# Inference and Modeling

## Section 1: Parameters and Estimates

## Sampling Model 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 p. The proportion of red beads in the urn is 1−p and the spread is 2p−1.
* 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 n from the urn described in the video:

library(tidyverse)

library(dslabs)

take\_poll(25) # draw 25 beads

## The Sample Average

### 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 p. From the estimate of p, we can easily calculate an estimate of the spread, 2p−1.
* Consider the random variable X that is 1 if a blue bead is chosen and 0 if a red bead is chosen. The proportion of blue beads in N draws is the average of the draws X1,...,XN.
* X¯ is the sample average. In statistics, a bar on top of a symbol denotes the average. X¯ is a random variable because it is the average of random draws - each time we take a sample, X¯ is different.

X¯=X1+X2+...+XNN

* The number of blue beads drawn in N draws, NX¯, is N times the proportion of values in the urn. However, we do not know the true proportion: we are trying to estimate this parameter p.

## Polling versus Forecasting

### Key points

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

## Properties of Our Estimate

### Key points

* When interpreting values of X¯, it is important to remember that X¯ is a random variable with an expected value and standard error that represents the sample proportion of positive events.
* The expected value of X¯ is the parameter of interest p. This follows from the fact that X¯ is the sum of independent draws of a random variable time a constant 1/N.

E(X¯) =p

* As the number of draws N increases, the standard error of our estimate X¯ decreases. The standard error of the average of X¯ over N draws is:

SE(X¯) =√(p(1−p)/N)

* + In theory, we can get more accurate estimates of p by increasing N. In practice, there are limits on the size of N 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 E(S) and standard error of the sum of draws SE(S).

E(S)=Np

SE(S)= √(Np(1−p))

## Section 2: The Central Limit Theorem in Practice

## The Central Limit Theorem in Practice

### Key points

* Because X¯ is the sum of random draws divided by a constant, the distribution of X¯ is approximately normal.
* We can convert X¯ to a standard normal random variable Z:

Z=X¯−E(X¯)SE(X¯)

* The probability that X¯ is within .01 of the actual value of p is:

Pr(Z≤.01/p(1−p)/N−−−−−−−−−−√)−Pr(Z≤−.01/p(1−p)/N−−−−−−−−−−√)

* The Central Limit Theorem (CLT) still works if X¯ is used in place of p. This is called a plug-in estimate. Hats over values denote estimates. Therefore:

SE^(X¯)=X¯(1−X¯)/N−−−−−−−−−−−√

* Using the CLT, the probability that X¯ is within .01 of the actual value of p is:

Pr(Z≤.01/X¯(1−X¯)/N−−−−−−−−−−−√)−Pr(Z≤−.01/X¯(1−X¯)/N−−−−−−−−−−−√)

### Code: Computing the probability of **X¯** being within .01 of **p**

X\_hat <- 0.48

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

pnorm(0.01/se) - pnorm(-0.01/se)

## Margin of Error

### Key points

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

## A Monte Carlo Simulation for the CLT

### Key points

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

library(tidyverse)

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

### Key points

* The spread between two outcomes with probabilities p and 1−p is 2p−1.
* The expected value of the spread is 2X¯−1.
* The standard error of the spread is 2SE^(X¯).
* The margin of error of the spread is 2 times the margin of error of X¯.

## 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

library(tidyverse)

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()

## Section 3: Confidence Interval and p-Value

## Confidence Intervals

### Key points

* We can use statistical theory to compute the probability that a given interval contains the true parameter p.
* 95% confidence intervals are intervals constructed to have a 95% chance of including p. 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 z for which Pr(−z≤Z≤z) equals the desired confidence. For example, a 99% confidence interval requires calculating z for Pr(−z≤Z≤z)=0.99.
* For a confidence interval of size q, we solve for z=1−1−q2.
* 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")

### 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

### Code: Solving for **z** 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

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

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

## A Monte Carlo Simulation for Confidence Intervals

### Key points

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

### 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)

## The Correct Language

### Key points

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

## 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)

## 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 p=0.5.
* 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 z that corresponds to the observed result, and then use that z 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))

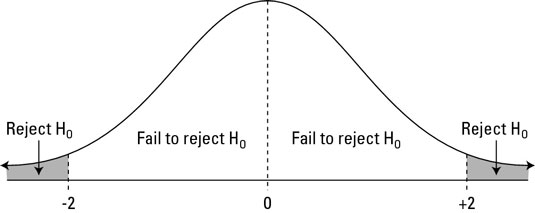
## Another Explanation of p-Values

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 ∣z∣≥2. ∣z∣≥2

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 z=0 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.



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 μ=0 and standard deviation of σ=1. 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.

## Section 4: Statistical Model

## 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

### 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)

round(moe\*100, 1)

## Pollsters and Multilevel Models

### 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

### 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 p the proportion voting for Clinton and 1−p the proportion voting for Trump. We are interested in the spread d=2p−1.
* Poll results are a random normal variable with expected value of the spread d and standard error 2p(1−p)/N−−−−−−−−−−√.
* 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

library(dslabs)

data(polls\_us\_election\_2016)

names(polls\_us\_election\_2016)

# 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())

# 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)))

## Data-Driven Models

### 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 d=2p−1.
* Our new standard error σ now factors in pollster-to-pollster variability. It can no longer be calculated from p or d and is an unknown parameter.
* The central limit theorem still works to estimate the sample average of many polls X1,...,XN because the average of the sum of many random variables is a normally distributed random variable with expected value d and standard error σ/N−−√.
* We can estimate the unobserved σ 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)

## Section 5: Bayesian Statistics

## Bayesian Statistics

### Key points

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

## 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:

Pr(A∣B)=Pr(B∣A)Pr(A)Pr(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, D=0 indicates no disease, and D=1 indicates the disease is present.

Probability of having the disease given a positive test: Pr(D=1∣+)

99% test accuracy when disease is present: Pr(+∣D=1)=0.99

99% test accuracy when disease is absent: Pr(−∣D=0)=0.99

Rate of cystic fibrosis: Pr(D=1)=0.00025

Bayes' theorem can be applied like this:

Pr(D=1∣+)=Pr(+∣D=1)⋅Pr(D=1)Pr(+)

Pr(D=1∣+)=Pr(+∣D=1)⋅Pr(D=1)Pr(+∣D=1)⋅Pr(D=1)+Pr(+∣D=0)⋅Pr(D=0)

Substituting known values, we obtain:

0.99⋅0.000250.99⋅0.00025+0.01⋅0.99975=0.02

### 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)

## 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

### 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:

p∼N(μ,τ) describes player-to-player variability in natural ability to hit, which has a mean μ and standard deviation τ.

Y∣p∼N(p,σ) describes a player's observed batting average given their ability p, which has a mean p and standard deviation σ=√(p(1−p)/N). 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 p given that we have observed data Y.
* By the continuous version of Bayes' rule, the expected value of the posterior distribution p given Y=y is a weighted average between the prior mean μ and the observed data Y:

E(p∣y)=Bμ+(1−B)Y     where     B=σ2σ2+τ2

* The standard error of the posterior distribution SE(p∣Y)2 is 11/σ2+1/τ2. 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 σ is large, B is close to 1 and our prediction of p shrinks towards the mean (\mu). When σ is small, B is close to 0 and our prediction of p is more weighted towards the observed data Y.

## Section 6: Election forecasting

## Election Forecasting

### Key points

* In our model:
  + - The spread d∼N(μ,τ) describes our best guess in the absence of polling data. We set μ=0 and τ=0.035 using historical data.
    - The average of observed data X¯∣d∼N(d,σ) 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 E(p∣Y) and SE(p∣Y).
* Given an estimate of E(p∣Y) and SE(p∣Y), we can use pnorm to compute the probability that d>0.
* 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 from previous videos defines the results object used for empirical Bayes election forecasting.

library(tidyverse)

library(dslabs)

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

posterior\_se

# 95% credible interval

posterior\_mean + c(-1.96, 1.96)\*posterior\_se

# probability of d > 0

1 - pnorm(0, posterior\_mean, posterior\_se)

## Mathematical Representations of Models

### Key points

* If we collect several polls with measured spreads X1,...,Xj with a sample size of N, these random variables have expected value d and standard error 2p(1−p)/N−−−−−−−−−−√.
* We represent each measurement as Xi,j=d+b+hi+ϵi,j where:
  + - The index i represents the different pollsters
    - The index j represents the different polls
    - Xi,j is the jth poll by the ith pollster
    - d is the actual spread of the election
    - b is the general bias affecting all pollsters
    - hi represents the house effect for the ith pollster
    - ϵi,j represents the random error associated with the i,jth poll.
* The sample average is now X¯=d+b+1N∑i=1NXi with standard deviation SE(X¯)=σ2/N+σ2b−−−−−−−−−√.
* The standard error of the general bias σb does not get reduced by averaging multiple polls, which increases the variability of our final estimate.

### Code: Simulated data with **Xj=d+ϵj**

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 **Xi,j=d+ϵi,j**

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 **Xi,j=d+hi+ϵi,j**

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 **d>0** with general bias

Note that sigma now includes an estimate of the variability due to general bias σb=.025.

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)

## 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 d. 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:

library(tidyverse)

library(dslabs)

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

head(results\_us\_election\_2016)

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

### 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))

# 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)

# 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

Note there is a small error in the video code: B should be defined as sigma^2/(sigma^2 + tau^2).

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))

### 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

# histogram of outcomes

data.frame(clintonEV) %>%

ggplot(aes(clintonEV)) +

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

## Forecasting

### Key points

* In poll results, p 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 bt to model the time effect.
* Pollsters also try to estimate f(t), the trend of p given time t using a model like:

Yi,j,t=d+b+hj+bt+f(t)+ϵi,j,t

* 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

# 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)

### 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))

## The t-Distribution

### Key points

* In models where we must estimate two parameters, p and σ, 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 σ.
* Given s as an estimate of σ, then Z=X¯−ds/N√ follows a t-distribution with N−1 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)

# quantile from t-distribution versus normal distribution

qt(0.975, 14) # 14 = nrow(one\_poll\_per\_pollster) - 1

qnorm(0.975)

## Section 7: Association Tests

## Association Tests

### 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

library(tidyverse)

library(dslabs)

data(research\_funding\_rates)

research\_funding\_rates

# 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)

# 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))

### 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

# p-value calculation with Fisher's Exact Test

fisher.test(tab, alternative = "greater")

## Chi-Squared Tests

### 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

# 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

# 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))

# chi-squared test

chisq\_test <- two\_by\_two %>%

select(-awarded) %>%

nbsp; chisq.test()

chisq\_test$p.value

### 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

### 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()