

ANNOUNCEMENT

See course website for information
on test & office hours during
Reading Week and week of test.

STA130H1S - Class #6

Inferential Thinking Part 3: Estimation

Prof. A. Gibbs

February 12, 2018

Today

Big idea:

We estimate a characteristic of a population from incomplete, imperfect observed data.

What is a range of plausible values for what it could actually be?

Important concepts:

1. Percentiles
2. Population parameters and sample statistics to estimate them
3. Sampling distribution
4. Bootstrap sampling distribution
5. Confidence intervals

Recommended reading:

Sections 7.1, 7.2, 7.3 of *Modern Data Science with R*

(You can safely ignore any mention of standard error or standard deviation.)

Code in textbook relies on mosaic
package
We won't use it

Some necessary background:
Percentiles

Percentiles

We are often interested in the values of a numerical variable after they've been sorted in increasing or decreasing order.

Definition of percentile for a numerical variable:

For p a number between 0 and 100, the p th percentile is the the smallest value that is at least as large as $p\%$ of all of the values.

Percentiles are calculated in R with the quantile function.

- When a percentile lies between two data values, there are various ways to interpolate between them to estimate the percentile.
- You can see the default method for how R does this in the help for the `quantile` function, but you are not responsible for knowing these details.
- You are responsible for interpreting a percentile that has been calculated in R, but not for calculating it yourself.

A question:

Suppose your score on a test is the 95th percentile of the class. Did you do well or not well compared to the rest of the class?



Example:

Scores on a test for a class of 10 students: 70, 55, 90, 67, 76, 92, 71, 82, 85, 79

Sorted scores: 55, 67, 70, 71, 76, 79, 82, 85, 90, 92

marks <- c(70, 55, 90, 67, 76, 92, 71, 82, 85, 79) # the 50th percentile
quantile(marks, 0.5) # the 50th percentile

50%
77.5

quantile(marks, 0.8) # the 80th percentile

80%
86

Interpolated

```
quantile(marks, c(0.5, 0.8)) # the 50th and the 80th percentile
```

```
## 50% 80%
## 77.5 86.0
```

Vector of quantiles to get more than one

Some particular percentiles:

- **median**: the 50th percentile
- **first quartile**: the 25th percentile
- **third quartile**: the 75th percentile

Inferential Thinking for Estimation

Statistical Inference

- Imagine we have a *real world* where we observe data, and a *theoretical world* (a population or scientific model) that we want to make conclusions about.
- Inference connects what we observe in the real world to what we can say about the theoretical world.
- *Last two weeks:* The null hypothesis gave us a model for the theoretical world.
- *Today:* No hypotheses that presume something about the theoretical world.

REAL WORLD

observe

sample
data

statistic

THEORETICAL WORLD

??

population

conclusions \Rightarrow

estimate \Rightarrow parameter

Today: What is a range of plausible values for the parameter, given the data.

Populations and samples

- A **population** is a complete collection of individuals that we are interested in.
- A **sample** is a subset of a population.
- We want to understand something about the population (our theoretical world).
- We can't measure every individual in the population because we don't have the time or the money.
- So we measure a sample (the real world).
- A good sample is chosen randomly to ~~ensure~~ ensure it is representative of the population.

Parameters and statistics

population

- Recall: A **parameter** is a numerical value associated with the theoretical world.
- If we have the relevant data for the entire population, we can simply calculate the parameter.
- In most situations we only have data collected from a random sample.
- We estimate the value of a parameter from the data.
- A **statistic** is an estimate of the parameter, calculated from the data.
- Every random sample drawn from the population will give a different value of the statistic.

Sample

A question:

Since the values of the estimate vary from sample to sample, what are the possible values this estimate might have been?

→ Sampling distributions

Sampling distributions

Sampling from a population

- To demonstrate the idea of *sampling distribution*, we'll consider the unrealistic scenario where we are examining samples of observations from a population and we have all the data in the population.
- This artificial situation allows us to examine what possible values we could get for an estimate of a parameter from various possible samples.

Example:

2013 flights from New York to San Francisco

The population:

All flights leaving New York for San Francisco (airport code: SFO) in 2013.

We'll store these in a data frame called `SF`.

We're interested in the numerical variable `arr_delay`.

```
library(tidyverse)
library(nycflights13)
SF <- flights %>% filter(dest == "SFO", !is.na(arr_delay))
```

population
I'm
interested in

get rid of missing values

Some values calculated from our population

```
SF %>% summarize(mean_delay=mean(arr_delay), median_delay=median(arr_delay),
max_delay=max(arr_delay), perc98_delay=quantile(arr_delay, 0.98))
```

98th percentile

```
## # A tibble: 1 x 4
##   mean_delay median_delay max_delay perc98_delay
##       <dbl>        <dbl>      <dbl>        <dbl>
## 1     2.672892         -8      1007        153
```

Are these *parameters* or *statistics*?

Have all the information
the population

Samples of size 25

Now suppose we only have a random sample of 25 observations (25 flights) from our population.

The function `sample_n` in `dplyr` can be used to draw samples.

The default is sampling without replacement – so we'll get a sample of 25 different flights.

```
sample25 <- SF %>% sample_n(size = 25) # sample of 25 flights from our population
```

Some values calculated from our sample

```
sample25 %>% summarize(mean_delay=mean(arr_delay), median_delay=median(arr_delay),  
max_delay=max(arr_delay), perc98_delay=quantile(arr_delay, 0.98))
```

```
## # A tibble: 1 x 4  
##   mean_delay median_delay max_delay perc98_delay  
##       <dbl>        <dbl>      <dbl>        <dbl>  
## 1     -4.32          -7        50       44.72
```

Are these *parameters* or *statistics*?

based on a sample of data

Another sample of size 25

```
sample25 <- SF %>% sample_n(size = 25)

sample25 %>% summarize(mean_delay=mean(arr_delay), median_delay=median(arr_delay),
                         max_delay=max(arr_delay), perc98_delay=quantile(arr_delay, 0.98))
```

```
## # A tibble: 1 x 4
##   mean_delay median_delay max_delay perc98_delay
##       <dbl>        <dbl>      <dbl>        <dbl>
## 1        -2          -5       101       89.96
```

Values on the extremes vary a lot from sample to sample

And another sample of size 25

```
sample25 <- SF %>% sample_n(size = 25)

sample25 %>% summarize(mean_delay=mean(arr_delay), median_delay=median(arr_delay),
                         max_delay=max(arr_delay), perc98_delay=quantile(arr_delay, 0.98))
```

```
## # A tibble: 1 x 4
##   mean_delay median_delay max_delay perc98_delay
##       <dbl>        <dbl>      <dbl>        <dbl>
## 1     -1.8          -9        53       49.64
```



Very from sample to sample

Sampling distribution of the mean

The *sampling distribution* of the mean of arr_delay is the distribution of all of the values that mean_delay can be for random samples of size 25.

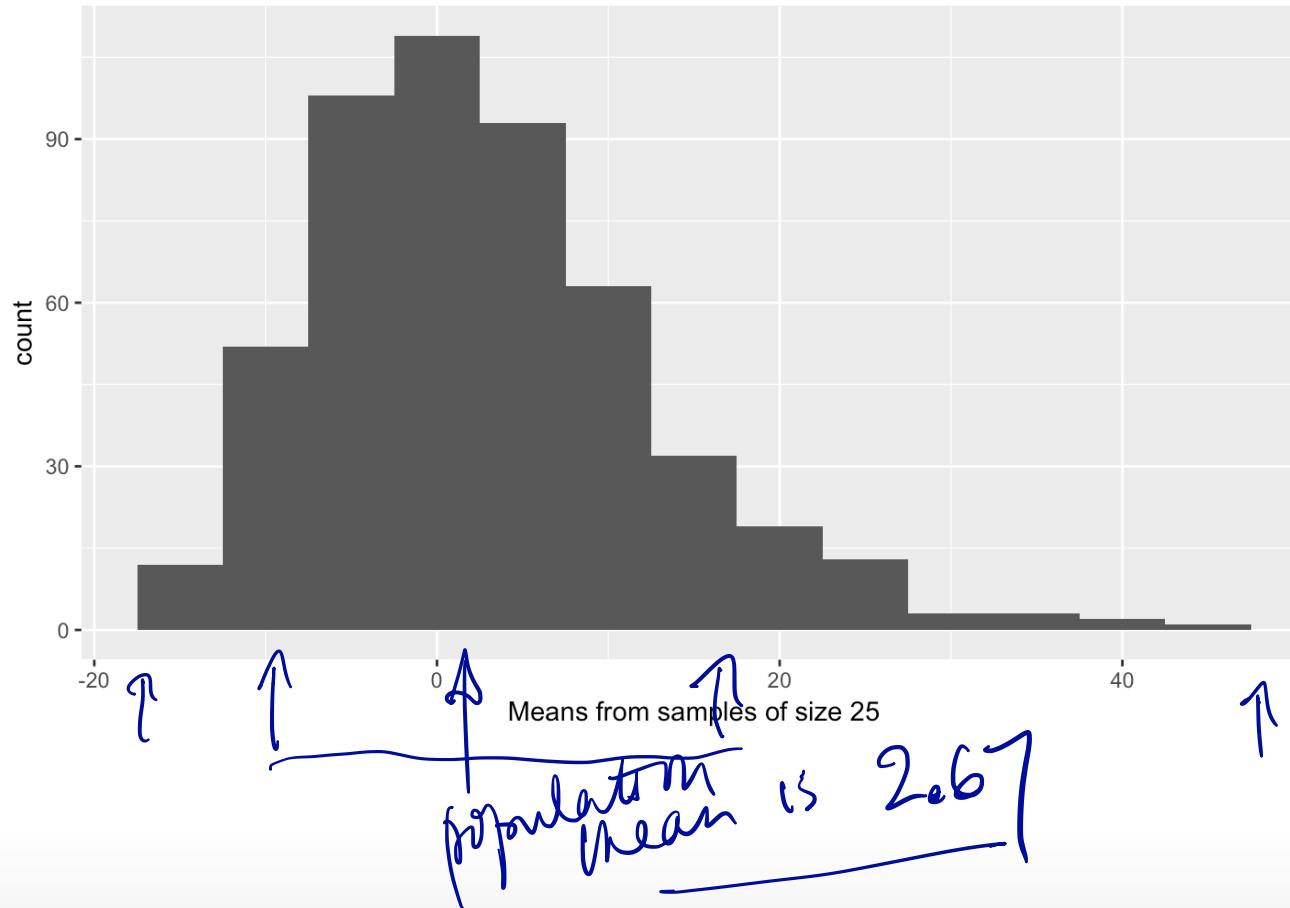
To explore the sampling distribution, let's look at 500 values of mean_delay, calculated from 500 possible random samples of size 25.

```
sample_means <- rep(NA, 500) # where we'll store the means  
for (i in 1:500) do 500 times {  
  sample25 <- SF %>% sample_n(size = 25)  
  sample_means[i] <- as.numeric(sample25 %>% summarize(mean(arr_delay)))  
}  
  
sample_means <- data_frame(mean_delay=sample_means)
```

a sample of size 25 from the population
— calculate mean from the sample data

```
ggplot(sample_means, aes(x=mean_delay)) + geom_histogram(binwidth=5) +
  labs(x="Means from samples of size 25",
       title="Sampling distribution for the mean of arr_delay")
```

Sampling distribution for the mean of arr_delay



The sampling distribution of the mean of arr_delay

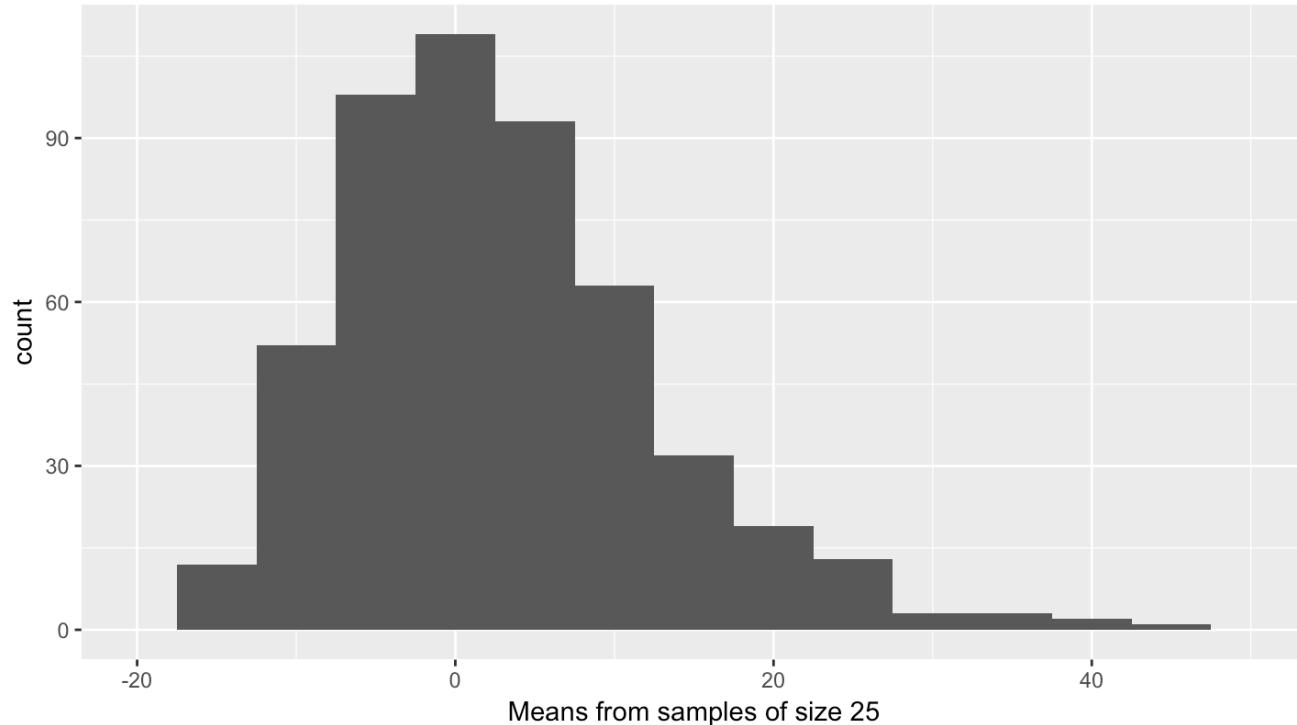
For a *sample size* of 25 observations, the sampling distribution of the mean of arr_delay:

- Has one mode
- The mode is near the mean for the population (2.67)
- Is slightly right-skewed
- Values range from about -20 to 50 but most values are between -10 and 15

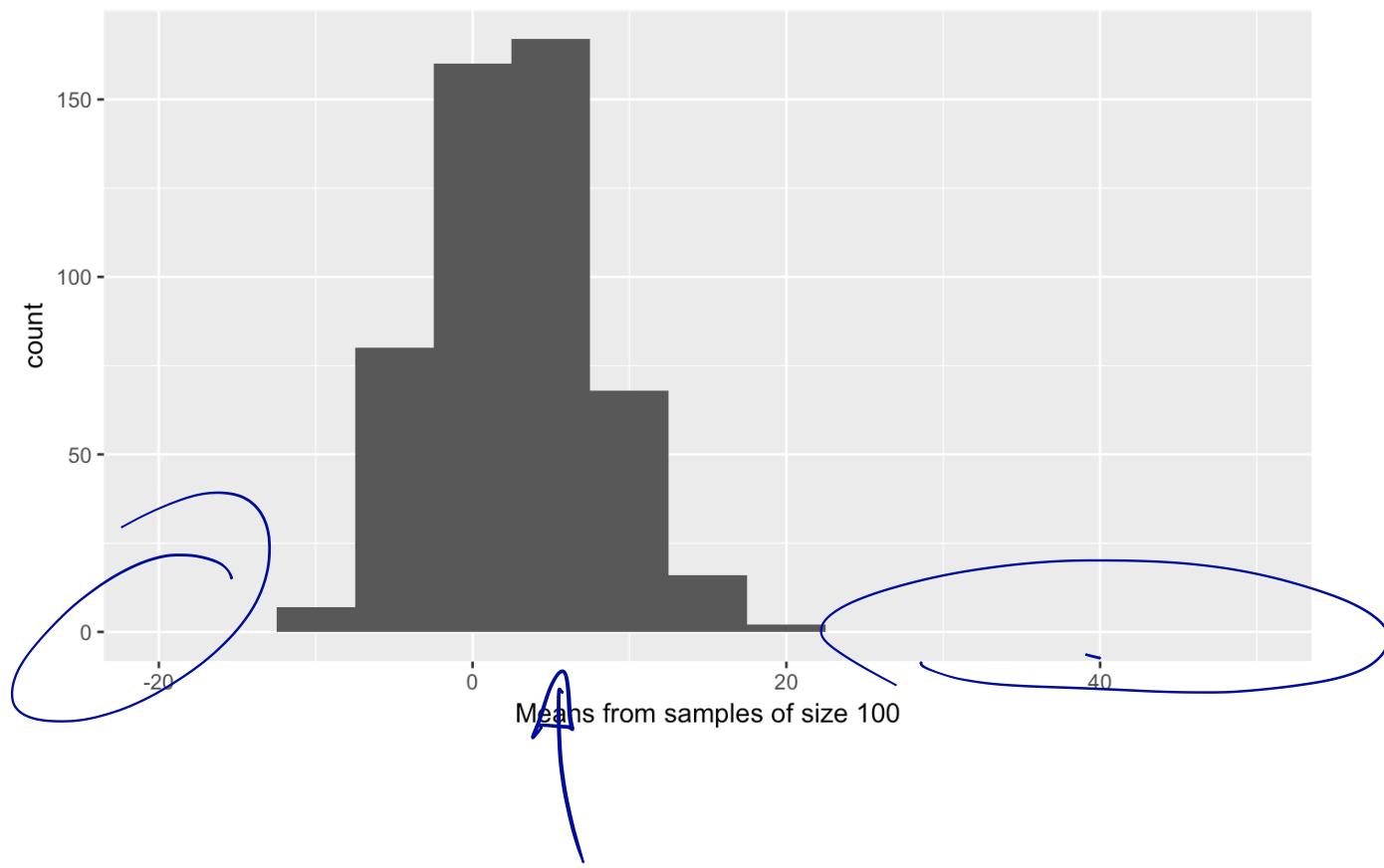
What if our sample size was 100?

```
sample_means100 <- rep(NA, 500) # where we'll store the means
for (i in 1:500)
{
  sample100 <- SF %>% sample_n(size = 100)
  sample_means100[i] <- as.numeric(sample100 %>% summarize(mean(arr_delay)))
}
sample_means100 <- data_frame(mean_delay=sample_means100)
```

Sample size 25



Sample size 100



How the sampling distribution of the mean differs with sample size

Comparing the sampling distribution of the mean of `arr_delay` for samples of size 25 and size 100:

- Both sampling distributions have a single mode at the same value (approximately). *which is the population mean*
- There is less variability in the values of the mean for samples of size 100 than for samples of size 25.
- The distribution of the mean for samples of size 100 is less right-skewed (more symmetric) than the distribution of the mean for samples of size 25.

A reality check

What if we only have sample data from one sample and not the population?

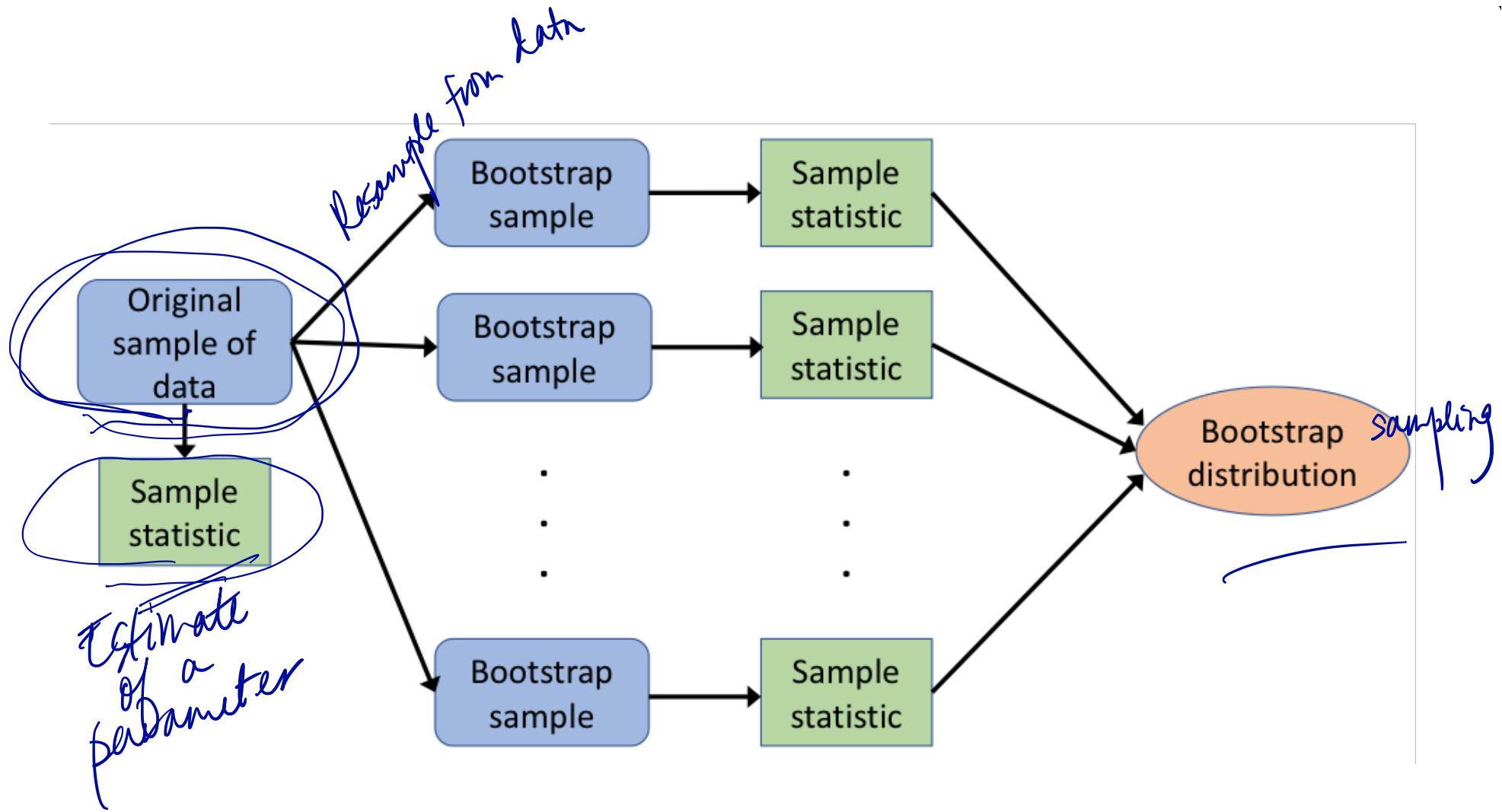
The Bootstrap

Resampling from the sample

- Use resampling in the real world situation where all we have is a dataset that is one sample from the population.
- Treat the observed sample of data as a good representation of the population.
- Resample from the observed data: sample *with replacement*, with samples the same size as the observed data. These are **bootstrap samples**.
- If the data resemble the population, the bootstrap samples will also resemble the population.
- Note that the bootstrap doesn't create new data. It works when our sample data is a reasonable representation of the population.

The bootstrap sampling distribution

- For each bootstrap sample, a statistic can be calculated to estimate a parameter from the population.
- The distribution of the values of the statistic for all bootstrap samples is the **bootstrap sampling distribution**. It gives us an estimate of the sampling distribution of the statistic.



Suppose we do not observe the population.
We have observed a sample of size 200.

→ need larger sample size to use bootstrap.

Here it is:

```
observed_data <- SF %>% sample_n(size = 200, replace = FALSE)
```

We're still interested in the mean of arr_delay.

Here is the mean of arr_delay for our observed data:

```
observed_mean <- as.numeric(observed_data %>% summarize(mean(arr_delay)))  
observed_mean
```

[1] 0.285

Is this a *parameter* or a *statistic*?

A bootstrap sample from our data

size of data

```
boot_samp <- observed_data %>% sample_n(size = 200, replace=TRUE)  
boot_samp %>% summarize(mean_delay=mean(arr_delay))
```

```
## # A tibble: 1 x 1  
##   mean_delay  
##   <dbl>  
## 1 -5.07
```

Another bootstrap sample

```
boot_samp <- observed_data %>% sample_n(size = 200, replace=TRUE)
```

```
boot_samp %>% summarize(mean_delay=mean(arr_delay))
```

```
## # A tibble: 1 x 1
##   mean_delay
##       <dbl>
## 1     3.255
```



5000 bootstrap samples

Typically need lots of replications when bootstrapping.

How many? Typically at least 1000.

As with all simulations, results vary.

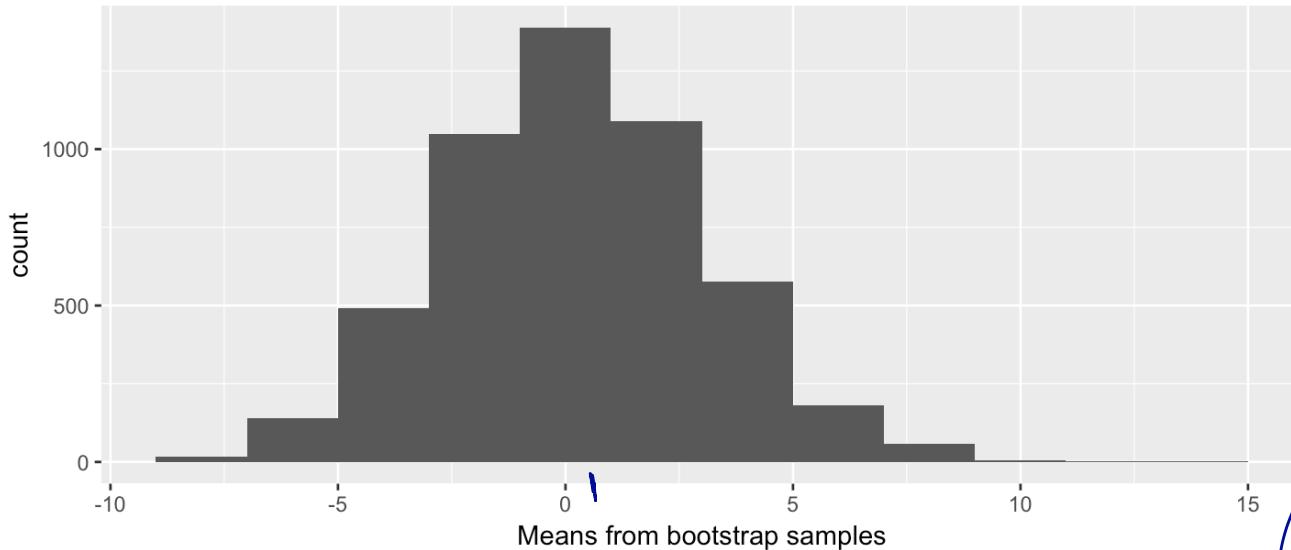
You can experiment with how many replications are needed to give stable estimates to the desired accuracy.

```
boot_means <- rep(NA, 5000) # where we'll store the means  
for (i in 1:5000)  
{  
  boot_samp <- observed_data %>% sample_n(size = 200, replace=TRUE)  
  boot_means[i] <- as.numeric(boot_samp %>% summarize(mean(arr_delay)))  
}  
  
boot_means <- data_frame(mean_delay=boot_means)
```

one bootstrap sample
statistic for
the bootstrap sample

```
ggplot(boot_means, aes(x=mean_delay)) + geom_histogram(binwidth=2) +
  labs(x="Means from bootstrap samples",
       title="Bootstrap distribution for the mean of arr_delay")
```

Bootstrap distribution for the mean of arr_delay



population
mean: 2.67
(don't know
this now)

Mean for my
sample data:

0.285

estimate
of population
mean

Where is the centre of the distribution? Does this make sense?

Do the bootstrap estimates capture the population parameter?

Remember the value of the population mean:

```
population_mean <- SF %>% summarize(population_mean_delay=mean(arr_delay))  
population_mean
```

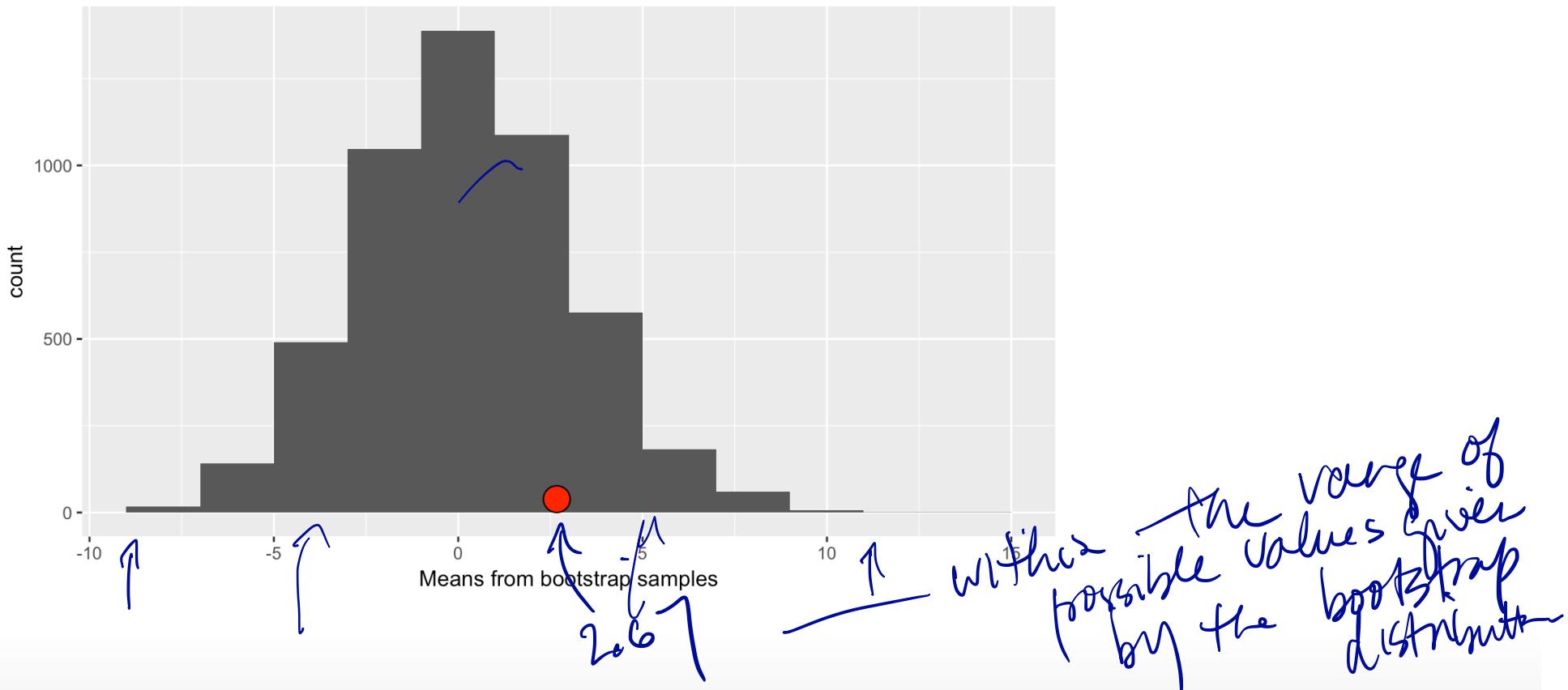
```
## # A tibble: 1 x 1  
##   population_mean_delay  
##       <dbl>  
## 1     2.672892
```

```

ggplot(boot_means, aes(x=mean_delay)) + geom_histogram(binwidth=2) +
  geom_dotplot(data=population_mean, aes(x=population_mean_delay), fill="red") +
  labs(x="Means from bootstrap samples",
       title="Bootstrap distribution for the mean of arr_delay")

```

Bootstrap distribution for the mean of arr_delay

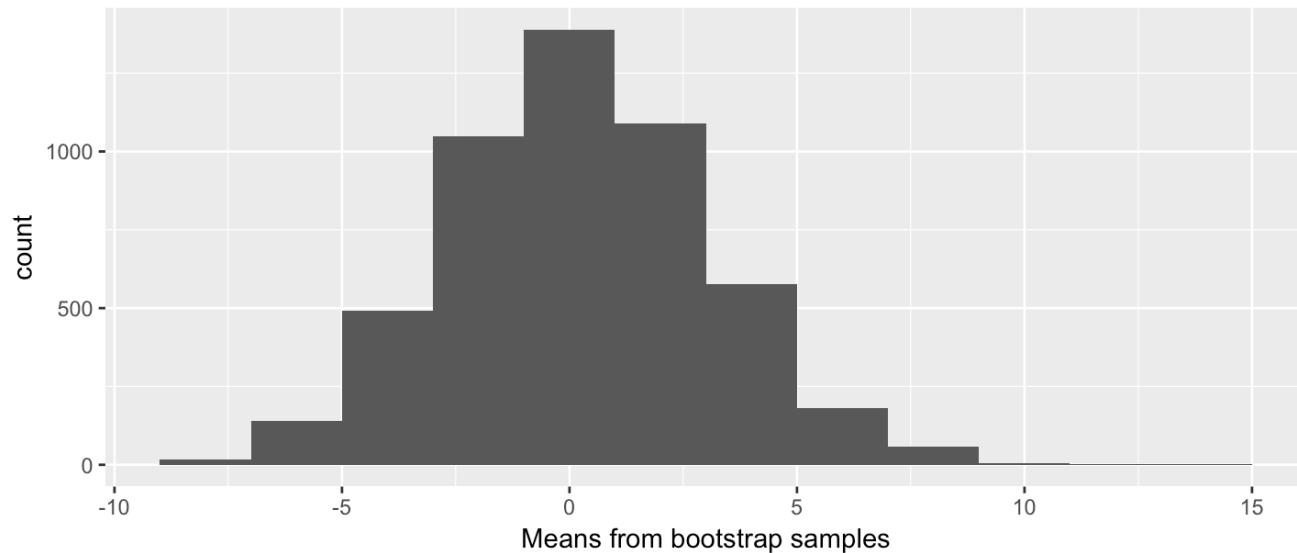


Based on the bootstrap distribution, what other population values of the mean might be plausible?

- In the real world we observe our sample of data, we can construct the bootstrap sampling distribution of the mean, and we don't know the population mean.
- We'd like to make inferences about the population, such as *what other population values of the mean might be plausible?*
- To answer this, look at the range of values that the bootstrap distribution covers, but exclude the values way out in the tails.
- Typically, we take the middle 95% of the bootstrap distribution of resampled means.

- These are values from the 2.5th percentile to the 97.5th percentile.


Bootstrap distribution for the mean of arr_delay



2.5th and 97.5th percentiles:

```
quantile(boot_means$mean_delay, c(0.025, 0.975))
```

```
##      2.5%    97.5%
## -5.14000  5.98525
```

Range of plausible values for the population (-5.14, 5.99)

Will this procedure always give an interval that captures the population mean?

Our interval that is the middle 95% of our bootstrap distribution is (-5.14, 5.99). It includes the population mean (2.673).

To see how often an interval calculated this way from a sample of size 200 would capture the population mean, we can take advantage of the situation here where we have the population and repeat this procedure many times. We can:

1. Randomly draw another data sample of size 200 from the population.
2. Find the bootstrap sampling distribution of the mean from 5000 replications of bootstrap samples of this new data.
3. Find the interval that is the middle 95% of the bootstrap distribution.
4. Repeat 1. to 3. 100 times.

Statistical theory says that these intervals should capture the population mean 95% of the time.

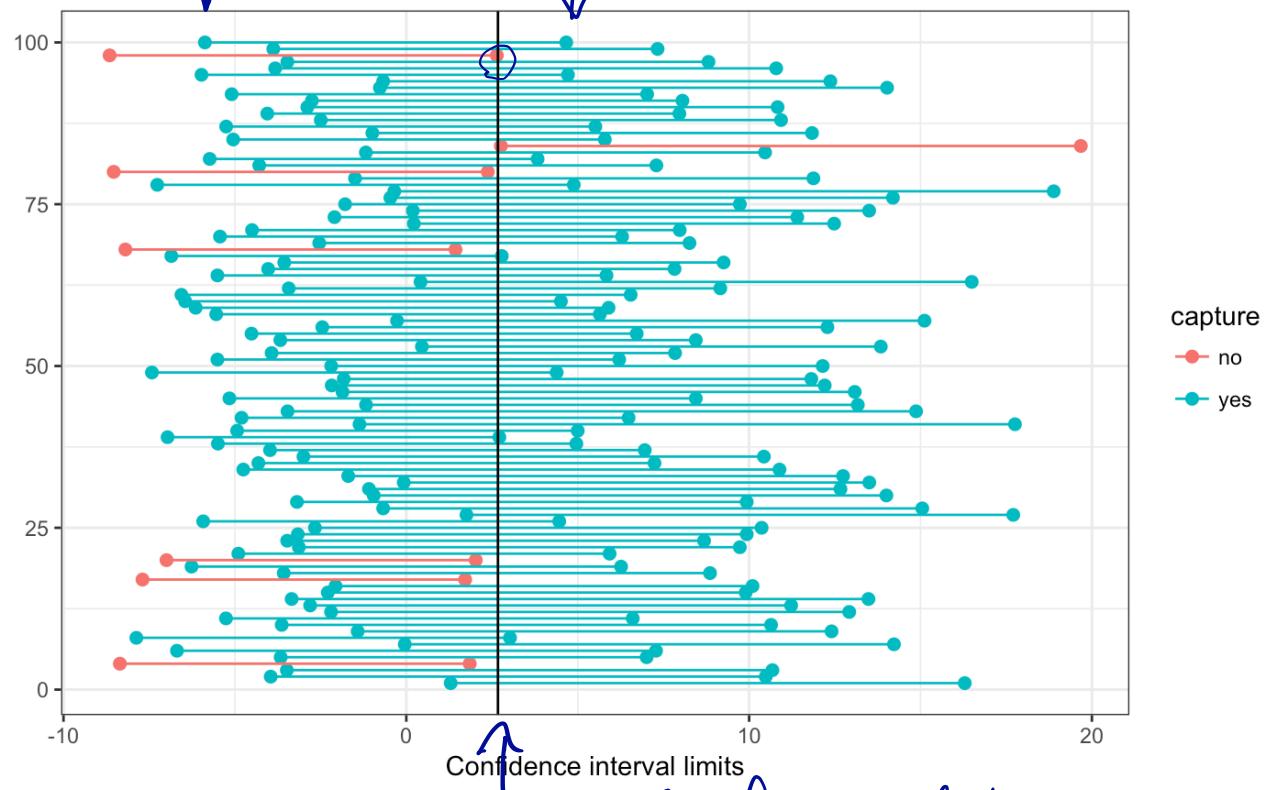
These are called 95% **confidence intervals** for the mean.

To see if this holds:

- Need to know the population mean
- Need to take a number of random samples, each representing a possible dataset
- Need to calculate bootstrap intervals for the mean for each dataset
- Need to check how many of these confidence intervals contain the population mean

Code to calculate 100 bootstrap confidence intervals for the mean of `arr_delay`, each calculated from a random sample from the population of size 200, is in the R markdown document for this lecture. Note that it takes a while to run.
Results are on the next slide...

100 bootstrap confidence intervals for the mean, each calculated from a random sample from the population of size 200



How many of the confidence intervals capture the population mean?

2.67 (population mean)

93 / 100

93%

- Each of these confidence intervals gives a range of plausible values for what our parameter might be. This range is based on the incomplete and imperfect information we have in each set of data.
- A "good" interval captures the population mean.
- Since our intervals are the middle 95% of the bootstrap sampling distribution of the mean, we expect that 95% will be "good".
- Sometimes, because of chance, our randomly sampled data leads to a confidence interval that does not capture the population mean. We expect that this will happen for ~~5%~~ of datasets.

5%

Confidence Intervals

What is a confidence interval?

A 95% **confidence interval** for a population parameter is calculated from sample data in such a way that the interval will include the parameter for 95% of possible samples.

95% is the **confidence level**.

90% and 99% confidence intervals are also common.

[Note: A comment in your textbook near the bottom of page 153 that you can ignore, along with all references to standard error and standard deviation: As taught in introductory statistics courses, often a 95% confidence interval is calculated from the mean and standard error of the sampling distribution.

You'll learn the theory behind this statement in your second year statistics courses.]

How to calculate a bootstrap confidence interval

1. Take a bootstrap sample of the data by sampling with replacement, the same number of observations as the original data.
2. For the bootstrap sample, calculate the statistic that estimates the parameter you are interested in.
3. Repeat steps 1. and 2. many times to get a distribution of bootstrap statistics.
4. A 95% confidence interval for the parameter is the middle 95% of values of the bootstrap statistics.

Example from Week 4:

Kissing the Right Way

- Güntürkün (2003) recorded the direction kissing couples tilted their heads.
- Of the 124 couples he observed, 80 turned their heads to the right.
- 64.5% of couples in Güntürkün's sample tilted their heads to the right.
- *Today:* Find a 95% confidence interval for the proportion of all couples who tilt their heads to the right when they kiss.

```
# Create a data frame
direction <- c( rep("right", 80), rep("left", 124-80) )
kissdata <- data_frame(direction)
```

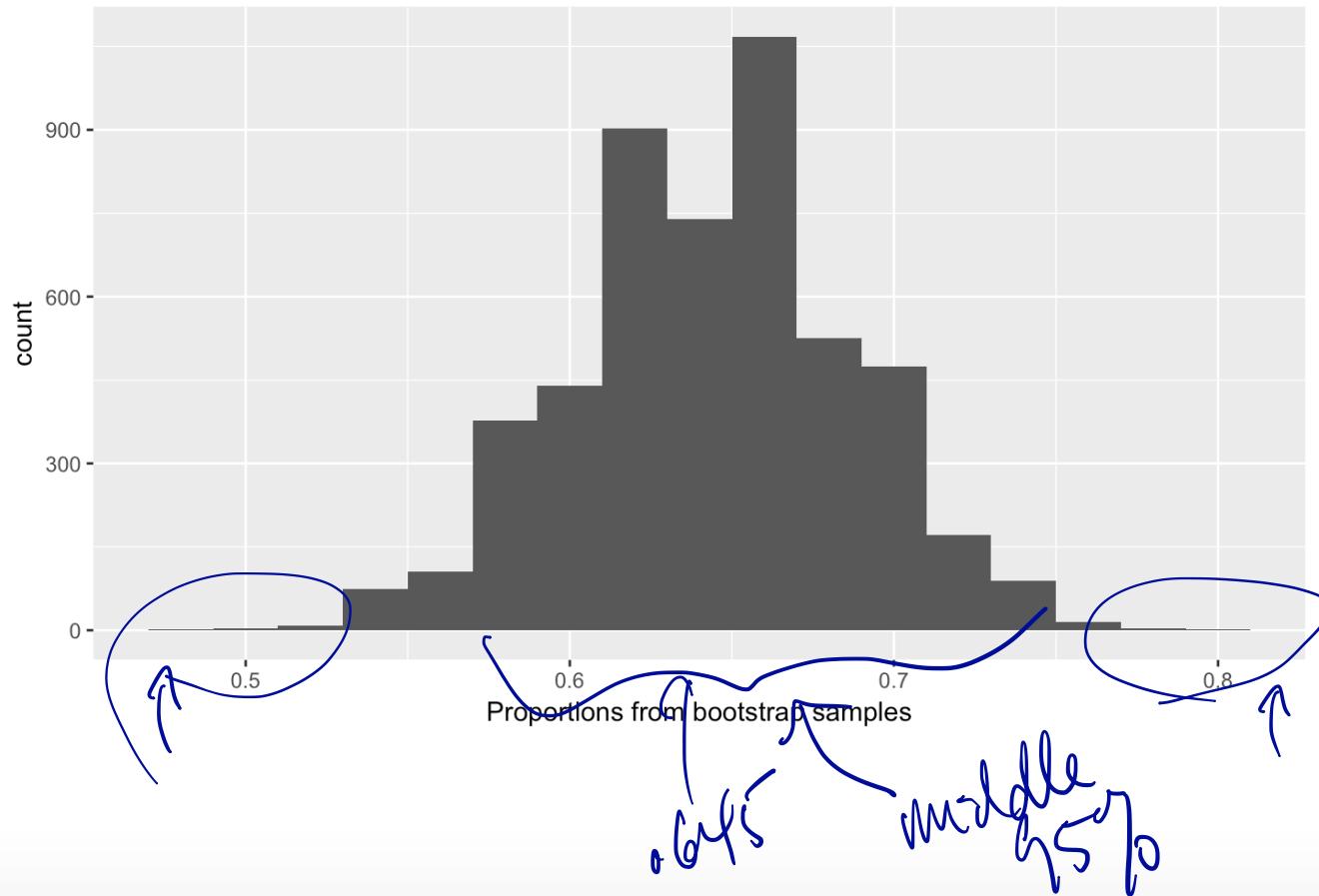
Bootstrap distribution for the proportion of couples who tilt their heads to the right when they kiss

```
boot_p <- rep(NA, 5000) # where we'll store the bootstrap proportions
for (i in 1:5000)
{
  boot_samp <- kissdata %>% sample_n(size = 124, replace=TRUE)
  boot_p[i] <- as.numeric(boot_samp %>% filter(direction == "right") %>%
    summarize(n()))/124
}
boot_p <- data_frame(boot_p)
```

→ proportion of "right" in bootstrap sample

```
ggplot(boot_p, aes(x=boot_p)) + geom_histogram(binwidth=0.02) +  
  labs(x="Proportions from bootstrap samples",  
    title="Bootstrap distribution of proportion who kiss right")
```

Bootstrap distribution of proportion who kiss right



```
quantile(boot_p$boot_p, c(0.025, 0.975))
```

```
##      2.5%    97.5%
## 0.5564516 0.7258065
```

A 95% confidence interval for the proportion of couples who tilt their heads to the right when they kiss is:

$$(0.56, 0.73)$$

```
quantile(boot_p$boot_p, c(0.005, 0.995))
```

```
##      0.5%    99.5%
## 0.5403226 0.7500000
```

A 99% confidence interval for the proportion of couples who tilt their heads to the right when they kiss is:

$$(0.54, 0.75)$$

```
quantile(boot_p$boot_p, c(0.05, 0.95))
```

```
##      5%      95%
## 0.5725806 0.7177419
```

A 90% confidence interval for the proportion of couples who tilt their heads to the right when they kiss is: $(0.57, 0.72)$

90% confidence interval:	$(0.57, 0.72)$	
95% confidence interval:	$(0.56, 0.73)$	
99% confidence interval:	$(0.54, 0.75)$	

As the confidence level increases, it is more likely that our confidence interval captures the population ~~median~~ proportion

What's the downside to having a higher confidence level?

Confidence intervals get wider
 - less precise

Be careful interpreting confidence intervals

A 95% confidence interval for the proportion of couples who tilt their heads to the right when they kiss is (0.56, 0.73).

Incorrect interpretation #1:

I am 95% confident that the proportion for my sample data will be in my confidence interval.

CI tries to capture population parameter

Incorrect interpretation #2:

The probability that the proportion for the population is in my confidence interval is 95%.

- estimates vary from sample to sample of data, but the parameter doesn't
- the 95% probability is associated with the point estimate, not the parameter
- the parameter is either in the interval or & not

Be careful interpreting confidence intervals

A 95% confidence interval for the proportion of couples who tilt their heads to the right when they kiss is (0.56, 0.73).

Incorrect interpretation #3:

95% of my data is in the interval (0.56, 0.73).

- trying to capture parameter (not data.)

Incorrect interpretation #4:

In 95% of samples I'd get a proportion in the interval (0.56, 0.73).

I'd get a
CI that
captures
the population
parameter

↳ for 1 sample of data

A few notes about the bootstrap

- The bootstrap re-uses our data.
 - Typically larger samples reflect the population better. The bootstrap may work poorly when the sample has a small number of observations.
 - If the sample is biased, the bootstrap confidence interval will also be biased.
 - Using the bootstrap doesn't give us better estimates than the original data of the parameter of interest.
 - It does give us an indication of the accuracy of our estimate.

A few notes about the bootstrap

- The confidence interval method we've used is the *percentile bootstrap method*.
 - There are other bootstrap methods that are more robust, that is they are better at capturing the parameter the correct percentage of the time.
 - The percentile bootstrap method works best for large samples and when the bootstrap distribution is approximately symmetric and continuous.

Reminder of today's concepts:

1. Percentiles
2. Population parameters and sample statistics to estimate them
3. Sampling distribution
4. Bootstrap sampling distribution
5. Confidence intervals