C` that contains the odds of getting the prediction right for grade C- polls  
odds\_C <- (totals[[2,2]] / sum(totals[[2]])) /   
 (totals[[1,2]] / sum(totals[[2]]))  
  
# Generate a variable called `odds\_A` that contains the odds of getting the prediction right for grade A- polls  
odds\_A <- (totals[[2,3]] / sum(totals[[3]])) /  
 (totals[[1,3]] / sum(totals[[3]]))  
  
# Calculate the odds ratio to determine how many times larger the odds ratio is for grade A- polls than grade C- polls  
odds\_A/odds\_C

## [1] 0.6554539

1. We did not find meaningful differences between the poll results from grade A- and grade C- polls in this subset of the data, which only contains polls for about a week before the election.

Imagine we expanded our analysis to include all election polls and we repeat our analysis. In this hypothetical scenario, we get that the p-value for the difference in prediction success if 0.0015 and the odds ratio describing the effect size of the performance of grade A- over grade B- polls is 1.07.

Based on what we learned in the last section, which statement reflects the best interpretation of this result?

* ☐ A. The p-value is below 0.05, so there is a significant difference. Grade A- polls are significantly better at predicting winners.
* ☐ B. The p-value is too close to 0.05 to call this a significant difference. We do not observe a difference in performance.
* ☒ C. The p-value is below 0.05, but the odds ratio is very close to 1. There is not a scientifically significant difference in performance.
* ☐ D. The p-value is below 0.05 and the odds ratio indicates that grade A- polls perform significantly better than grade C- polls.

## Comprehensive Assessment: Brexit

## Brexit poll analysis - Part 1

**Directions**

There are 12 multi-part problems in this comprehensive assessment that review concepts from the entire course. The problems are split over 3 pages. Make sure you read the instructions carefully and run all pre-exercise code.

For numeric entry problems, you have 10 attempts to input the correct answer. For true/false problems, you have 2 attempts.

If you have questions, visit the “Brexit poll analysis” discussion forum that follows the assessment.

*IMPORTANT*: Some of these exercises use **dslabs** datasets that were added in a July 2019 update. Make sure your package is up to date with the command install.packages("dslabs").

**Overview**

In June 2016, the United Kingdom (UK) held a referendum to determine whether the country would “Remain” in the European Union (EU) or “Leave” the EU. This referendum is commonly known as Brexit. Although the media and others interpreted poll results as forecasting “Remain” (), the actual proportion that voted “Remain” was only 48.1% () and the UK thus voted to leave the EU. Pollsters in the UK were criticized for overestimating support for “Remain”.

In this project, you will analyze real Brexit polling data to develop polling models to forecast Brexit results. You will write your own code in R and enter the answers on the edX platform.

**Important definitions**

***Data Import***

Import the brexit\_polls polling data from the **dslabs** package and set options for the analysis:

# suggested options  
options(digits = 3)  
  
# load brexit\_polls object  
data(brexit\_polls)

***Final Brexit parameters***

Define as the actual percent voting “Remain” on the Brexit referendum and as the actual spread of the Brexit referendum with “Remain” defined as the positive outcome:

p <- 0.481 # official proportion voting "Remain"  
d <- 2\*p-1 # official spread

#### Question 1: The final proportion of voters choosing “Remain” was p=0.481. Consider a poll with a sample of N=1500 voters.

What is the expected total number of voters in the sample choosing “Remain”?

p <- 0.481  
N <- 1500  
N\*p

## [1] 722

What is the standard error of the total number of voters in the sample choosing “Remain”?

sqrt(N\*p\*(1-p))

## [1] 19.4

What is the expected value of , the proportion of “Remain” voters?

X\_hat <- p  
X\_hat

## [1] 0.481

What is the standard error of , the proportion of “Remain” voters?

sqrt(p\*(1-p)/N)

## [1] 0.0129

What is the expected value of , the spread between the proportion of “Remain” voters and “Leave” voters?

2\*p-1

## [1] -0.038

What is the standard error of , the spread between the proportion of “Remain” voters and “Leave” voters?

2\*sqrt(p\*(1-p)/N)

## [1] 0.0258

#### Question 2: Load and inspect the brexit\_polls dataset from dslabs, which contains actual polling data for the 6 months before the Brexit vote.

Raw proportions of voters preferring “Remain”, “Leave”, and “Undecided” are available (remain, leave, undecided) The spread is also available (spread), which is the difference in the raw proportion of voters choosing “Remain” and the raw proportion choosing “Leave”.

Calculate x\_hat for each poll, the estimate of the proportion of voters choosing “Remain” on the referendum day (), given the observed spread and the relationship . Use mutate() to add a variable x\_hat to the `brexit\_polls object by filling in the skeleton code below:

brexit\_polls <- brexit\_polls %>%  
 mutate(x\_hat = \_\_\_\_\_\_\_\_\_\_)

What is the average of the observed spreads (spread)?

brexit\_polls <- brexit\_polls %>%  
 mutate(x\_hat = (spread+1)/2)  
  
mean(brexit\_polls$spread)

## [1] 0.0201

What is the standard deviation of the observed spreads?

sd(brexit\_polls$spread)

## [1] 0.0588

What is the average of x\_hat, the estimates of the parameter ?

mean(brexit\_polls$x\_hat)

## [1] 0.51

What is the standard deviation of x\_hat?

sd(brexit\_polls$x\_hat)

## [1] 0.0294

#### Question 3: Confidence interval of a Brexit poll

Consider the first poll in brexit\_polls, a YouGov poll run on the same day as the Brexit referendum:

brexit\_polls[1,]

## startdate enddate pollster poll\_type samplesize remain leave undecided  
## 1 2016-06-23 2016-06-23 YouGov Online 4772 0.52 0.48 0  
## spread x\_hat  
## 1 0.04 0.52

Use qnorm to compute the 95% confidence interval for .

What is the lower bound of the 95% confidence interval?

x\_hat <- 0.52  
N <- 4772  
se\_hat <- sqrt(x\_hat\*(1-x\_hat)/N)  
x\_hat - qnorm(.975)\*se\_hat

## [1] 0.506

What is the upper bound of the 95% confidence interval?

x\_hat + qnorm(.975)\*se\_hat

## [1] 0.534

Does the 95% confidence interval predict a winner (does not cover )? Does the 95% confidence interval cover the true value of observed during the referendum?

!between(0.5, x\_hat - qnorm(.975)\*se\_hat, x\_hat + qnorm(.975)\*se\_hat) # predicts winner

## [1] TRUE

between(0.481, x\_hat - qnorm(.975)\*se\_hat, x\_hat + qnorm(.975)\*se\_hat) # does not cover p

## [1] FALSE

* ☐ A. The interval predicts a winner and covers the true value of .
* ☒ B. The interval predicts a winner but does not cover the true value of .
* ☐ C. The interval does not predict a winner but does cover the true value of .
* ☐ D. The interval does not predict a winner and does not cover the true value of .

## Brexit poll analysis - Part 2

#### Question 4: Create the data frame june\_polls containing only Brexit polls ending in June 2016 (enddate of “2016-06-01” and later).

We will calculate confidence intervals for all polls and determine how many cover the true value of .

First, use mutate() to calculate a plug-in estimate se\_x\_hat for the standard error of the estimate for each poll given its sample size and value of (x\_hat). Second, use mutate() to calculate an estimate for the standard error of the spread for each poll given the value of se\_x\_hat. Then, use mutate() to calculate upper and lower bounds for 95% confidence intervals of the spread. Last, add a column hit that indicates whether the confidence interval for each poll covers the correct spread .

How many polls are in june\_polls?

june\_polls <- brexit\_polls %>%  
 filter(enddate >= "2016-06-01")  
 nrow(june\_polls)

## [1] 32

What proportion of polls have a confidence interval that covers the value 0?

june\_polls <- june\_polls %>%  
 mutate(se\_x\_hat = sqrt(x\_hat\*(1-x\_hat)/samplesize),  
 se\_spread = 2\*se\_x\_hat,  
 ci\_lower\_spread = spread - qnorm(0.975)\*se\_spread,  
 ci\_upper\_spread = spread + qnorm(0.975)\*se\_spread)  
 mean(june\_polls$ci\_lower\_spread < 0 & june\_polls$ci\_upper\_spread > 0)

## [1] 0.625

What proportion of polls predict “Remain” (confidence interval entirely above 0)?

mean(june\_polls$ci\_lower\_spread > 0)

## [1] 0.125

What proportion of polls have a confidence interval covering the true value of ?

june\_polls <- june\_polls %>%  
 mutate(hit = (2\*p-1 > ci\_lower\_spread) & (2\*p-1 < ci\_upper\_spread))  
 mean(june\_polls$hit)

## [1] 0.562

#### Question 5: Group and summarize the june\_polls object by pollster to find the proportion of hits for each pollster and the number of polls per pollster.

Use arrange() to sort by hit rate.

june\_polls %>%  
 group\_by(pollster) %>%  
 summarize(hit\_rate=mean(hit), n()) %>%  
 arrange(hit\_rate)

## # A tibble: 12 x 3  
## pollster hit\_rate `n()`  
## <fct> <dbl> <int>  
## 1 BMG Research 0 2  
## 2 ORB/Telegraph 0 1  
## 3 Populus 0 1  
## 4 ComRes 0.333 3  
## 5 Ipsos MORI 0.5 2  
## 6 Survation 0.5 2  
## 7 YouGov 0.556 9  
## 8 Opinium 0.667 3  
## 9 ORB 0.667 3  
## 10 ICM 1 3  
## 11 Survation/IG Group 1 1  
## 12 TNS 1 2

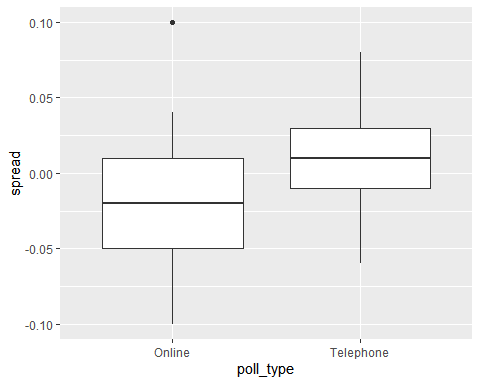
Which of the following are TRUE?

* ☐ A. Unbiased polls and pollsters will theoretically cover the correct value of the spread 50% of the time.
* ☐ B. Only one pollster had a 100% success rate in generating confidence intervals that covered the correct value of the spread.
* ☐ C. The pollster with the highest number of polls covered the correct value of the spread in their confidence interval over 60% of the time.
* ☐ D. All pollsters produced confidence intervals covering the correct spread in at least 1 of their polls.
* ☒ E. The results are consistent with a large general bias that affects all pollsters.

#### Question 6: Boxplot of Brexit polls by poll type

Make a boxplot of the spread in june\_polls by poll type.

june\_polls %>% group\_by(poll\_type) %>%  
 ggplot(aes(poll\_type,spread)) +  
 geom\_boxplot()



Which of the following are TRUE?

* ☐ A. Online polls tend to show support for “Remain” (spread > 0).
* ☒ B. Telephone polls tend to show support “Remain” (spread > 0).
* ☒ C. Telephone polls tend to show higher support for “Remain” than online polls (higher spread).
* ☒ D. Online polls have a larger interquartile range (IQR) for the spread than telephone polls, indicating that they are more variable.
* ☒ E. Poll type introduces a bias that affects poll results.

#### Question 7: Calculate the confidence intervals of the spread combined across all polls in june\_polls, grouping by poll type.

Recall that to determine the standard error of the spread, you will need to double the standard error of the estimate.

Use this code (which determines the total sample size per poll type, gives each spread estimate a weight based on the poll’s sample size, and adds an estimate of p from the combined spread) to begin your analysis:

combined\_by\_type <- june\_polls %>%  
 group\_by(poll\_type) %>%  
 summarize(N = sum(samplesize),  
 spread = sum(spread\*samplesize)/N,  
 p\_hat = (spread + 1)/2)

#What is the lower bound of the 95% confidence interval for online voters?

combined\_by\_type <- june\_polls %>%  
 group\_by(poll\_type) %>%  
 summarize(N = sum(samplesize),  
 spread = sum(spread\*samplesize)/N,  
 p\_hat = (spread + 1)/2,  
 se\_spread = 2\*sqrt(p\_hat\*(1-p\_hat)/N),  
 spread\_lower = spread - qnorm(.975)\*se\_spread,  
 spread\_upper = spread + qnorm(.975)\*se\_spread)  
combined\_by\_type %>%  
 filter(poll\_type == "Online") %>%  
 pull(spread\_lower)

## [1] -0.0165

What is the upper bound of the 95% confidence interval for online voters?

combined\_by\_type %>%  
 filter(poll\_type == "Online") %>%  
 pull(spread\_upper)

## [1] 0.00165

#### Question 8: Interpret the confidence intervals for the combined spreads for each poll type calculated in the previous problem.

Which of the following are TRUE about the confidence intervals of the combined spreads for different poll types?

Select ALL correct answers.

* ☒ A. Neither set of combined polls makes a prediction about the outcome of the Brexit referendum (a prediction is possible if a confidence interval does not cover 0).
* ☐ B. The confidence interval for online polls is larger than the confidence interval for telephone polls.
* ☒ C. The confidence interval for telephone polls is covers more positive values than the confidence interval for online polls.
* ☐ D. The confidence intervals for different poll types do not overlap.
* ☒ E. Neither confidence interval covers the true value of 𝑑=−0.038.

## Brexit poll analysis - Part 3

#### Question 9: Define brexit\_hit, with the following code, which computes the confidence intervals for all Brexit polls in 2016 and then calculates whether the confidence interval covers the actual value of the spread :

brexit\_hit <- brexit\_polls %>%  
 mutate(p\_hat = (spread + 1)/2,  
 se\_spread = 2\*sqrt(p\_hat\*(1-p\_hat)/samplesize),  
 spread\_lower = spread - qnorm(.975)\*se\_spread,  
 spread\_upper = spread + qnorm(.975)\*se\_spread,  
 hit = spread\_lower < d & spread\_upper > d) %>%  
 select(poll\_type, hit)

Use brexit\_hit to make a two-by-two table of poll type and hit status. Then use the chisq.test() function to perform a chi-squared test to determine whether the difference in hit rate is significant.

What is the p-value of the chi-squared test comparing the hit rate of online and telephone polls?

brexit\_chisq <- table(brexit\_hit$poll\_type, brexit\_hit$hit)  
chisq.test(brexit\_chisq)$p.value

## [1] 0.00101

Determine which poll type has a higher probability of producing a confidence interval that covers the correct value of the spread. Also determine whether this difference is statistically significant at a p-value cutoff of 0.05.

# online > telephone  
hit\_rate <- brexit\_hit %>%  
 group\_by(poll\_type) %>%  
 summarize(avg = mean(hit))  
hit\_rate$avg[hit\_rate$poll\_type == "Online"] > hit\_rate$avg[hit\_rate$poll\_type == "Telephone"]

## [1] TRUE

# statistically significant  
chisq.test(brexit\_chisq)$p.value < 0.05

## [1] TRUE

Which of the following is true?

* ☒ A. Online polls are more likely to cover the correct value of the spread and this difference is statistically significant.
* ☐ B. Online polls are more likely to cover the correct value of the spread, but this difference is not statistically significant.
* ☐ C. Telephone polls are more likely to cover the correct value of the spread and this difference is statistically significant.
* ☐ D. Telephone polls are more likely to cover the correct value of the spread, but this difference is not statistically significant.

#### Question 10: Use the two-by-two table constructed in the previous exercise to calculate the odds ratio between the hit rate of online and telephone polls to determine the magnitude of the difference in performance between the poll types.

Calculate the odds that an online poll generates a confidence interval that covers the actual value of the spread.

# from previous question  
brexit\_chisq <- table(brexit\_hit$poll\_type, brexit\_hit$hit)  
  
# convert to data frame  
chisq\_df <- as.data.frame(brexit\_chisq)  
  
online\_true <- chisq\_df$Freq[chisq\_df$Var1 == "Online" & chisq\_df$Var2 == "TRUE"]  
online\_false <- chisq\_df$Freq[chisq\_df$Var1 == "Online" & chisq\_df$Var2 == "FALSE"]  
  
online\_odds <- online\_true/online\_false  
online\_odds

## [1] 1.3

Calculate the odds that a telephone poll generates a confidence interval that covers the actual value of the spread.

phone\_true <- chisq\_df$Freq[chisq\_df$Var1 == "Telephone" & chisq\_df$Var2 == "TRUE"]  
phone\_false <- chisq\_df$Freq[chisq\_df$Var1 == "Telephone" & chisq\_df$Var2 == "FALSE"]  
  
phone\_odds <- phone\_true/phone\_false  
phone\_odds

## [1] 0.312

Calculate the odds ratio to determine how many times larger the odds are for online polls to hit versus telephone polls.

online\_odds/phone\_odds

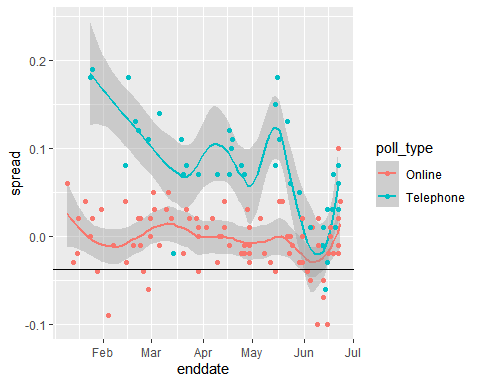
## [1] 4.15

#### Question 11: Use brexit\_polls to make a plot of the spread (spread) over time (enddate) colored by poll type (poll\_type).

Use geom\_smooth() with method = "loess" to plot smooth curves with a span of 0.4. Include the individual data points colored by poll type. Add a horizontal line indicating the final value of .

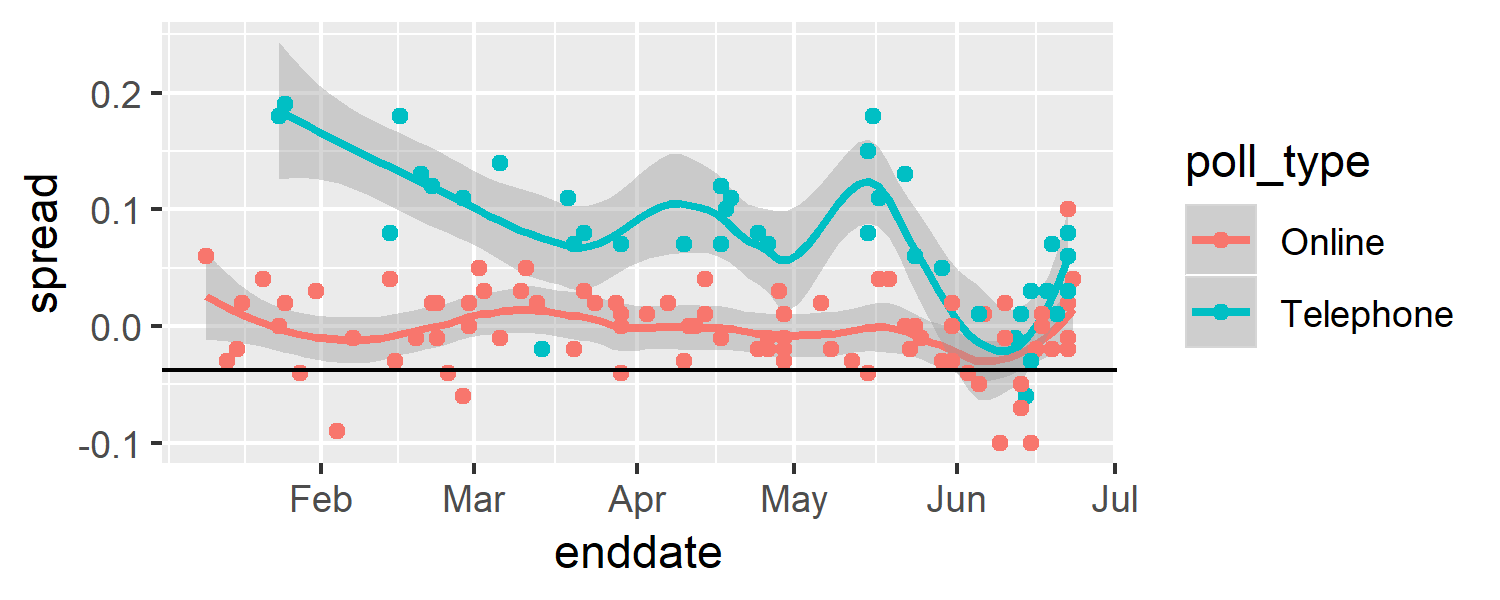
brexit\_polls %>%  
 ggplot(aes(enddate, spread, color = poll\_type)) +  
 geom\_smooth(method = "loess", span = 0.4) +  
 geom\_point() +  
 geom\_hline(aes(yintercept = -.038))

## `geom\_smooth()` using formula 'y ~ x'



Which of the following plots is correct?

* ☒ B.



Plotting spread over time

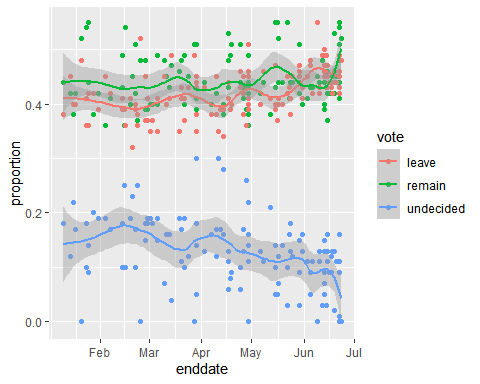
#### Question 12: Use the following code to create the object brexit\_long, which has a column vote containing the three possible votes on a Brexit poll (“remain”, “leave”, “undecided”) and a column proportion containing the raw proportion choosing that vote option on the given poll:

brexit\_long <- brexit\_polls %>%  
 gather(vote, proportion, "remain":"undecided") %>%  
 mutate(vote = factor(vote))

Make a graph of proportion over time colored by vote. Add a smooth trendline with geom\_smooth() and method = "loess" with a span of 0.3.

brexit\_long %>%  
 ggplot(aes(enddate, proportion, col=vote)) +  
 geom\_point() +  
 geom\_smooth(method = "loess", span=0.3)

## `geom\_smooth()` using formula 'y ~ x'



Which of the following are TRUE?

* ☒ A. The percentage of undecided voters declines over time but is still around 10% throughout June.
* ☒ B. Over most of the date range, the confidence bands for “Leave” and “Remain” overlap.
* ☒ C. Over most of the date range, the confidence bands for “Leave” and “Remain” are below 50%.
* ☒ D. In the first half of June, “Leave” was polling higher than “Remain”, although this difference was within the confidence intervals.
* ☐ E. At the time of the election in late June, the percentage voting “Leave” is trending upwards.