MAT 3850 : Week 10

Fall 2022

App State

Section 1

Outline for the week

By the end of the week: Bootstrapping and Confidence Intervals

- Bootstrapping and Confidence Intervals: Introduction, Resampling activity
- Computer simulation of resampling
- Key features of the Bootstrap and understanding confidence intervals
- Constructing and interpreting confidence intervals

Section 2

Bootstrapping and Confidence Intervals: Introduction

Introduction

In the previous chapter, we studied sampling:

- We started with a "tactile" exercise where we wanted to know the proportion of balls in the sampling bowl that are red.
  - An exhaustive count would have been a tedious process.
  - So instead, we used a shovel to extract a sample of 50 balls and used the resulting proportion that were red as an **estimate**.
  - Furthermore, we made sure to mix the bowl's contents before every use of the shovel.
  - Because of the randomness created by the mixing, different uses of the shovel yielded different proportions red and hence different estimates of the proportion of the bowl's balls that are red.

Introduction

- We then mimicked this "tactile" sampling exercise with an equivalent "virtual" sampling exercise performed on the computer.
  - Using our computer's random number generator, we quickly mimicked the above sampling procedure a large number of times.
  - Specifically, we repeated this sampling procedure 1000 times, using three different "virtual" shovels with 25, 50, and 100 slots.
  - From the visualization of the results, we saw that as the sample size increased, the variation in the estimates decreased.

Introduction

- In performing the sampling exercise performed on the computer, we constructed **sampling distributions**.
  - The motivation for taking 1000 repeated samples and visualizing the resulting estimates was to study how these estimates varied from one sample to another;
  - in other words, we wanted to study the effect of sampling variation.
  - We quantified the variation of these estimates using their standard deviation, which has a special name: the standard error.
  - In particular, we saw that as the sample size increased from 25 to 50 to 100, the standard error decreased and thus the sampling distributions narrowed.
  - Larger sample sizes led to more precise estimates that varied less around the center.

Finally, we then tied these sampling exercises to terminology and mathematical notation related to sampling.

Introduction

- However, both the tactile and virtual sampling exercises are not what one would do in real life.
  - this was merely an activity used to study the effects of sampling variation.
- In a real-life situation:
  - We would not take 1000 samples of size n, but rather take a **single** representative sample that's as large as possible.
  - We will not know what the true proportion of the bowl's balls that were red (this was 37.5%). Because if we did, then why would we take a sample to estimate it?

Introduction

An example of a realistic sampling situation would be a poll, like the **Obama poll** you saw earlier.

- Pollsters did not know the true proportion of all young Americans who supported President Obama in 2013, and thus
  - they took a single sample of size $n = 2089$ young Americans to estimate this value.

Introduction

- So how does one quantify the effects of sampling variation when you only have a single sample to work with?
- You cannot directly study the effects of sampling variation when you only have one sample.
- One common method to study this is **bootstrapping resampling**.

Introduction

Furthermore, what if we would like not only a single estimate of the unknown population parameter, but also a range of highly plausible values?

- Going back to the Obama poll article, it stated that the pollsters' estimate of the proportion of all young Americans who supported President Obama was 41%.
- But in addition it stated that the poll's "margin of error was plus or minus 2.1 percentage points."
- This "plausible range" was

$$[41\% - 2.1\%, 41\% + 2.1\%] = [38.9\%, 43.1\%]$$

This range of plausible values is what's known as a **confidence interval**.

Needed packages

```
library(tidyverse)
library(moderndive)
library(infer)
```

Pennies activity

What is the average year on US pennies in 2019?

- Try to imagine all the pennies being used in the United States in 2019. That's a lot of pennies!
- Now say we're interested in the average year of minting of all these pennies.

Pennies activity

- Approach 1: compute this value would be to gather up all pennies being used in the US, record the year, and compute the average.
  - However, this would be near impossible!
- Approach 2: let's collect a sample of 50 pennies from a local bank in downtown Northampton, Massachusetts, USA.

Pennies activity

An image of these 50 pennies can be seen below:

- For each of the 50 pennies starting in the top left, progressing row-by-row, and ending in the bottom right
  - we assigned an "ID" identification variable and
  - marked the year of minting.

Pennies activity

The `moderndive` package contains this data on our 50 sampled pennies in the `pennies_sample` data frame:

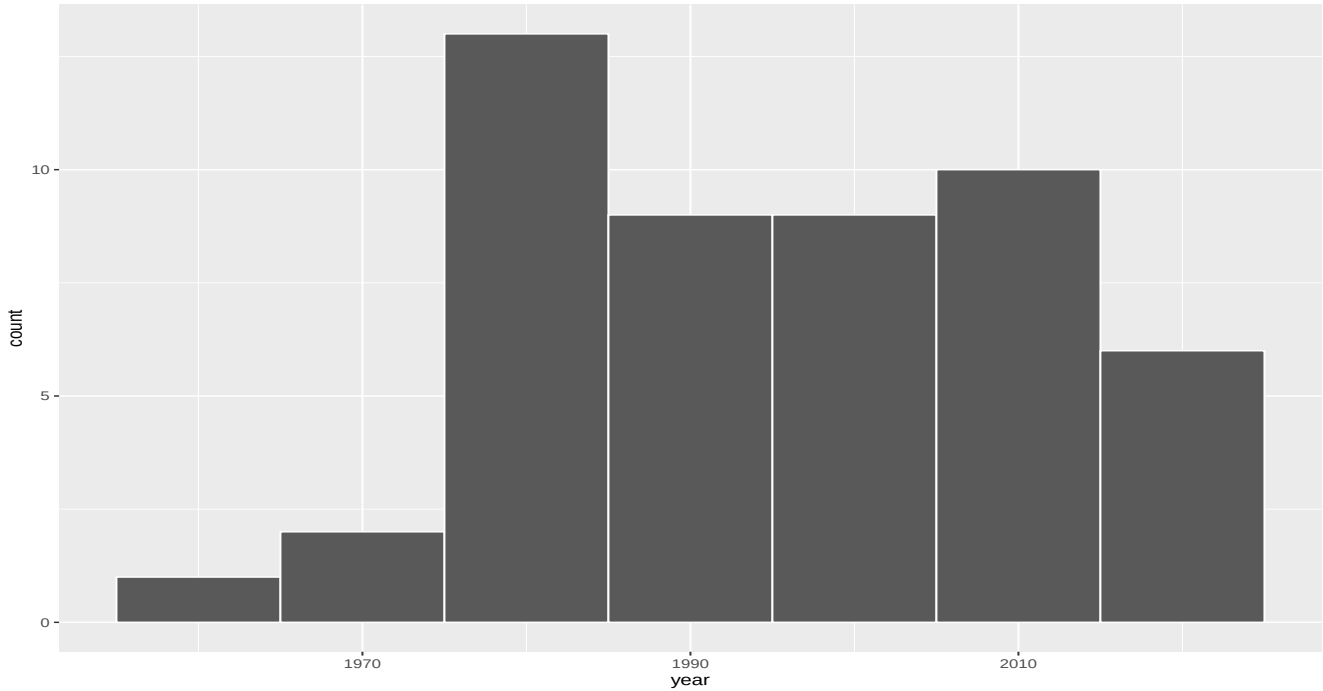
```
pennies_sample
```

```
## # A tibble: 50 x 2
##       ID  year
##    <int> <dbl>
##  1     1  2002
##  2     2  1986
##  3     3  2017
##  4     4  1988
##  5     5  2008
##  6     6  1983
##  7     7  2008
##  8     8  1996
##  9     9  2004
## 10    10  2000
## # ... with 40 more rows
```

Pennies activity

Let's first visualize the distribution of the year of these 50 pennies.

```
ggplot(pennies_sample, aes(x = year)) +
  geom_histogram(binwidth = 10, color = "white")
```



Pennies activity

```
x_bar <- pennies_sample %>% summarize(mean_year = mean(year))
x_bar
```

```
## # A tibble: 1 x 1
##   mean_year
##       <dbl>
## 1     1995.
```

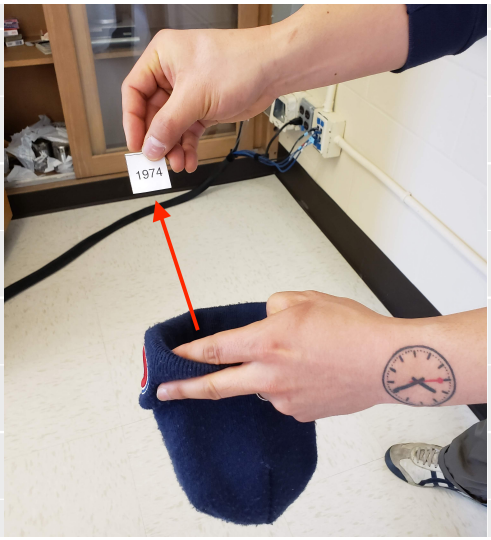
Thus, if we're willing to assume that `pennies_sample` is a representative sample from all US pennies, a "good guess" of the average year of minting of all US pennies would be 1995.44 (around 1995).

Pennies activity

We summarize the correspondence between the sampling bowl exercise and our pennies exercise.

Scenario	Population parameter	Notation	Point estimate	Symbol(s)
1	Population proportion	p	Sample proportion	$\hat{p}$
2	Population mean	μ	Sample mean	$\overline{x}$ or $\widehat{\mu}$

Resampling once



Resampling once

What we just performed was a **resampling** of the original sample of 50 pennies.

- Now ask yourselves, why did we replace our resampled slip of paper back into the hat in Step 4?
  - replacing the slips of paper induces sampling variation.
  - this termed resampling with replacement
- Had we left the slip of paper out of the hat each time we performed Step 4, this would be **resampling without replacement**.

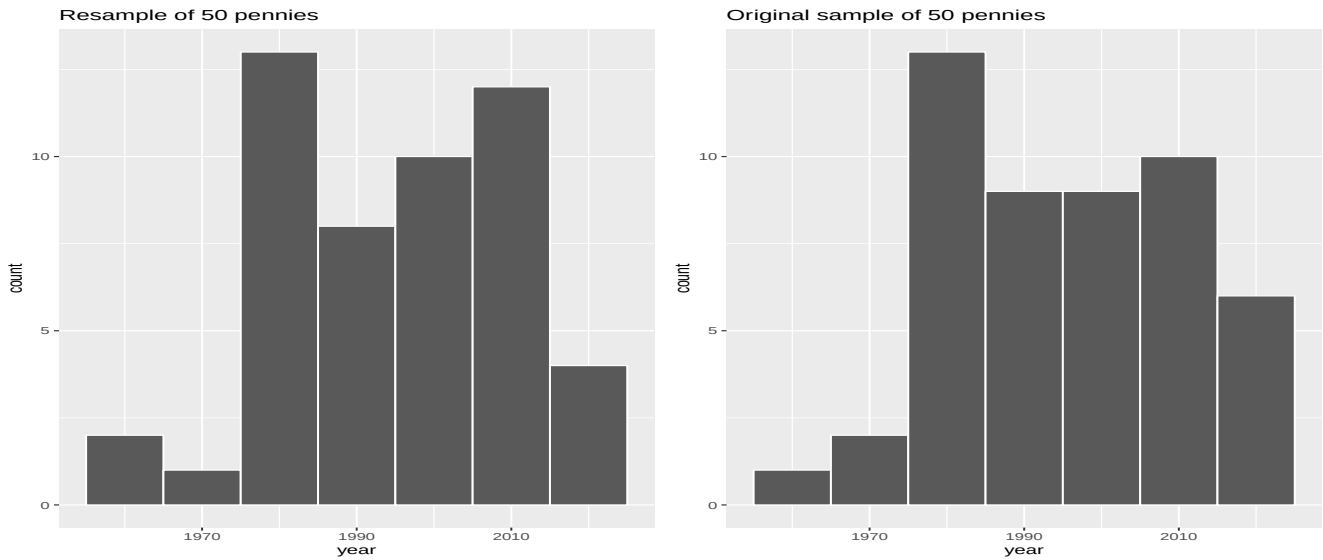
Resampling once

```
pennies_resample <- tibble(
  year = c(1976, 1962, 1976, 1983, 2017, 2015, 2015, 1962, 2016, 1976,
           2006, 1997, 1988, 2015, 2015, 1988, 2016, 1978, 1979, 1997,
           1974, 2013, 1978, 2015, 2008, 1982, 1986, 1979, 1981, 2004,
           2000, 1995, 1999, 2006, 1979, 2015, 1979, 1998, 1981, 2015,
           2000, 1999, 1988, 2017, 1992, 1997, 1990, 1988, 2006, 2000)
)
```

Resampling once

Let's compare the distribution of the numerical variable `year` from resampled pennies and original sample of 50 pennies.

```
P1<-ggplot(pennies_resample, aes(x = year)) +
  geom_histogram(binwidth = 10, color = "white") +labs(title = "Resample of 50 pennies")
P2<-ggplot(pennies_sample, aes(x = year)) +
  geom_histogram(binwidth = 10, color = "white") +labs(title = "Original sample of 50 pennies")
grid.arrange(P1, P2, ncol=2)
```



while the general shapes of both distributions of year are roughly similar, they are not identical.

Resampling once

```
pennies_resample %>% summarize(mean_year = mean(year)) %>%
  round(digits = 4)
```

```
## # A tibble: 1 x 1
##   mean_year
##       <dbl>
## 1     1995.
```

We obtained a different mean year of 1994.82. This variation is induced by the resampling with replacement we performed earlier.

Resampling 35 times

Each of our 35 friends will repeat the same five steps:

1. Start with 50 identically sized slips of paper representing the 50 pennies.
2. Put the 50 small pieces of paper into a hat or beanie cap.
3. Mix the hat's contents and draw one slip of paper at random. Record the year in a spreadsheet.
4. Replace the slip of paper back in the hat!
5. Repeat Steps 3 and 4 a total of 49 more times, resulting in 50 recorded years.

Since we had 35 of our friends perform this task, we ended up with $50 \times 35 = 1750$ values saved them in `pennies_resamples`.

Resampling 35 times

```
pennies_resamples
```

```
## # A tibble: 1,750 x 3
## # Groups:   name [35]
##    replicate name     year
##        <int> <chr>   <dbl>
##  1         1 Arianna  1988
##  2         1 Arianna  2002
##  3         1 Arianna  2015
##  4         1 Arianna  1998
##  5         1 Arianna  1979
##  6         1 Arianna  1971
##  7         1 Arianna  1971
##  8         1 Arianna  2015
##  9         1 Arianna  1988
## 10         1 Arianna  1979
## # ... with 1,740 more rows
```

Resampling 35 times

What did each of our 35 friends obtain as the mean year?

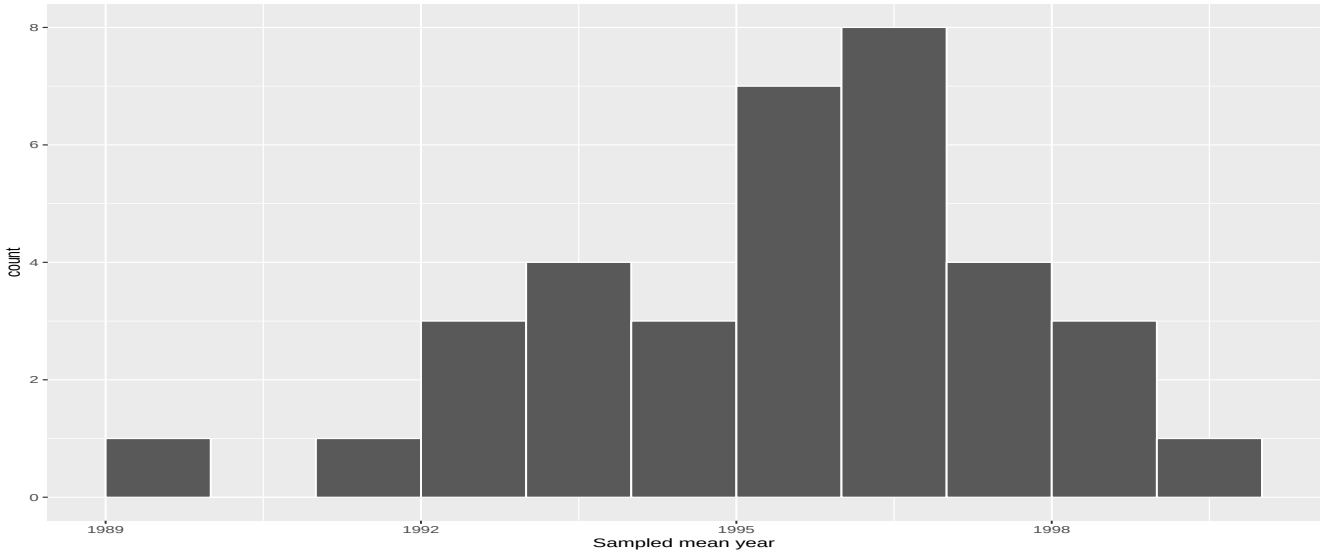
```
resampled_means <- pennies_resamples %>%
  group_by(name) %>%
  summarize(mean_year = mean(year))
resampled_means
```

```
## # A tibble: 35 x 2
##    name      mean_year
##    <chr>         <dbl>
##  1 Arianna       1992.
##  2 Artemis       1996.
##  3 Bea           1996.
##  4 Camryn        1997.
##  5 Cassandra     1991.
##  6 Cindy         1995.
##  7 Claire        1996.
##  8 Dahlia        1998.
##  9 Dan           1994.
## 10 Eindra        1994.
## # ... with 25 more rows
```

Resampling 35 times

Let's visualize this variation using a histogram

```
ggplot(resampled_means, aes(x = mean_year)) +
  geom_histogram(binwidth = 1, color = "white", boundary = 1990) +
  labs(x = "Sampled mean year")
```



- Observe that
  - the distribution looks roughly normal
  - distribution is roughly centered at 1995, which is close to the sample mean of 1995.44 of the original sample of 50 pennies from the bank.

What did we just do?

What we just demonstrated in this activity is the statistical procedure known as **bootstrap resampling with replacement**.

- We used resampling to mimic the sampling variation we studied earlier on sampling.
  - However, in this case, we did so using only a **single** sample from the population.
- In fact, the histogram of sample means from 35 resamples is called the **bootstrap distribution**.
  - It is an approximation to the sampling distribution of the sample mean.
  - Using this bootstrap distribution, we can study the effect of sampling variation on our estimates.
  - In particular, we'll study the typical "error" of our estimates, known as the standard error.

Section 3

Computer simulation of resampling

Virtually resampling once

- Recall, we are using the `pennies_sample` data frame included in the `moderndive` package.
  - The `pennies_sample` data contains the years of our original sample of 50 pennies from the bank.

```
set.seed(10)
virtual_resample <- pennies_sample %>% rep_sample_n(size = 50, replace = TRUE)
virtual_resample
```

```
## # A tibble: 50 x 3
## # Groups:   replicate [1]
##    replicate    ID  year
##        <int> <int> <dbl>
##  1         1    43  2018
##  2         1     9  2004
##  3         1    10  2000
##  4         1    48  1988
##  5         1    12  1995
##  6         1     8  1996
##  7         1    39  2015
##  8         1    19  1983
##  9         1    24  2017
## 10         1    15  1974
## # ... with 40 more rows
```

Virtually resampling once

```
set.seed(10)
virtual_resample %>% summarize(resample_mean = mean(year))
```

```
## # A tibble: 1 x 2
##   replicate resample_mean
##       <int>         <dbl>
## 1         1         1997.
```

Observe that the resulting mean year is different than the mean year of our 50 originally sampled pennies of 1995.44.

Virtually resampling 35 times

```
set.seed(10)
virtual_resamples <- pennies_sample %>% rep_sample_n(size = 50, replace = TRUE, reps = 35)
virtual_resamples
```

```
## # A tibble: 1,750 x 3
## # Groups:   replicate [35]
##    replicate    ID  year
##        <int> <int> <dbl>
##  1         1    43  2018
##  2         1     9  2004
##  3         1    10  2000
##  4         1    48  1988
##  5         1    12  1995
##  6         1     8  1996
##  7         1    39  2015
##  8         1    19  1983
##  9         1    24  2017
## 10         1    15  1974
## # ... with 1,740 more rows
```

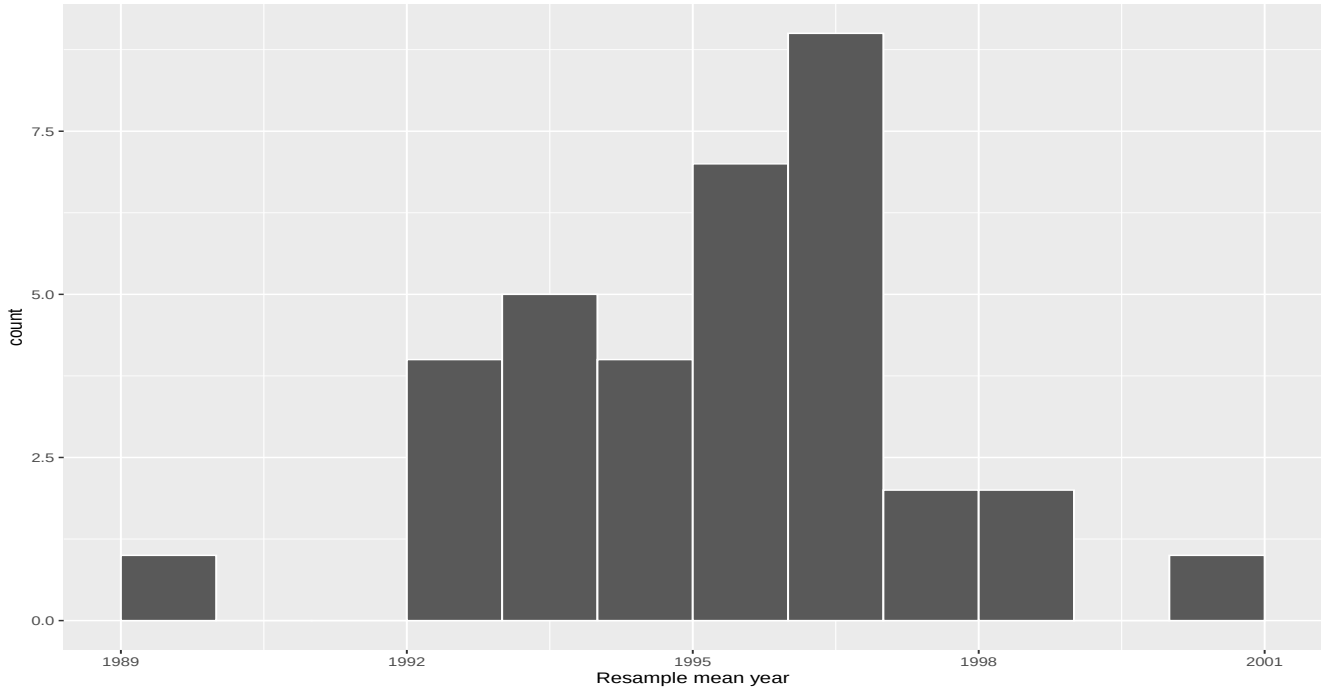
Virtually resampling 35 times

```
set.seed(10)
virtual_resampled_means <- virtual_resamples %>% group_by(replicate) %>%
  summarize(mean_year = mean(year))
virtual_resampled_means
```

```
## # A tibble: 35 x 2
##    replicate mean_year
##        <int>     <dbl>
##  1         1     1997.
##  2         2     1992.
##  3         3     1993.
##  4         4     1998.
##  5         5     1996.
##  6         6     1996.
##  7         7     1997.
##  8         8     1990.
##  9         9     1997.
## 10        10     1993.
## # ... with 25 more rows
```

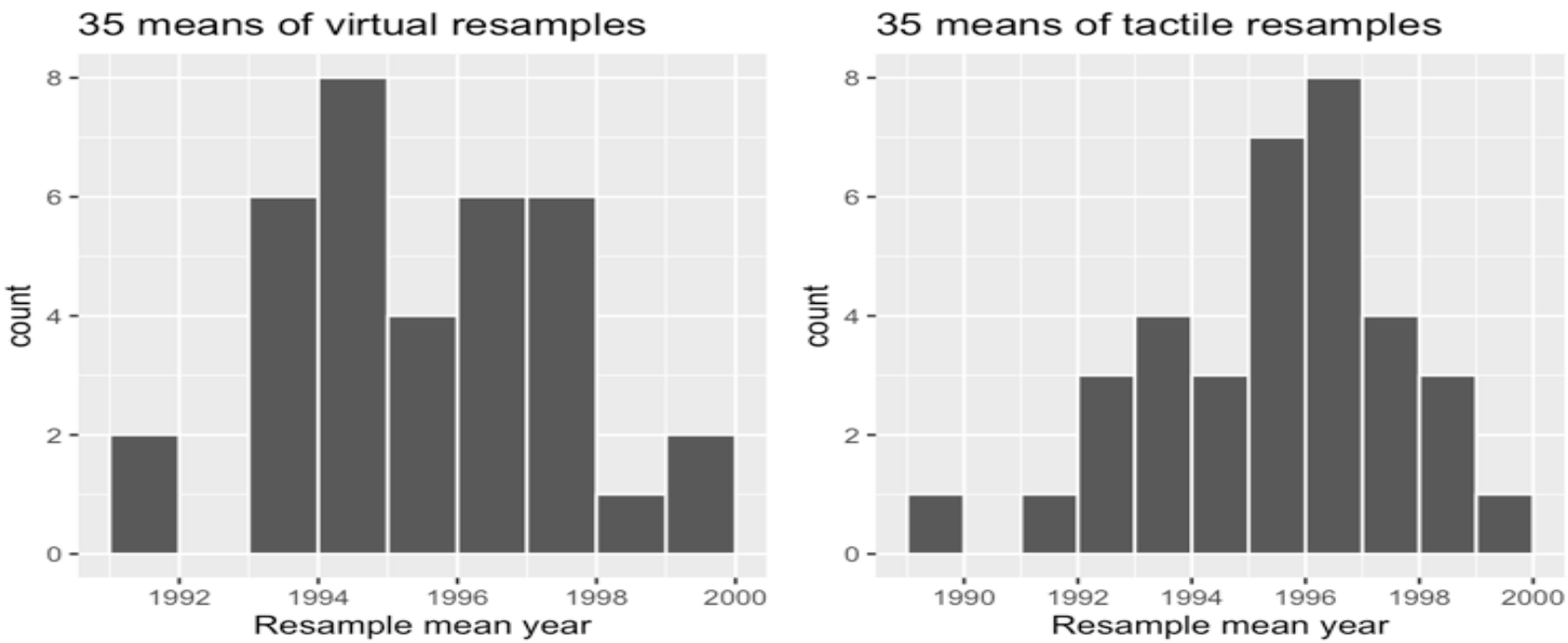
Virtually resampling 35 times

```
set.seed(10)
ggplot(virtual_resampled_means, aes(x = mean_year)) +
  geom_histogram(binwidth = 1, color = "white", boundary = 1990) +
  labs(x = "Resample mean year")
```



Virtually resampling 35 times

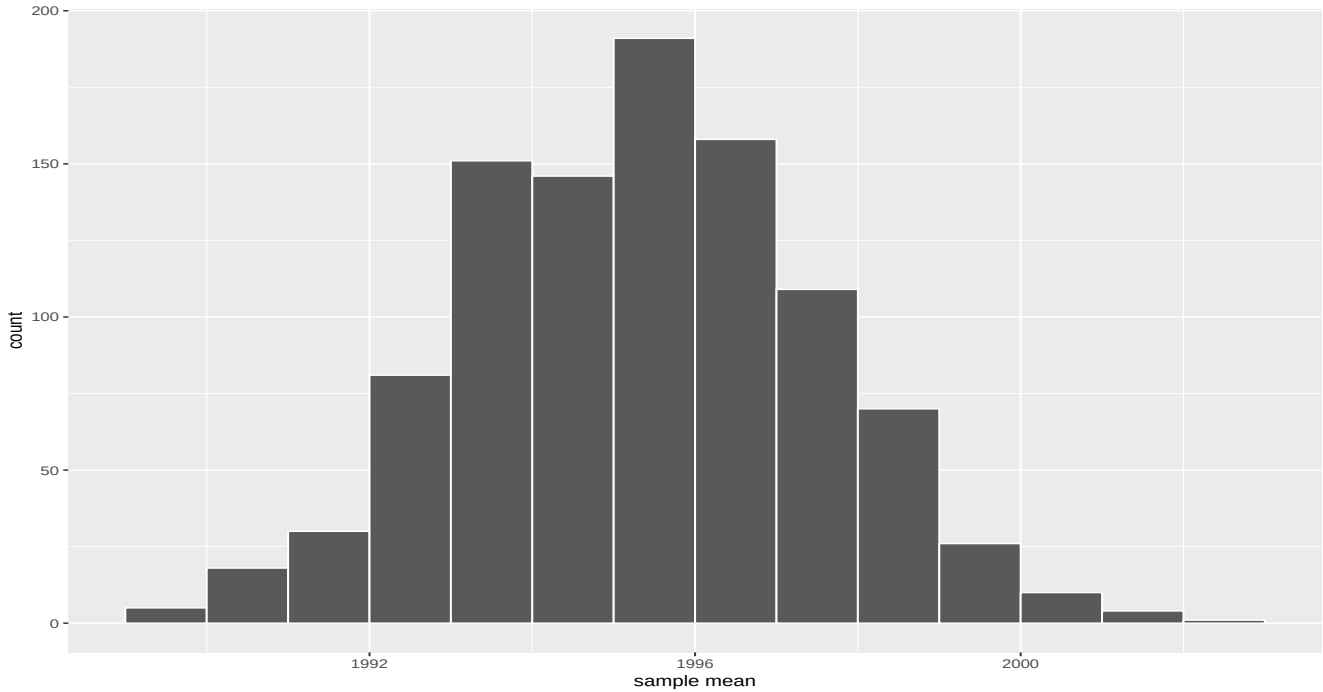
Let's compare our virtually constructed bootstrap distribution with the one our 35 friends constructed via our tactile resampling exercise.



Observe how they are somewhat similar, but not identical.

Virtually resampling 1000 times

```
set.seed(10)
virtual_resampled_means <- pennies_sample %>%
  rep_sample_n(size = 50, replace = TRUE, reps = 1000) %>%
  group_by(replicate) %>%
  summarize(mean_year = mean(year))
ggplot(virtual_resampled_means, aes(x = mean_year)) +
  geom_histogram(binwidth = 1, color = "white", boundary = 1990) +
  labs(x = "sample mean")
```



Virtually resampling 1000 times

```
virtual_resampled_means %>%
  summarize(mean_of_means = mean(mean_year))
```

```
## # A tibble: 1 x 1
##   mean_of_means
##           <dbl>
## 1         1995.
```

- The mean of these 1000 means is 1995.35, which is quite close to the mean of our original sample of 50 pennies of 1995.44.
- Congratulations! You've just constructed your first bootstrap distribution!

Example: Birth weight of a baby

The birth weight of a baby is of interest to health officials since many studies have shown possible links between this weight and conditions in later life, such as obesity or diabetes. Researchers look for possible relationships between the birth weight of a baby and age of the mother or whether or not she smokes cigarettes or drink alcohol during her pregnancy. The centers for disease control and prevention CDC, using data provided by the U.S. Department of Health and Human Services, National Center for Health Statistics, the Division of Vital Statistics as well as the CDC, maintain a database on all babies born in a given year. We will investigate different samples taken from the CDC's database of births.

Example: Birth weight of a baby

We will use the `NCBirths2004` data from the `resampledata` package.

```
library(resampledata)
Babies <- NCBirths2004
head(Babies)
```

```
##   ID MothersAge Tobacco Alcohol Gender Weight Gestation Smoker
## 1  1       30-34      No      No   Male   3827        40     No
## 2  2       30-34      No      No   Male   3629        38     No
## 3  3       35-39      No      No Female   3062        37     No
## 4  4       20-24      No      No Female   3430        39     No
## 5  5       25-29      No      No   Male   3827        38     No
## 6  6       35-39      No      No Female   3119        39     No
```

Example: Birth weight of a baby

```
Babies %>% summarize(Mean = mean(Weight), n = n())
```

```
##       Mean    n
## 1 3448.26 1009
```

- For the North Carolina data, the mean weight of the 1009 babies in the sample is $3448.26g$.
- We are interested in μ, **the true birth weight mean** for all North Carolina babies born in 2004;
  - this is probably not the same as the sample mean.
- We have already mentioned the different samples of the same size will yield different sample means, so how can we gauge the accuracy of the 3448.26 as an estimate to μ?

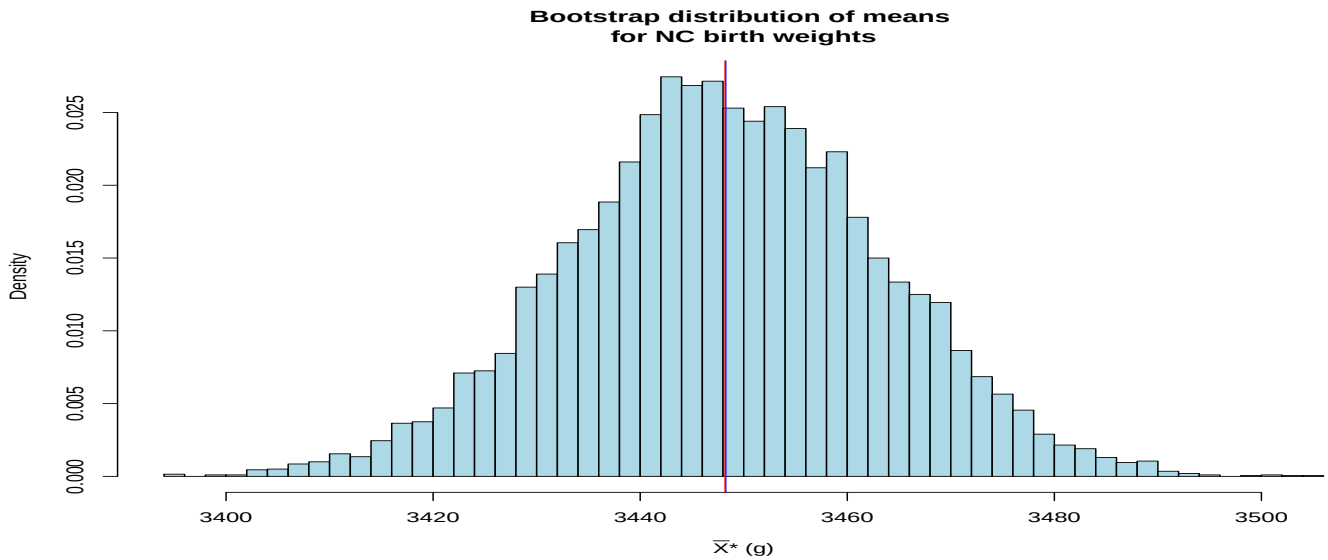
Example: Birth weight of a baby

- If we knew the sampling distribution of sample means for samples of size 1009 from the population of all 2004 North Carolina births,
  - then this would give us an idea of how means vary from sample to sample, for the standard error of the sampling distribution tells us how far means deviate from the population mean μ
  - But of course, since we do not have all the birth weights, we cannot guarantee the sampling distribution (and if we did have all the weights, we would know the true μ!).
- Hence, we would obtain the **bootstrap distribution**, that approximates the sampling distribution for the sample mean (or for other statistics).

Example: Birth weight of a baby

Own routine:

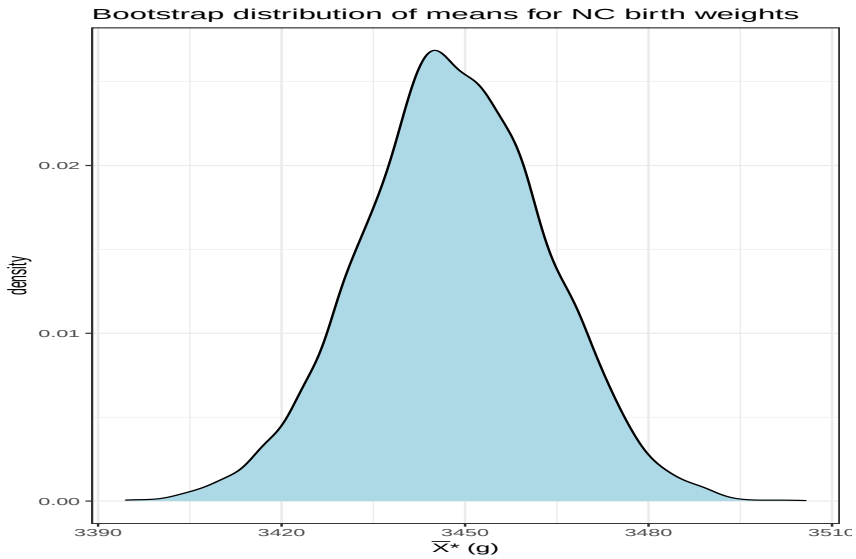
```
set.seed(13)
B <- 10000  ## Repeat this resampling process many times, say 10,000
my.boot.statB <- numeric(B)
for (i in 1:B){
  x <- sample(Babies$Weight, size = sum(!is.na(Babies$Weight)), replace = TRUE)
  my.boot.statB[i] <- mean(x)
}
hist(my.boot.statB, breaks = "Scott", col = 'lightblue', xlab = substitute(paste(bar(X),"* (g)")),
     main ="Bootstrap distribution of means \n for NC birth weights ", freq= FALSE)
abline(v = mean(Babies$Weight), col = "blue")
abline(v = mean(my.boot.statB), col = "red")
```

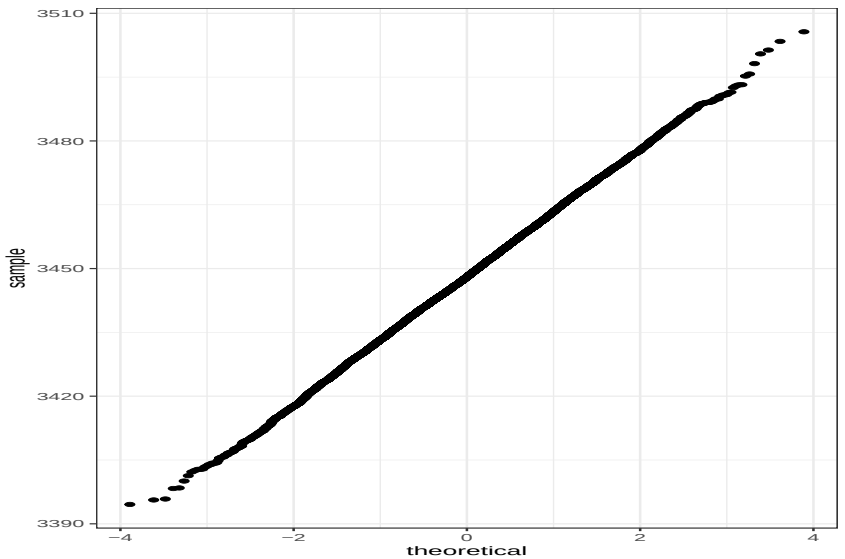


Example: Birth weight of a baby

```
p1<-ggplot(data = data.frame(x = my.boot.statB), aes(x = x)) +
  geom_density(fill = "lightblue") +
  labs(x = substitute(paste(bar(X),"* (g)")),
       title = "Bootstrap distribution of means for NC birth weights") +
  theme_bw()

p2 <- ggplot(data = data.frame(x = my.boot.statB), aes(sample = x)) +
  stat_qq() +
  theme_bw()
gridExtra::grid.arrange(p1, p2, ncol = 2)
```





Example: Birth weight of a baby

- Note that:
  - the bootstrap distribution is approximately normal.
  - Second, with mean 3448.1998813, it is centered at approximately the same location as the original mean, 3448.259663.
- We can quantify the variability by computing the standard deviation of the bootstrap distribution,
  - in this case 15.0452075.
  - This is the bootstrap standard error.

Example: Birth weight of a baby

The difference between where the bootstrap distribution is centered and the mean of the of original sample is the bootstrap bias in this example -0.0597818.

```
boot.bias <- mean(my.boot.statB) - mean(Babies$Weight)
boot.bias
```

```
## [1] -0.05978176
```

In general, the bias of an estimate $\hat{\theta}$:

$$Bias[\hat{\theta}] = E[\hat{\theta}] - \theta$$

The bootstrap estimate of bias is

$$Bias_{boot}[\hat{\theta}^*] = E[\hat{\theta}^*] - \theta$$

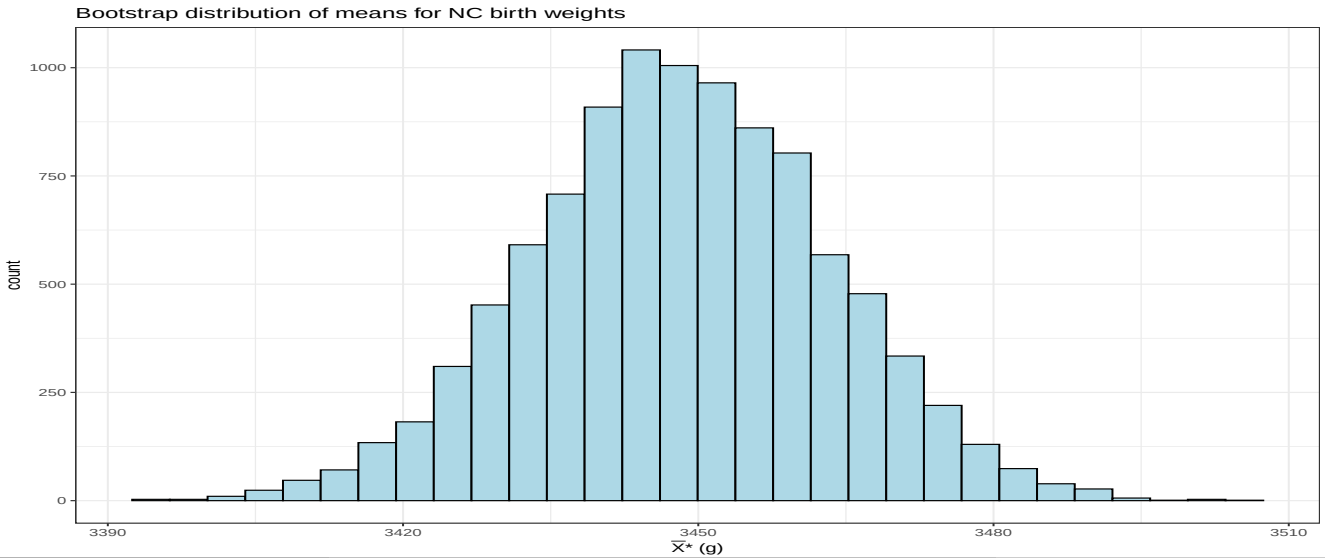
Example: Birth weight of a baby

Using `rep_sample_n` from the `infer` package.

```
set.seed(13)
Babies %>% rep_sample_n(size = 1009, replace = TRUE, reps = 10000) %>%
  group_by(replicate) %>% summarize(stat = mean(Weight)) -> BSD
c(mean(BSD$stat), sd(BSD$stat))
```

```
## [1] 3448.19988   15.04521
```

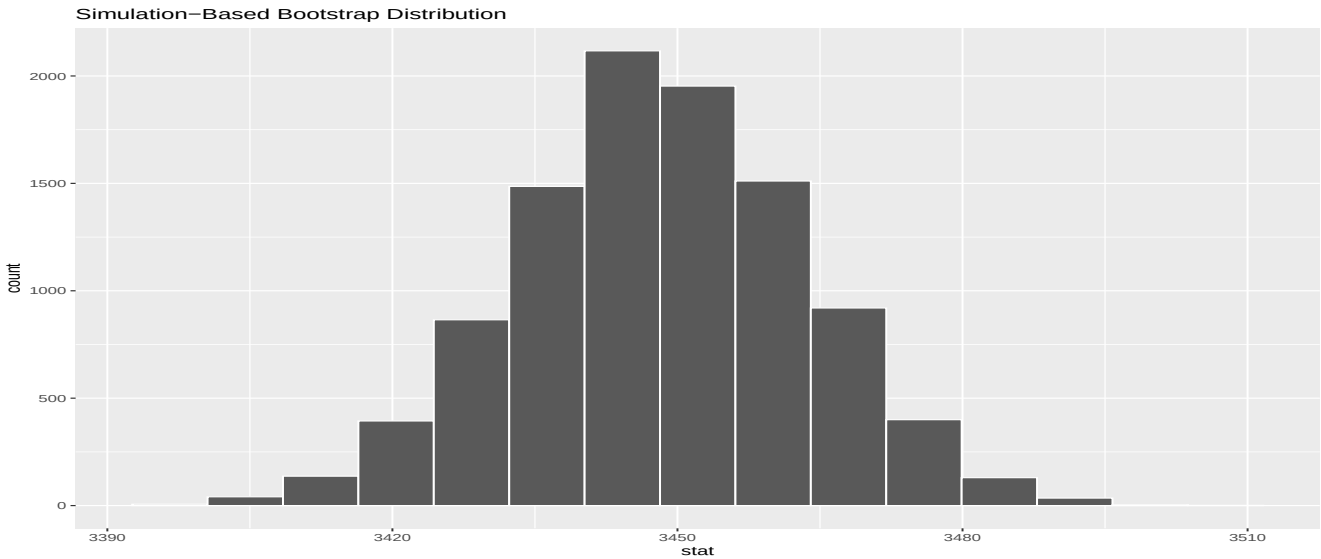
```
ggplot(data = BSD, aes(x = stat)) +
  geom_histogram(fill = "lightblue", color = "black") +
  labs(x = substitute(paste(bar(X),"* (g)")),
       title = "Bootstrap distribution of means for NC birth weights") +
  theme_bw()
```



Example: Birth weight of a baby

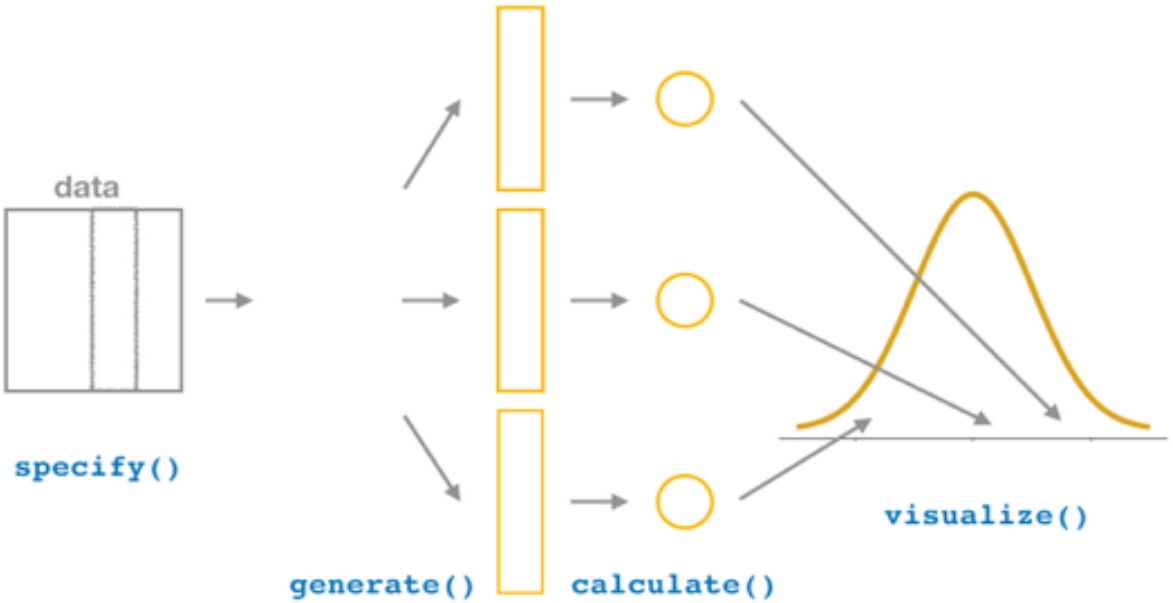
Using the `infer` pipeline

```
set.seed(13)
library(infer)
bsd <- Babies %>%
  specify(response = Weight) %>%
  generate(reps = 10^4, type = "bootstrap") %>%
  calculate(stat = "mean")
visualize(bsd)
```



The `infer` workflow is much simpler for confidence intervals and hypothesis testing with minimal changes to your code.

`infer` pipeline



- After we `specify()` the variables of interest, we pipe the results into the `generate()` function to generate replicates.
  - `calculate()` function summarizes each of the resamples of statistic to a single sample statistic value.
  - `visualize()` verb provides a quick way to visualize the bootstrap distribution.

Section 4

Key features of the Bootstrap

Key features of the Bootstrap: Example 1

To highlight some key features of the bootstrap distribution:

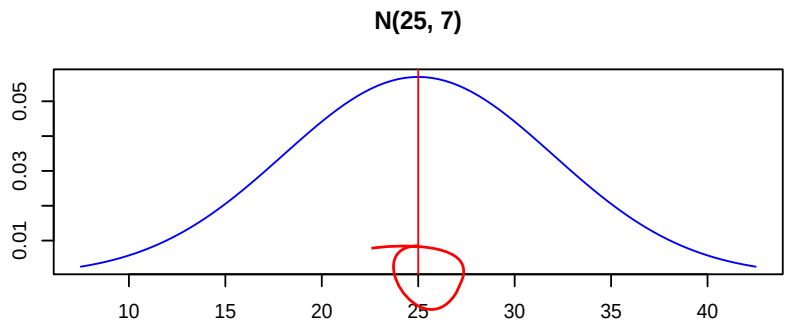
- we begin with two examples in which the theoretical sampling distributions of the mean are known.
- Example: Consider a random sample of size 49 drawn from a $N(25, 7)$.
  - Theory tells us that that the sampling distribution of the sample means is normal with mean 25 and standard error $\sigma = 7/\sqrt{49} = 1$.

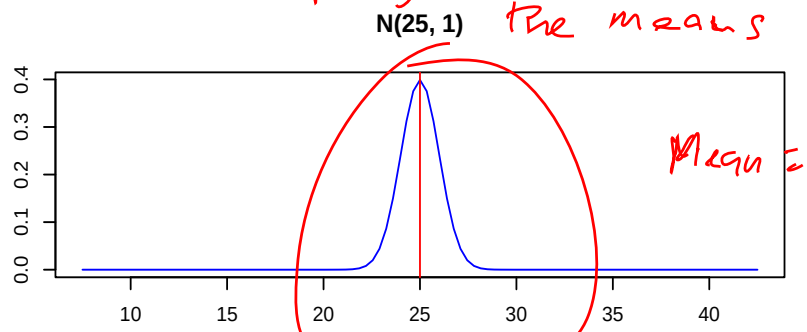
$$\bar{x} \sim N\left(\mu, \frac{6}{\sqrt{n}}\right) = N\left(25, \frac{7}{\sqrt{49}}\right) = N(25, 1)$$

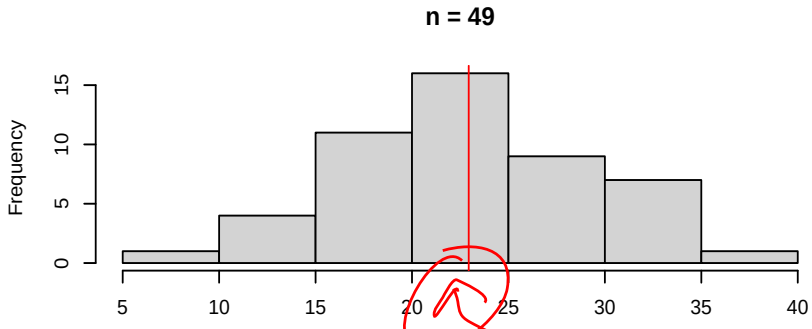
Key features of the Bootstrap: Example 1

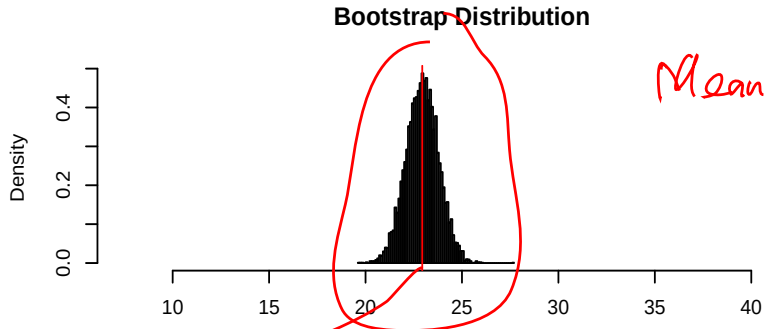
```
set.seed(11)
par(mfrow = c(3, 2))
curve(dnorm(x, 25, 7), from = 25 - 2.5*7, 25 + 2.5*7, col = "blue", main = "N(25, 7)", ylab = "", xlab = "")
abline(v = 25, col = "red")
curve(dnorm(x, 25, 1), from = 25 - 2.5*7, 25 + 2.5*7, col = "blue", main = "N(25, 1)", ylab = "", xlab = "")
abline(v = 25, col = "red")
rs1 <- rnorm(49, 25, 7)
rs2 <- rnorm(49, 25, 7)
hist(rs1, xlab = "", main = "n = 49")
abline(v = mean(rs1), col = "red")
B <- 10000
my.boot.stat1 <- numeric(B)
my.boot.stat2 <- numeric(B)
for (i in 1:B){
  x1 <- sample(rs1, size = 49, replace = TRUE)
  x2 <- sample(rs2, size = 49, replace = TRUE)
  my.boot.stat1[i] <- mean(x1)
  my.boot.stat2[i] <- mean(x2)
}
hist(my.boot.stat1, breaks = "Scott",  main ="Bootstrap Distribution", freq= FALSE, xlab = "",
xlim = c(25 - 2.5*7, 25 + 2.5*7))
abline(v = mean(rs1), col = "red")
hist(rs2, xlab = "", main = "n = 49")
abline(v = mean(rs2), col = "red")
hist(my.boot.stat2, breaks = "Scott",  main ="Bootstrap Distribution", freq= FALSE, xlab = "",
xlim = c(25 - 2.5*7, 25 + 2.5*7))
abline(v = mean(rs2), col = "red")
```

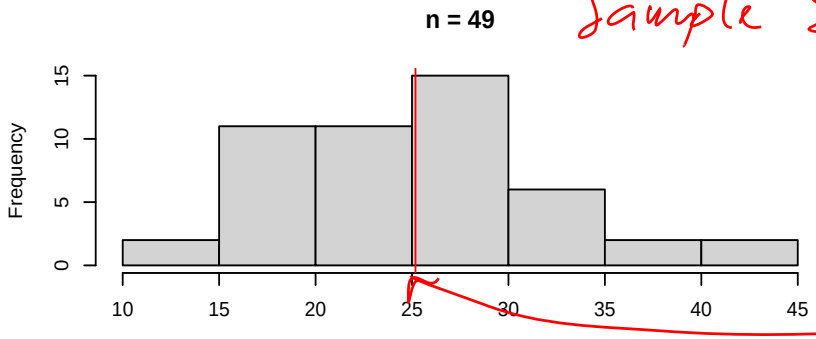
Key features of the Bootstrap: Example 1

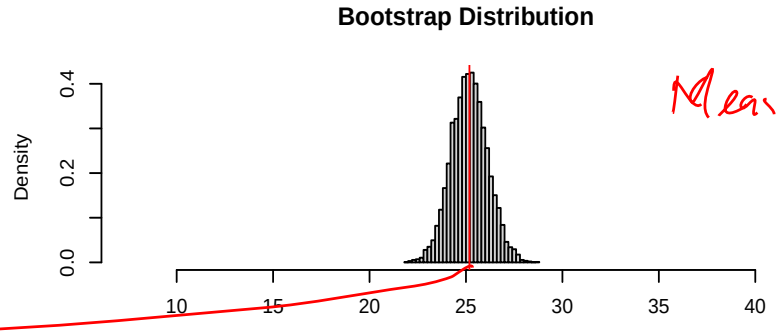












Key features of the Bootstrap: Example 1

- The previous code shows the distributions of two such random samples with sample means:
  - $\bar{x}_1 = 22.9402808$ and, $\bar{x}_2 = 25.1793901$
  - $s_1 = 5.9322341$ and, $s_2 = 6.6413206$
- Based on the graphs above, we can see that the bootstrap distribution:
  - has roughly the same spread and shape as the theoretical sampling distribution,
  - but the centers are different compared to the theoretical sampling distribution.

Key features of the Bootstrap: Example 1

- This example illustrates some important features of the bootstrap that hold for other statistics besides the mean:
  - the bootstrap distribution of a particular statistic $\hat{\theta}$ has approximately the same spread and shape as the sampling distribution of the statistic $\hat{\theta}$, but
  - the center of the bootstrap distribution is at the center of the original sample.
- Hence we do not use the center of the bootstrap distribution in its own right, but we do compare the center of the bootstrap distribution with the observed statistic; if they differ, it indicates bias.
- For most statistics, bootstrap distributions approximate the spread, bias, and shape of the actual sampling distribution.

Key features of the Bootstrap: Example 2

We now consider an example where neither the population nor the sampling distribution is normal.

- A random variable X that has a Gamma distribution is written $X \sim \Gamma(\alpha, \lambda)$.
- If X is a Gamma random variable, then:

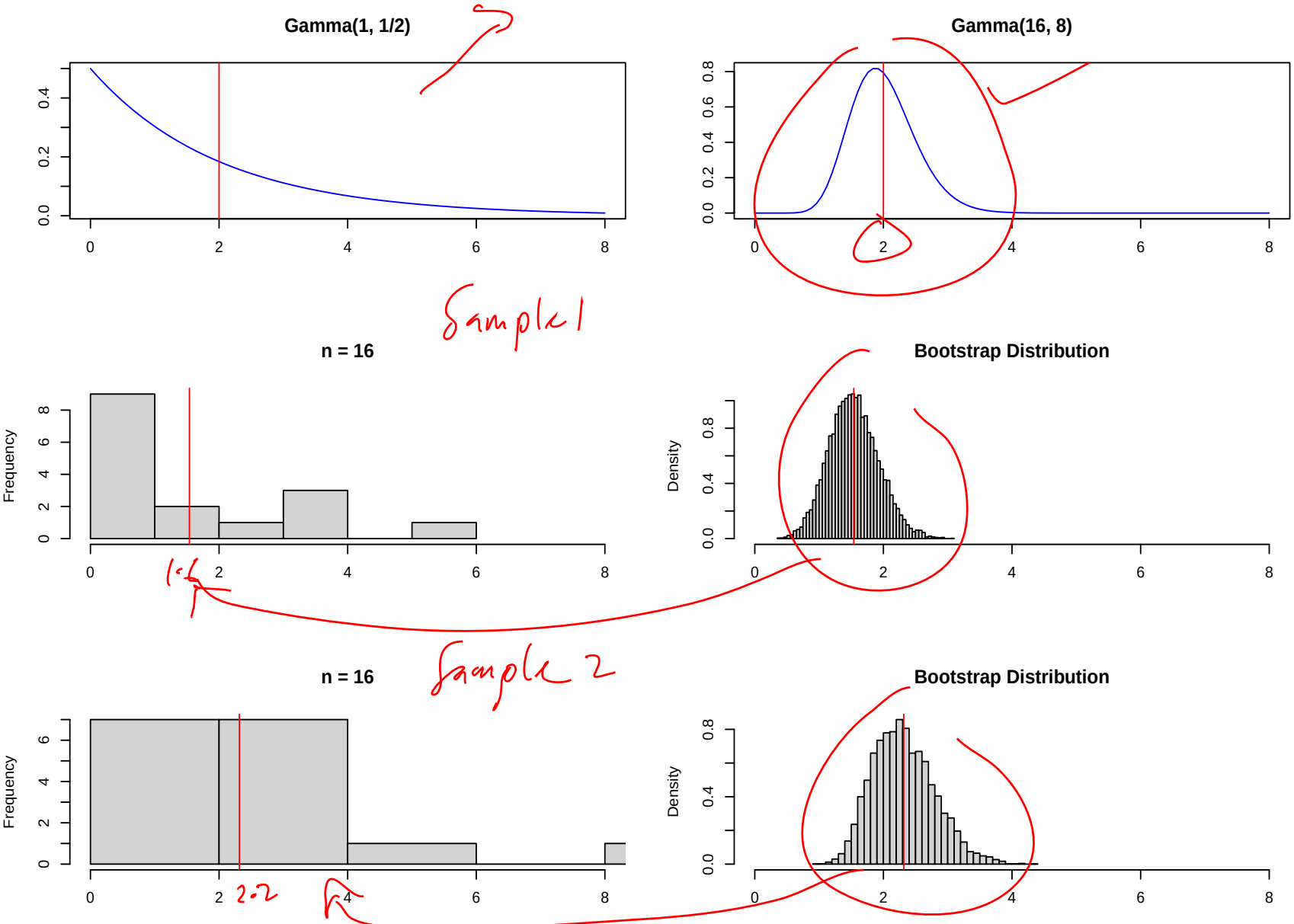
$$E[X] = \alpha/\lambda \qquad \text{and} \qquad Var[X] = \alpha/\lambda^2.$$

- Let $X_1, \ldots, X_n \sim \Gamma(\alpha, \lambda)$.
  - It is a fact that the sampling distribution of the mean $\bar{X}$ is $\Gamma(n\alpha, n\lambda)$.
- We draw a random sample of size $n = 16$ from a $\Gamma(\alpha = 1, \lambda = 1/2)$ (population mean 2, standard deviation 2).

Key features of the Bootstrap: Example 2

```
set.seed(18)
par(mfrow = c(3, 2))
curve(dgamma(x, 1, 1/2), from = 0, to = 8, col = "blue", main = "Gamma(1, 1/2)", ylab = "", xlab = "")
abline(v = 2, col = "red")
curve(dgamma(x, 16, 8), from = 0, 8, col = "blue", main = "Gamma(16, 8)", ylab = "", xlab = "")
abline(v = 2, col = "red")
rsg1 <- rgamma(16, 1, 1/2)
rsg2 <- rgamma(16, 1, 1/2)
hist(rsg1, xlab = "", main = "n = 16", xlim = c(0, 8))
abline(v = mean(rsg1), col = "red")
B <- 10000
my.boot.statg1 <- numeric(B)
my.boot.statg2 <- numeric(B)
for (i in 1:B){
  xg1 <- sample(rsg1, size = 16, replace = TRUE)
  xg2 <- sample(rsg2, size = 16, replace = TRUE)
  my.boot.statg1[i] <- mean(xg1)
  my.boot.statg2[i] <- mean(xg2)
}
hist(my.boot.statg1, breaks = "Scott",  main ="Bootstrap Distribution", freq= FALSE, xlab = "",
xlim = c(0, 8))
abline(v = mean(rsg1), col = "red")
hist(rsg2, xlab = "", main = "n = 16", xlim = c(0, 8))
abline(v = mean(rsg2), col = "red")
hist(my.boot.statg2, breaks = "Scott",  main ="Bootstrap Distribution", freq= FALSE, xlab = "",
xlim = c(0, 8))
abline(v = mean(rsg2), col = "red")
```

Key features of the Bootstrap: Example 2



Key features of the Bootstrap: Example 2

- The first graph in the second and third rows shows the distribution of a random sample with sample means and standard deviations
  - $\bar{x}_1 = 1.5386904$ and, $\bar{x}_2 = 2.3164796$
  - $s_1 = 1.6062418$ and, $s_2 = 1.9916418$
- Based on the graphs above, we can see that the bootstrap distribution:
  - has roughly the same spread and shape as the theoretical sampling distribution,
  - but the centers are different compared to the theoretical sampling distribution.

Key features of the Bootstrap

For most common estimators and under fairly general distribution assumptions, the following need to be noted:

- **Center**: The center of the bootstrap distribution is not an accurate approximation for the center of the sampling distribution.
  - For example, the center of the bootstrap distribution for $\bar{X}$ is centered at approximately $\bar{x} = \mu_{\hat{F}}$, the mean of the sample, whereas
  - the sampling distribution is centered at μ.
- **Spread**: The spread of the bootstrap distribution does reflect the spread of the sampling distribution.
- **Bias**: The bootstrap bias estimate does reflect the bias of the sampling distribution. Bias occurs if a sampling distribution is not centered at the parameter. $\text{Bias} = E(\hat{\theta}) - \theta$
- **Skewness**: The skewness of the bootstrap distribution does reflect the skewness of the sampling distribution.

Key features of the Bootstrap

- The first point bears emphasis.
  - It means that the bootstrap is not used to get better parameter estimates because the bootstrap distributions are centered around statistics $\hat{\theta}$ calculated from the data rather than unknown population values.
  - Drawing thousands of bootstrap observations from the original data is not like drawing observations from the underlying population, it does not create new data.
- Instead, the bootstrap distribution is useful for quantifying the behavior of a parameter estimate such as its :
  - standard error,
  - skewness, bias, or
  - for calculating confidence intervals.

Key features of the Bootstrap: Example 3

Arsenic is a naturally occurring element in the groundwater of Bangladesh. However, much of this groundwater is used for drinking water by rural populations, so arsenic poisoning is a serious health issue. Figure 1a displays the distribution of arsenic concentrations from 271 wells in Bangladesh. The sample mean and standard deviation are $\bar{x} = 125.32$ and $s = 297.98$, respectively (measured in micrograms per liter). We draw resamples of size 271 with replacement from the data and compute the mean for each resample.

Key features of the Bootstrap: Example 3

```
par(mfrow=c(2, 2))
Bang <- Bangladesh
Arsenic <- Bang$Arsenic
hist(Arsenic, breaks = "Scott", main = "Figure 1a", col = "lightblue")
qqnorm(Arsenic, main = "Figure 1b")
qqline(Arsenic, col = "red")
B <- 10000
n <- sum(!is.na(Arsenic))
arsenic.mean <- numeric(B)
set.seed(7)
for (i in 1:B){
  x <- sample(Arsenic, size = n, replace = TRUE)
  arsenic.mean[i] <- mean(x)
}
hist(arsenic.mean, main = "Figure 2a", col = "lightblue", breaks = "Scott", xlab = substitute(paste(bar(X),"*")))
qqnorm(arsenic.mean, main = "Figure 2b")
qqline(arsenic.mean, col = "red")
```

Key features of the Bootstrap: Example 3

Figure 1a

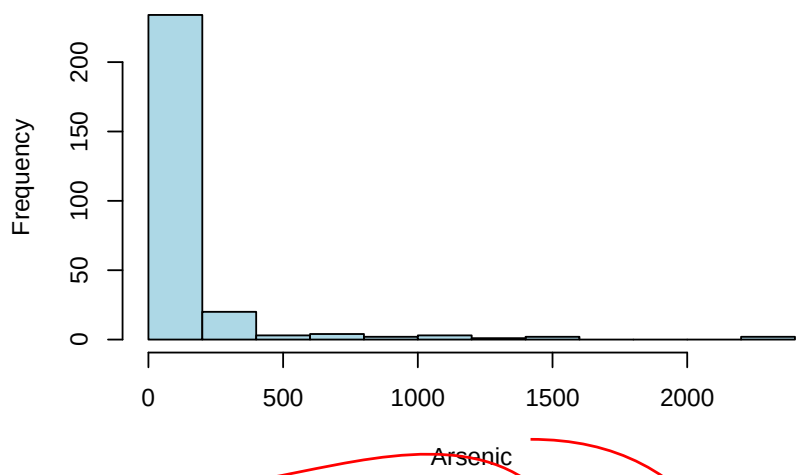


Figure 1b

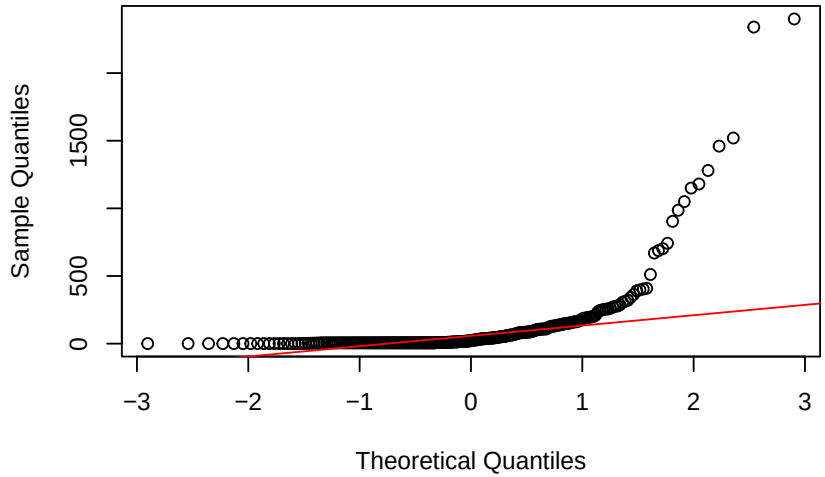


Figure 2a

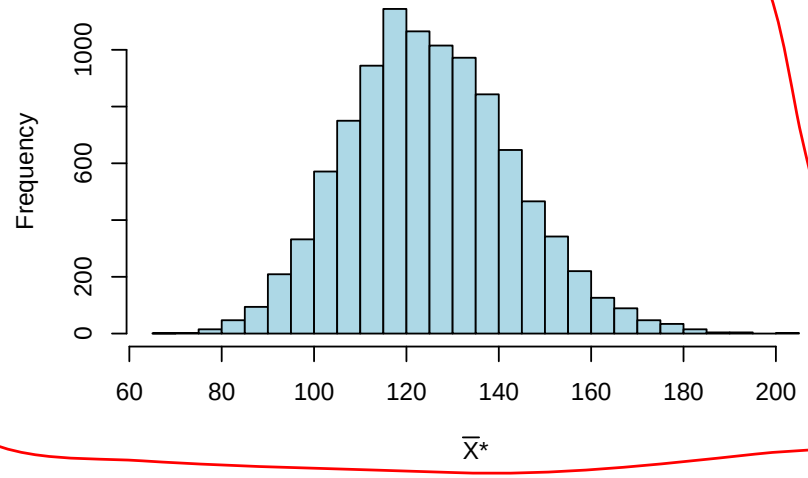
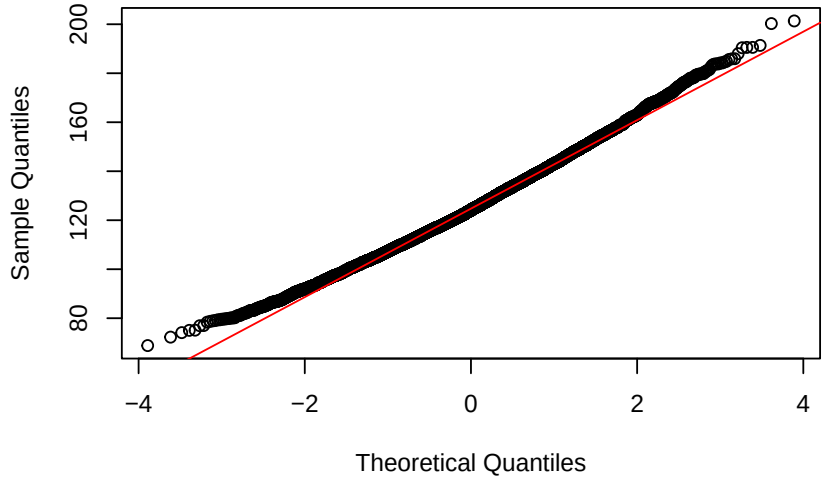


Figure 2b



Key features of the Bootstrap: Example 3

- Figures 2a and 2b show a histogram and a normal quantile plot of the bootstrap distribution, respectively.
- The bootstrap distribution looks quite normal, with some skewness.
- This is the central limit theorem at work—when the sample size is large enough, the sampling distribution for the mean is approximately normal, even if the population is not normal.

Section 5

Understanding confidence intervals

Understanding confidence intervals

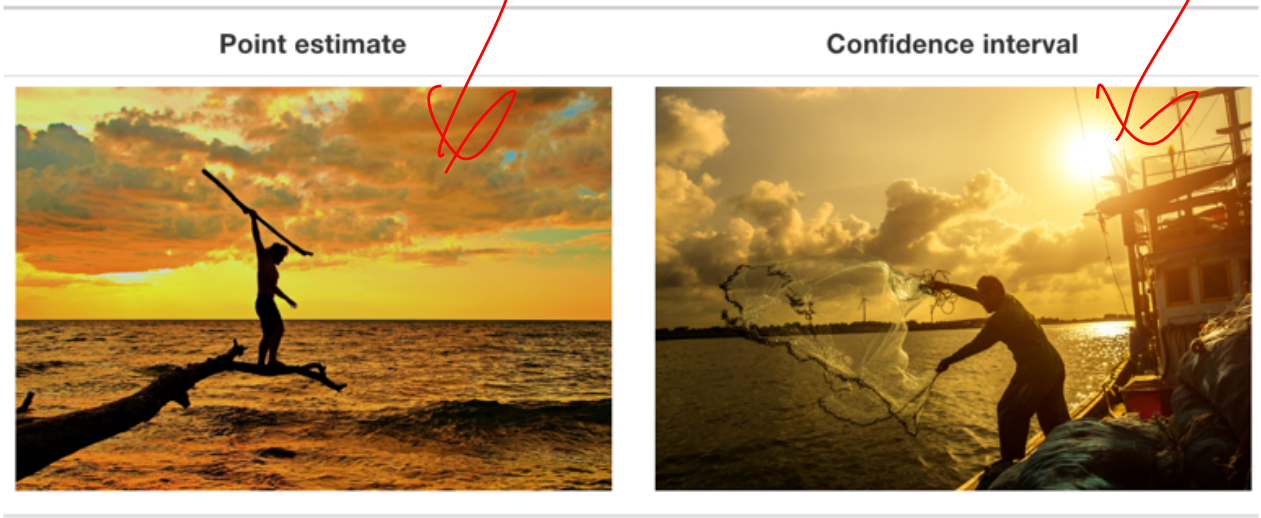
- Let's start this section with an analogy involving fishing. Say you are trying to catch a fish.
  - On one hand, you could use a spear, while on the other
  - you could use a net. Using the net will probably allow you to catch more fish!
- Now think back to our pennies exercise where you are trying to estimate the true population mean year μ of all US pennies.
  - Think of the value of μ as a fish.

Understanding confidence intervals

- On the one hand, we could use the appropriate point estimate/sample statistic to estimate μ, with the sample mean $\bar{x}$.
- Based on our sample of 50 pennies from the bank, the sample mean was 1995.44.
  - Think of using this value as "fishing with a spear."
- What would "fishing with a net" correspond to?
  - The bootstrap distribution.
  - Between which two years would you say that "most" sample means lie?
  - While this question is somewhat subjective, saying that most sample means lie between 1992 and 2000 would not be unreasonable.
  - Think of this interval as the "net."
- What we've just illustrated is the concept of a confidence interval, which we'll abbreviate with "CI".

Understanding confidence intervals

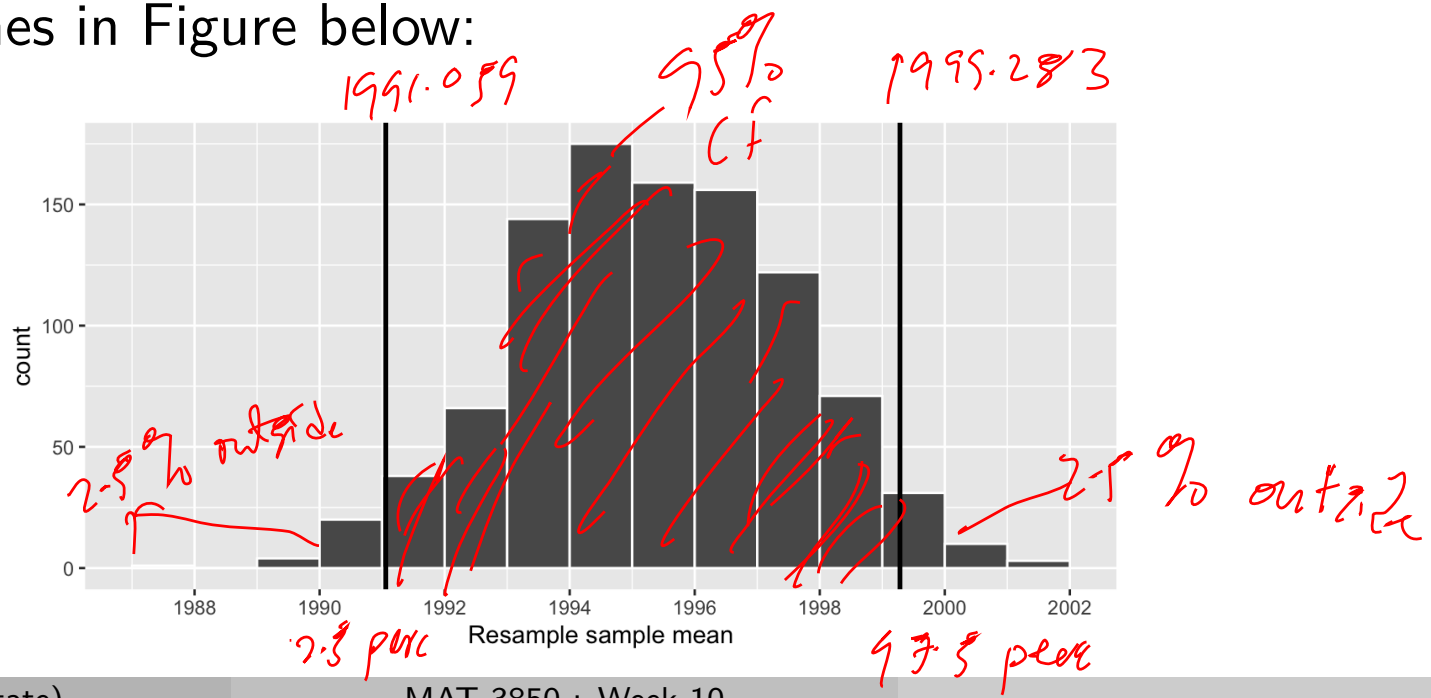
- As opposed to a point estimate/sample statistic that estimates the value of an unknown population parameter with a single value, a **confidence interval** gives what can be interpreted as a range of plausible values.
- Going back to our analogy, point estimates/sample statistics can be thought of as spears, whereas confidence intervals can be thought of as nets.



Percentile method

One method to construct a confidence interval

- To get the middle 95% of values of the bootstrap distribution:
  - We can do this by computing the 2.5th and 97.5th percentiles,
  - which are 1991.059 and 1999.283, respectively.
  - This is known as the percentile method for constructing confidence intervals.
- Let's mark these percentiles on the bootstrap distribution with vertical lines in Figure below:



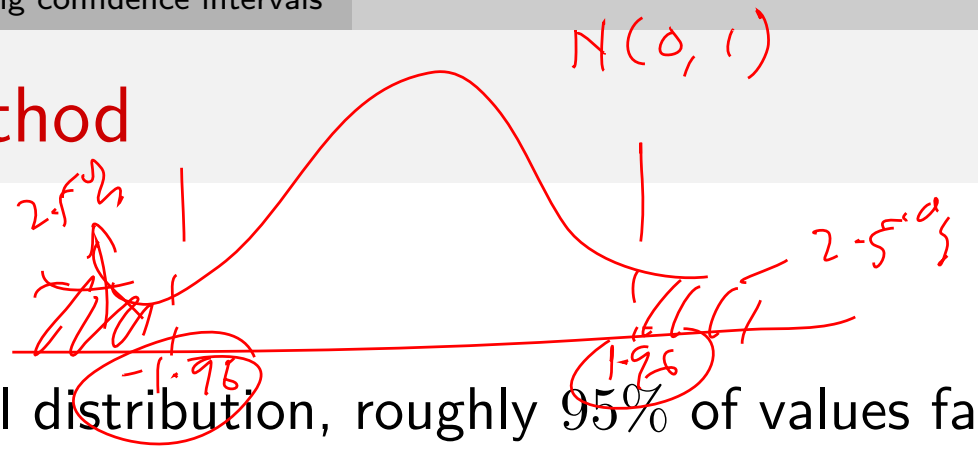
Standard error method

- Given that our bootstrap distribution based on 1000 resamples with replacement in Figure above is normally shaped,
  - let's use this fact about normal distributions to construct a confidence interval in a different way.
- First, recall the bootstrap distribution has a mean equal to 1995.36.
  - This value almost coincides exactly with the value of the sample mean $\bar{x}$ of our original 50 pennies of 1995.44.
- Second, let's compute the standard deviation of the bootstrap distribution using the values of `mean_year` in the `virtual_resampled_means` data frame:

```
virtual_resampled_means %>% summarize(SE = sd(mean_year))
```

```
## # A tibble: 1 x 1
##      SE
##   <dbl>
## 1  2.14
```

Standard error method



Recall that for a normal distribution, roughly 95% of values fall between ± 1.96 standard deviations of the mean.

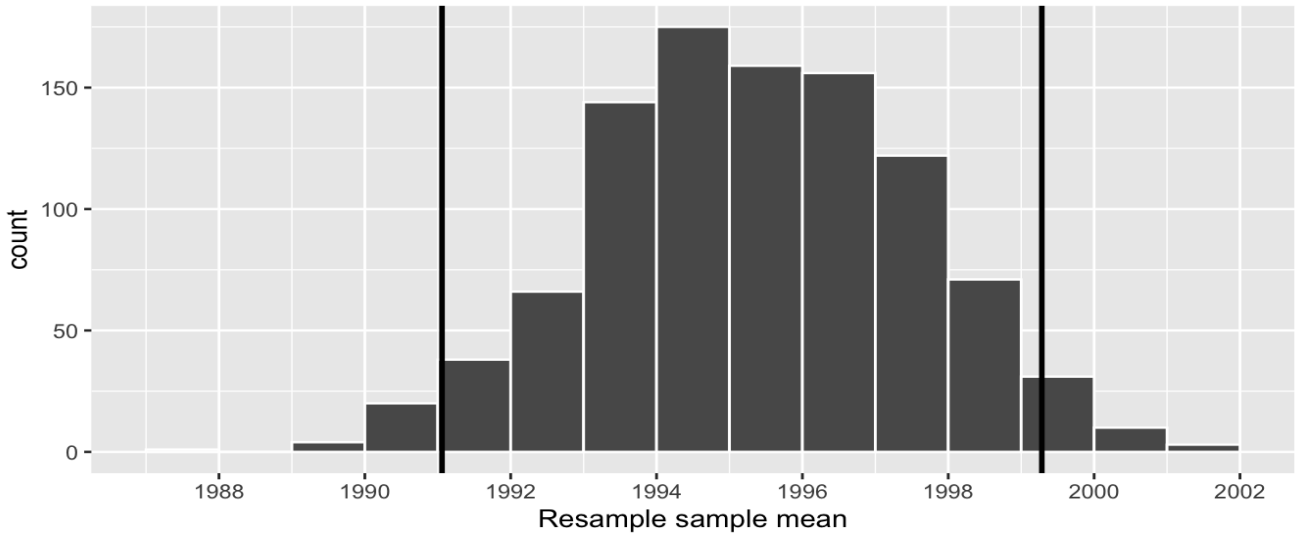
- Thus, using our 95% rule of thumb about normal distributions, we can use the following formula to determine the lower and upper endpoints of a 95% confidence interval for μ.

2-score

$$\begin{aligned} \bar{x} \pm 1.96 \cdot SE &= (\bar{x} - 1.96 \cdot SE, \bar{x} + 1.96 \cdot SE) \\ &= (1995.44 - 1.96 \cdot 2.15, \bar{x} + 1.96 \cdot 2.15) \\ &= (1991.15, 1999.73) \end{aligned}$$

1995.44

Standard error method vs. Percentile method



- We see that both methods produce nearly identical 95% confidence intervals for μ
  - with the percentile method yielding (1991.06,1999.28)
  - while the standard error method produces (1991.22,1999.66)
- However, we can only use the standard error rule when the bootstrap distribution is roughly normally shaped.

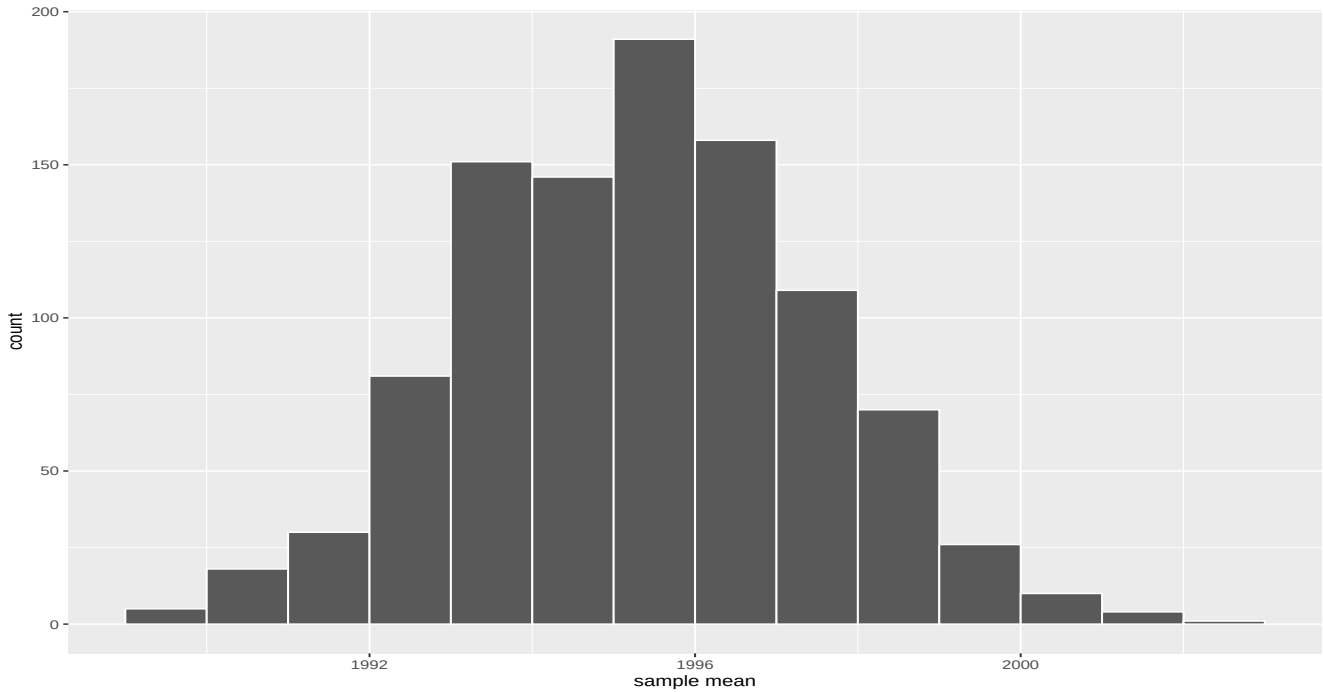
Section 6

Constructing confidence intervals

Percentile method example: Pennies Activity

Using `rep_sample_n` from the `infer` package.

```
set.seed(10)
virtual_resampled_means <- pennies_sample %>%
  rep_sample_n(size = 50, replace = TRUE, reps = 1000) %>%
  group_by(replicate) %>%
  summarize(mean_year = mean(year))
ggplot(virtual_resampled_means, aes(x = mean_year)) +
  geom_histogram(binwidth = 1, color = "white", boundary = 1990) +
  labs(x = "sample mean")
```



Percentile method example: Pennies Activity

Using `rep_sample_n` from the `infer` package.

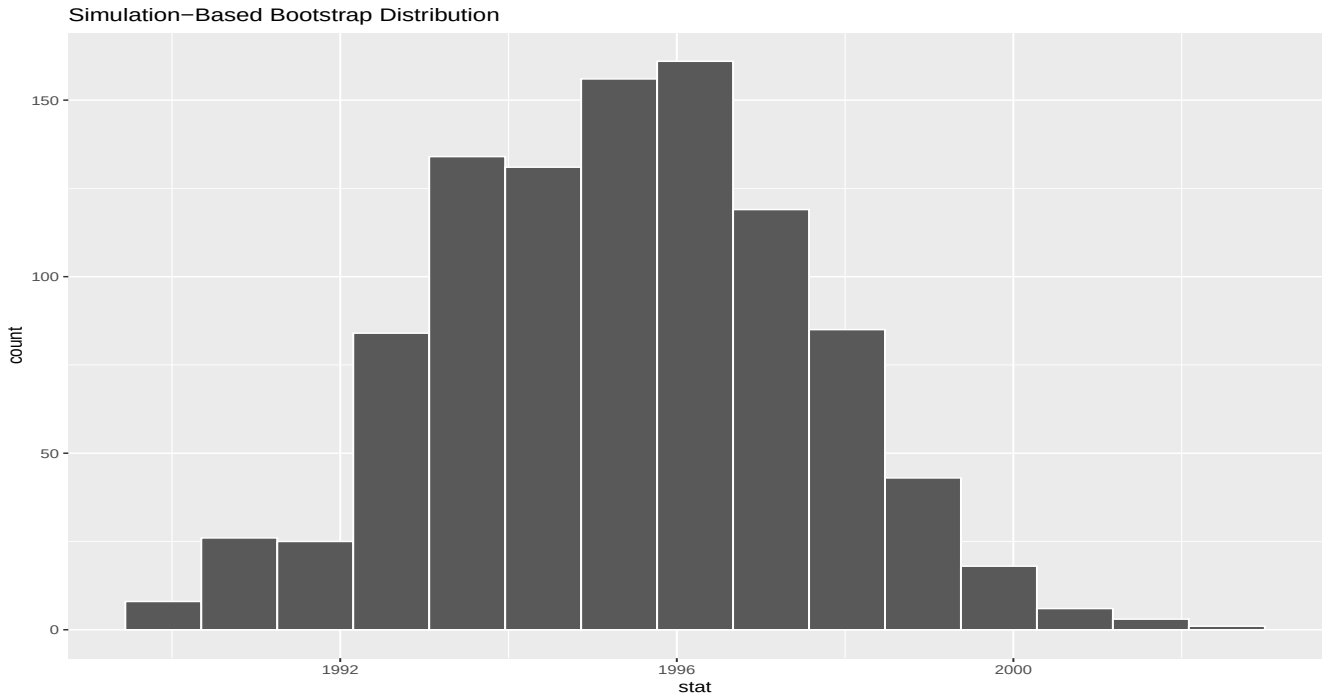
```
quantile(virtual_resampled_means$mean_year, prob = c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 1991.099 1999.460
```

Percentile method example: Pennies Activity

Using the `infer` pipeline

```
set.seed(10)
bootstrap_distribution <- pennies_sample %>%
  specify(response = year) %>%
  generate(reps = 1000) %>%
  calculate(stat = "mean")
visualize(bootstrap_distribution)
```



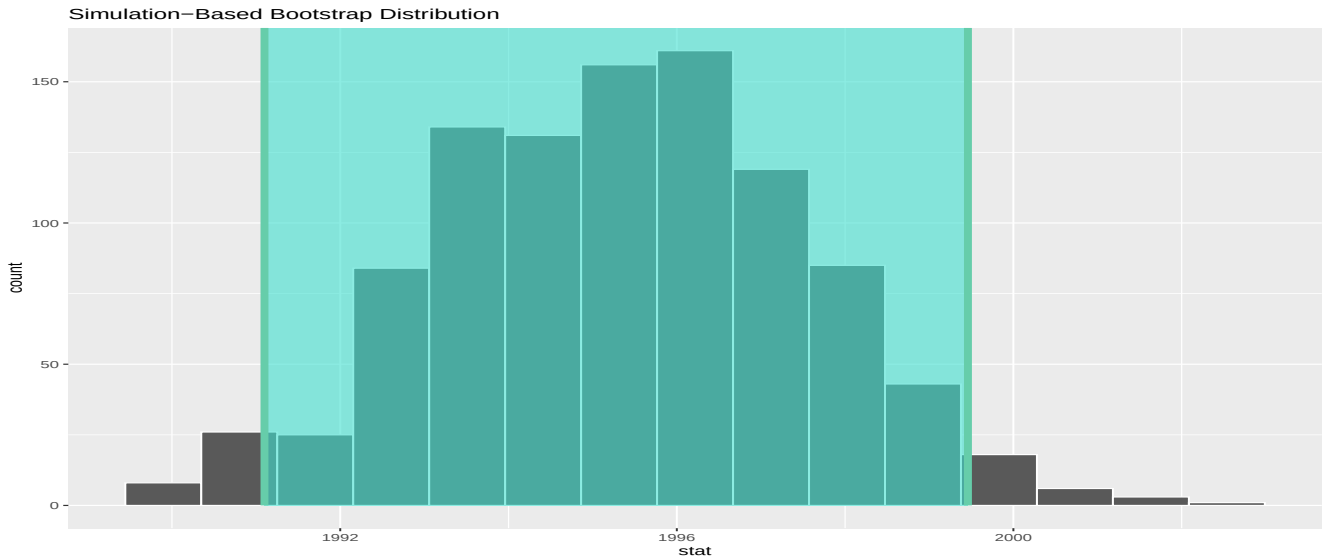
Percentile method example: Pennies Activity

Using the `infer` pipeline

```
percentile_ci <- bootstrap_distribution %>%
  get_confidence_interval(level = 0.95, type = "percentile")
percentile_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    1991.    1999.
```

```
visualize(bootstrap_distribution) +
  shade_confidence_interval(endpoints = percentile_ci)
```



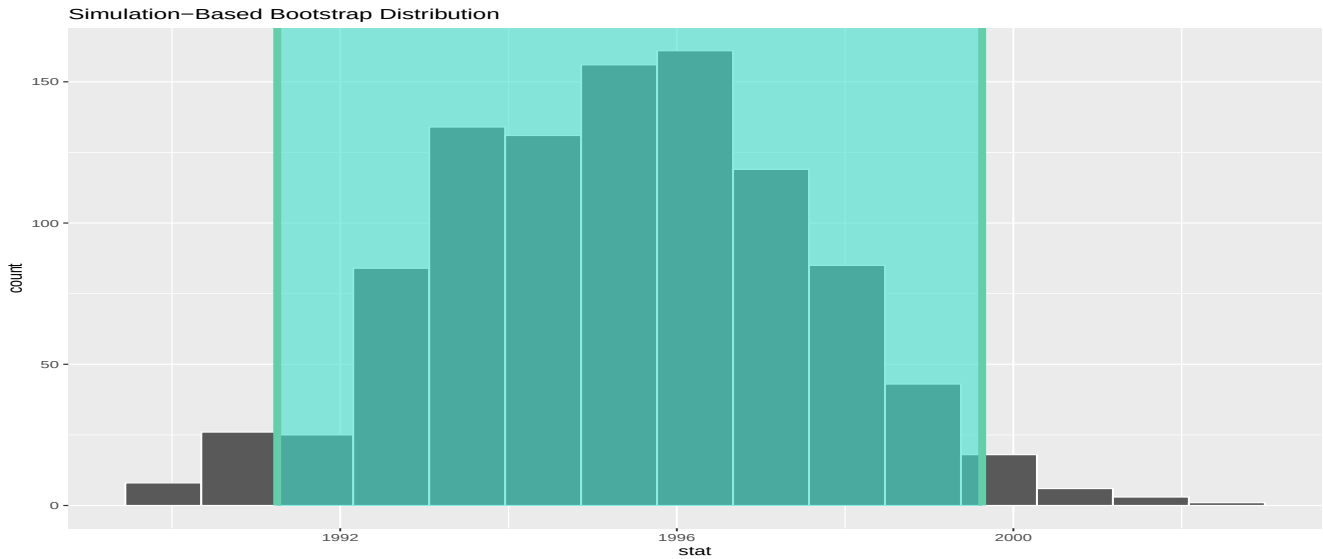
Standard error example: Pennies Activity

Using the `infer` pipeline

```
standard_error_ci <- bootstrap_distribution %>%
  get_confidence_interval(type = "se", point_estimate = x_bar, level = 0.95)
standard_error_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    1991.    2000.
```

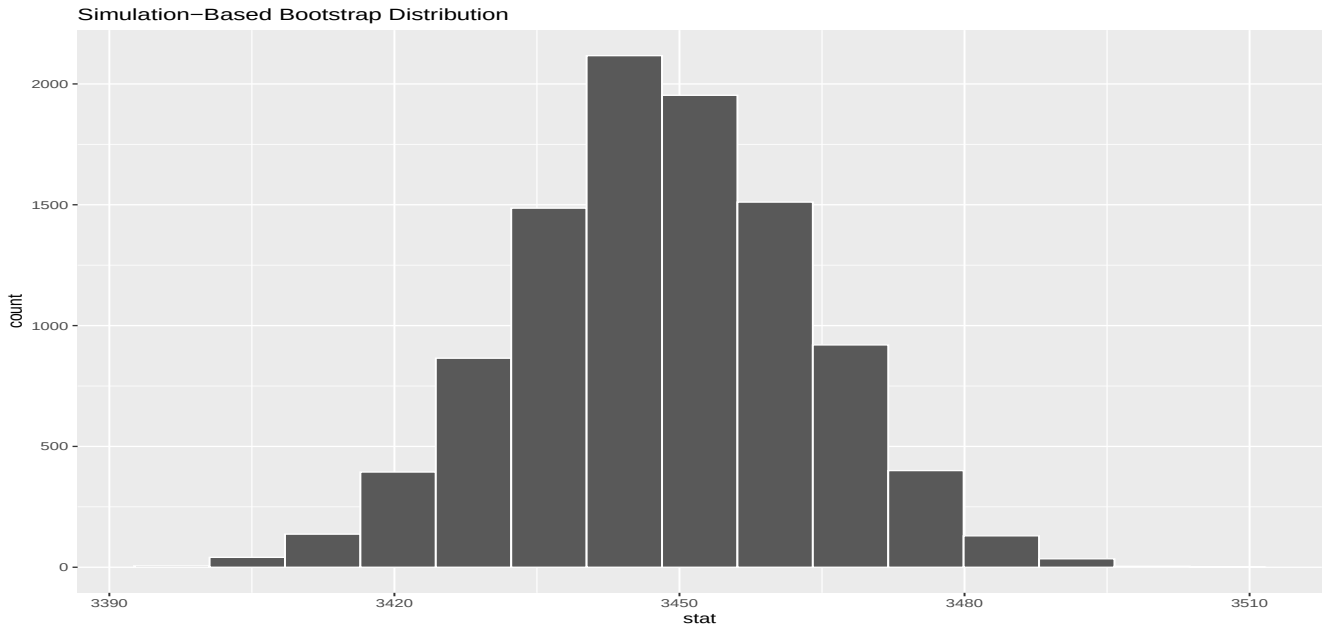
```
visualize(bootstrap_distribution) +
  shade_confidence_interval(endpoints = standard_error_ci)
```



Percentile method example: Birth weight of a baby

Using the `infer` pipeline

```
library(resampledata)
Babies <- NCBirths2004
set.seed(13)
bsd <- Babies %>%
  specify(response = Weight) %>%
  generate(reps = 10^4, type = "bootstrap") %>%
  calculate(stat = "mean")
visualize(bsd)
```



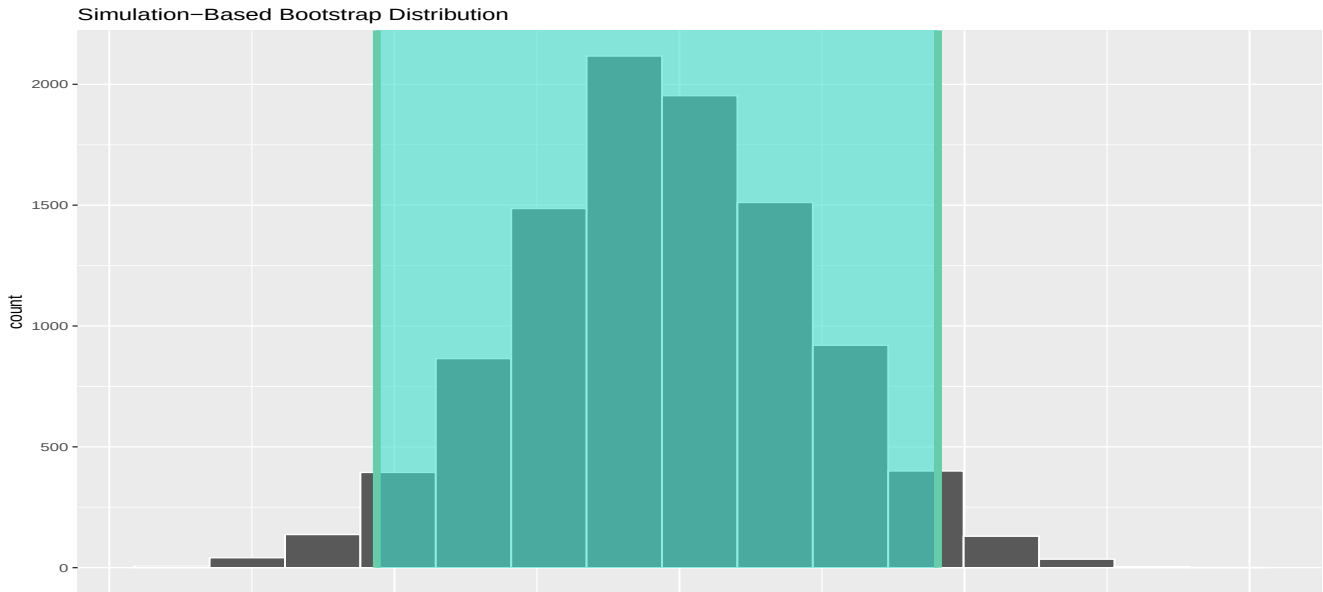
Percentile method example: Birth weight of a baby

Using the `infer` pipeline

```
percentile_ci <- bsd %>%
  get_confidence_interval(level = 0.95, type = "percentile")
percentile_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    3418.    3477.
```

```
visualize(bsd) +
  shade_confidence_interval(endpoints = percentile_ci)
```



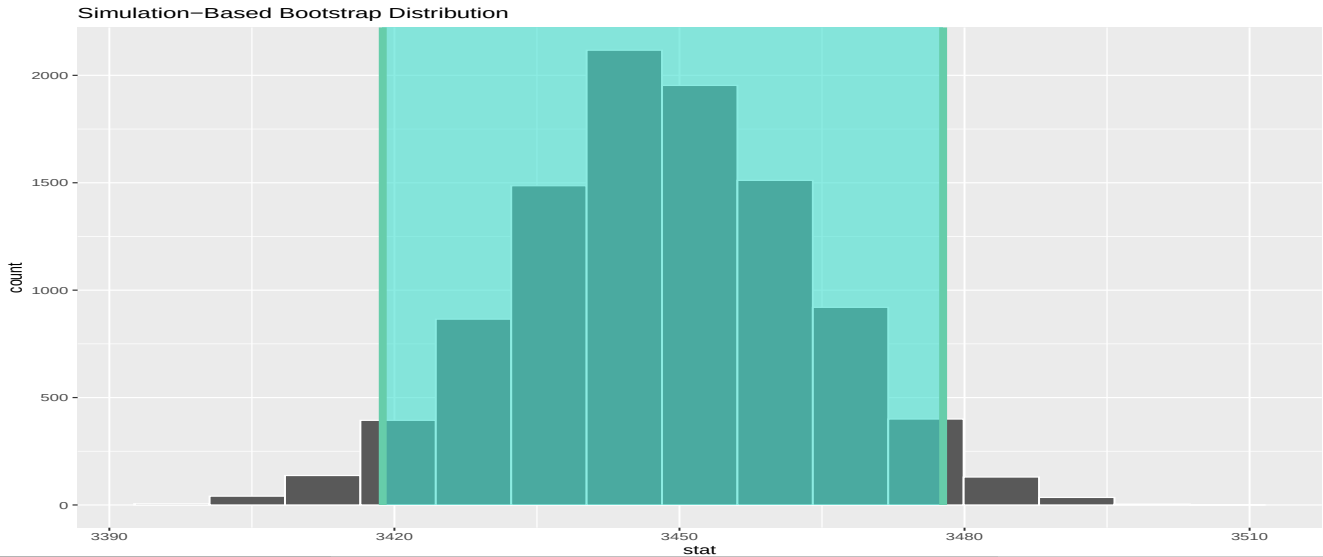
Standard error example: Birth weight of a baby

Using the `infer` pipeline

```
x_bar_babies<-Babies %>% summarize(Mean = mean(Weight))
standard_error_ci <- bsd %>%
  get_confidence_interval(type = "se", point_estimate = x_bar_babies, level = 0.95)
standard_error_ci
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1    3419.    3478.
```

```
visualize(bsd) +
  shade_confidence_interval(endpoints = standard_error_ci)
```



Section 7

Interpreting confidence intervals

Interpreting confidence intervals

- Now that we've shown you how to construct confidence intervals using a sample drawn from a population, let's now focus on how to interpret their effectiveness.
- The effectiveness of a confidence interval is judged by whether or not it contains the true value of the population parameter.
- Going back to our fishing analogy, this is like asking, "Did our net capture the fish?".
  - So, for example, does our percentile-based confidence interval of (1991.24, 1999.42) "capture" the true mean year μ of all US pennies?
  - Alas, we'll never know, because we don't know what the true value of μ is.
  - After all, we're sampling to estimate it!

Interpreting confidence intervals

- In order to interpret a confidence interval's effectiveness, we need to know what the value of the population parameter is.
  - That way we can say whether or not a confidence interval "captured" this value.
- Let's revisit our sampling bowl example. What proportion of the bowl's 2400 balls are red? Let's compute this:

```
bowl %>% summarize(p_red = mean(color == "red"))
```

```
## # A tibble: 1 x 1
##   p_red
##   <dbl>
## 1 0.375
```

In this case we know that the population proportion p is 0.375. In other words, we know that 37.5% of the bowl's balls are red.

Interpreting confidence intervals

- Recall that we had 33 groups of friends each take samples of size 50 from the bowl and then compute the sample proportion of red balls $\hat{p}$.
  - Let's use the `bowl_sample_1` data frame in the `moderndive` package

```
bowl_sample_1
```

```
## # A tibble: 50 x 1
##    color
##    <chr>
##  1 white
##  2 white
##  3 red
##  4 red
##  5 white
##  6 white
##  7 red
##  8 white
##  9 white
## 10 white
## # ... with 40 more rows
```

- They observed 21 red balls out of 50 and thus their sample proportion $\hat{p} = 21/50 = 0.42 = 42\%$.
  - Think of this as the "spear" from our fishing analogy.

Percentile method example: sampling bowl example

Using the `infer` pipeline

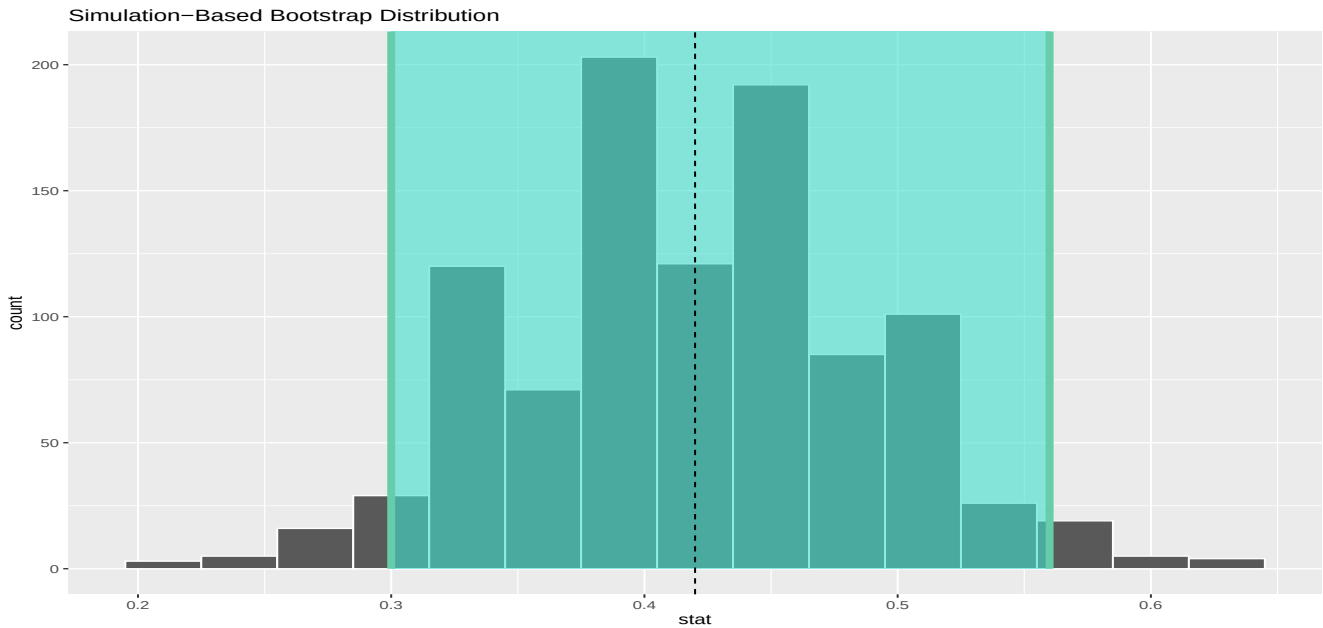
```
set.seed(10)
sample_1_bootstrap <- bowl_sample_1 %>%
  specify(response = color, success = "red") %>% ## need to specify the color
  generate(reps = 1000, type = "bootstrap") %>%
  calculate(stat = "prop")
percentile_ci_1 <- sample_1_bootstrap %>%
  get_confidence_interval(level = 0.95, type = "percentile")
percentile_ci_1
```

```
## # A tibble: 1 x 2
##   lower_ci upper_ci
##      <dbl>    <dbl>
## 1      0.3     0.56
```

Percentile method example: sampling bowl example

Using the `infer` pipeline

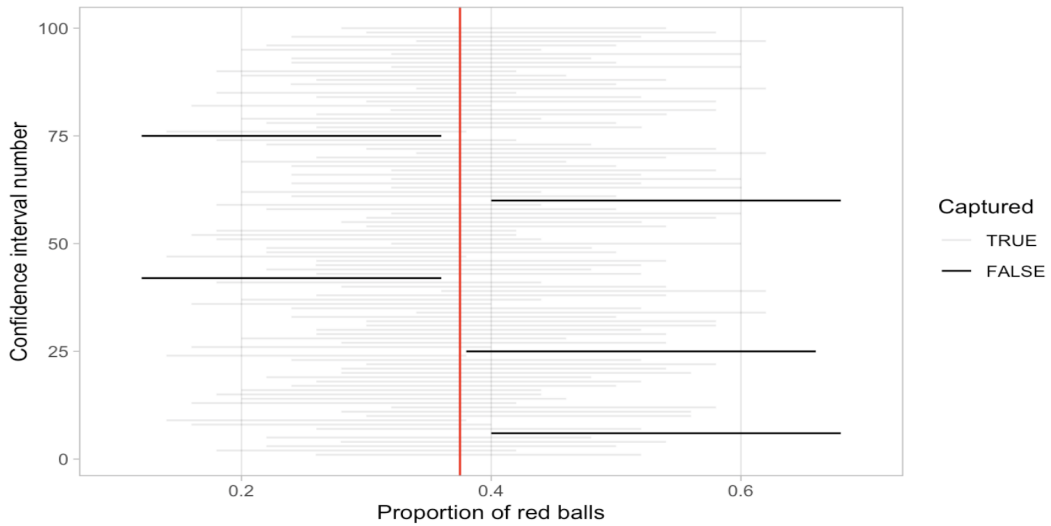
```
sample_1_bootstrap %>%
  visualize(bins = 15) +
  shade_confidence_interval(endpoints = percentile_ci_1) +
  geom_vline(xintercept = 0.42, linetype = "dashed")
```



In this case the 95% confidence interval for p, (0.3,0.56), based on their sample contain the true value of of 0.375.

Interpreting confidence intervals: Sampling bowl example

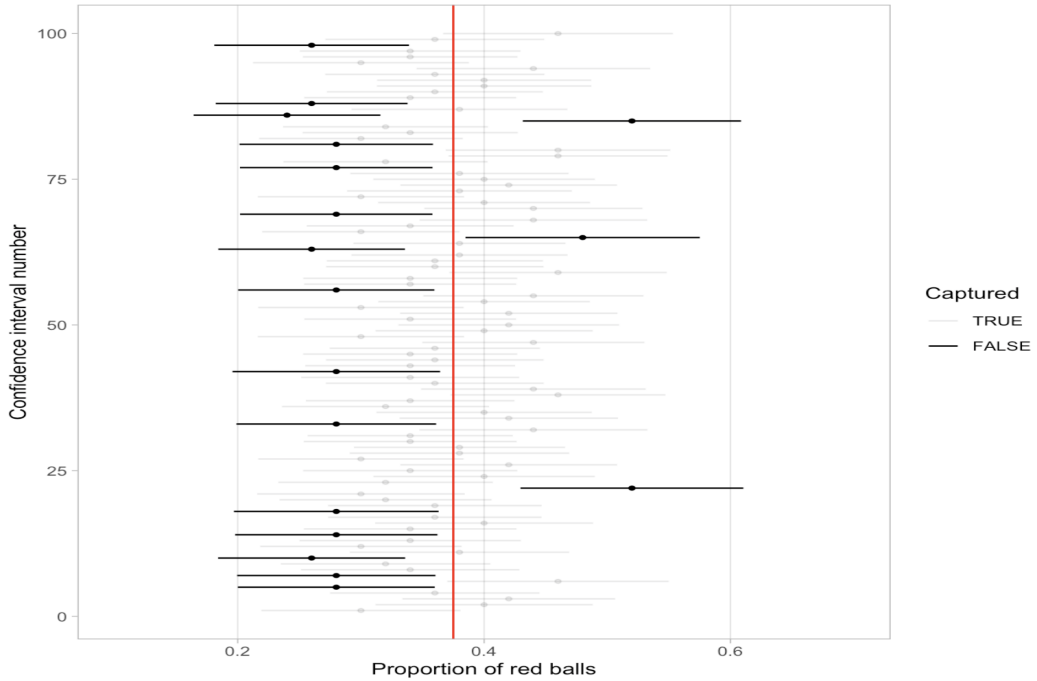
- However, will every 95% confidence interval for p capture this value?
- Let's now repeat this process 100 more times: we take 100 virtual samples from the bowl and construct 100 95% confidence intervals.



- Of the 100 95% confidence intervals, 95 of them captured the true value $p = 0.375$, whereas 5 of them didn't.
  - This is where the "95% confidence level" comes into play.

Interpreting confidence intervals: Sampling bowl example

- Now let's generate a figure similar to Figure above, but this time constructing 80% confidence intervals instead.



- Of the 100 80% confidence intervals, 82 of them captured the true value $p = 0.375$, whereas 18 of them didn't.

Interpreting confidence intervals

The precise and mathematically correct interpretation of a 95% confidence interval is a little long-winded:

> Precise interpretation: If we repeated our sampling procedure a large number of times, we expect about 95% of the resulting confidence intervals to capture the value of the population parameter.

The shorthand summary of the precise interpretation:

> Short-hand interpretation: We are 95% "confident" that a 95% confidence interval captures the value of the population parameter.

Width of confidence intervals

Let's go over some factors that determine their width.

1. Impact of confidence level.
2. Impact of sample size.