Bootstrap Confidence Intervals

Corinne Riddell (Instructor: Tomer Altman)

November 5, 2025

Confidence interval recap

- So far we've (mostly) been using formulas of the form estimate $\pm$ margin of error to generate confidence intervals
- We've applied these formulas to the mean $\bar{x}$ and the proportion $\hat{p}$. These "large sample" procedures use the Central Limit Theorem to calculate the standard error of the sampling distribution for $\bar{x}$ and $\hat{p}$

Confidence interval recap

- We did this when the data met certain assumptions, including having an SRS and that either the data is normally distributed or the sample size is large enough that the CLT kicks in
- Often times, we're stuck if the sample size is too small (though we saw alternative methods when n is small for the proportion)
- **We need a procedure to calculate confidence intervals when the sample size is small and when we can't assume that the CLT kicks in**
- Bootstrapping is still a "large sample" method, but can sometimes perform better than standard methods if the sample size is small and the sampling distribution not perfectly symmetric

Confidence intervals for other parameters

- Bootstrapping can be also very useful when analytically deriving a confidence interval is difficult or impossible
- For instance, what if we wanted a confidence interval for the median? Or a confidence interval for the first quartile, Q_1, or some other parameter?
- We couldn't use the regular formula for calculating the confidence interval because we don't know how to calculate the standard error for these other parameters
- **We need a procedure to calculate confidence intervals for other parameters, such as the median or any other quartile or percentile. That is, this procedure can calculate CIs when we don't know the formula for the standard error.**

Enter the Bootstrap Confidence Interval

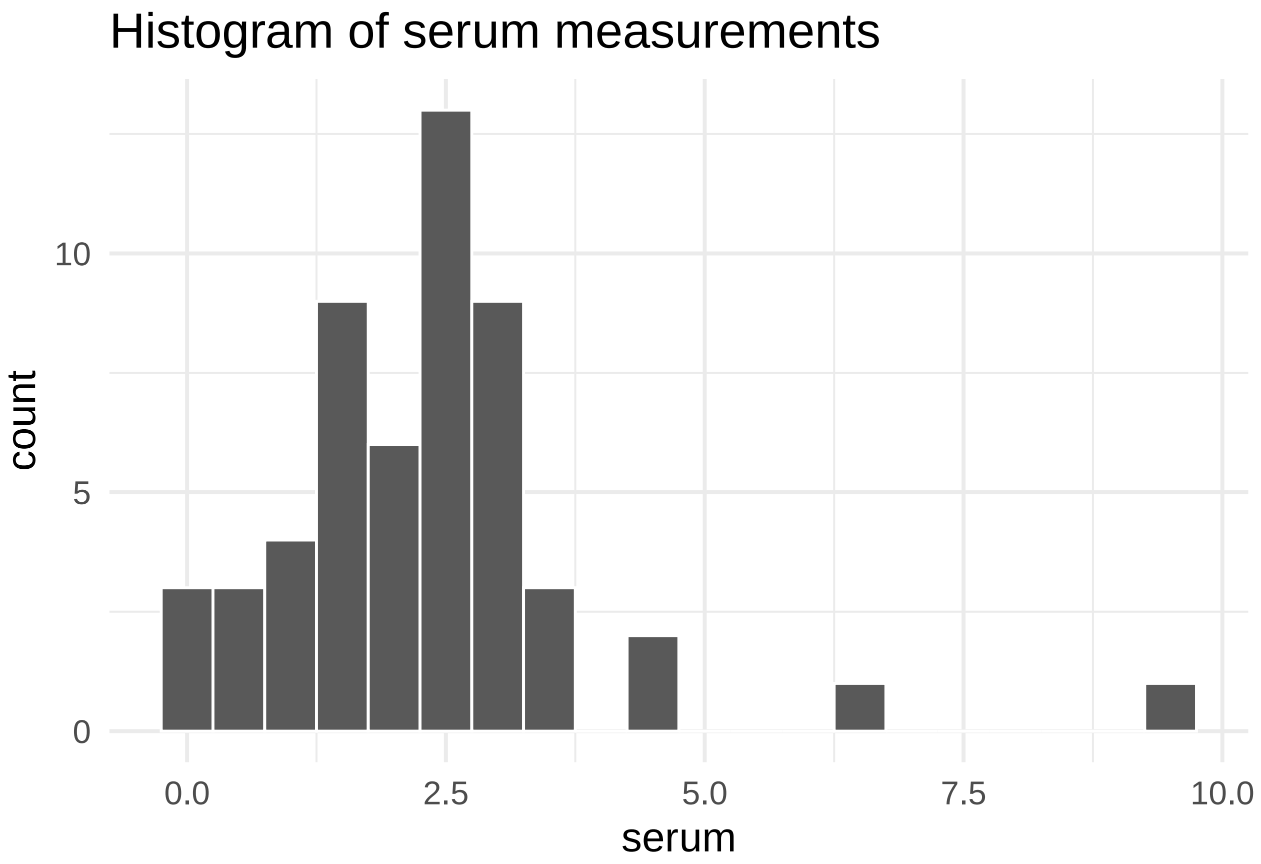
- Depending on the type of bootstrap CI used, the CI can still work well (in some cases) when the sampling distribution of our estimator is not Normally distributed
- The Bootstrap CI can be made for almost any parameter (based on its sample statistic)
- It was so-named because it uses only the information from the sample you have to estimate the CI, such that the sample is "pulling itself up by its own bootstraps"
  - The idiom describes an impossible task, but also means to achieve a goal solely through one's own efforts

- This method gained popularity in 1979 when popularized by Bradley Efron, and harnesses current computing power
- **It is a very popular method used to compute confidence intervals today**

Bootstrap example

This example came from bootstrap's advocate, Bradley Efron. Suppose we have measures of serum from 54 patients:

```
serum_data <- data.frame(serum = c(0.1, 0.1, 0.2, 0.4, 0.4, 0.6,
                                   0.8, 0.8, 0.9, 0.9, 1.3, 1.3,
                                   1.4, 1.5, 1.6, 1.6, 1.7, 1.7,
                                   1.7, 1.8, 2.0, 2.0, 2.2, 2.2,
                                   2.2, 2.3, 2.3, 2.4, 2.4, 2.4,
                                   2.4, 2.4, 2.4, 2.5, 2.5, 2.5,
                                   2.7, 2.7, 2.8, 2.9, 2.9, 2.9,
                                   3.0, 3.1, 3.1, 3.2, 3.2, 3.3,
                                   3.3, 3.5, 4.4, 4.5, 6.4, 9.4))
```



First, calculate the theory-based 95% CI

Using the skills we already know, we can calculate the **95% CI for the mean** for these data using `dplyr`

```
## find t* to use to calculate the 95% CI
t_star <- qt(0.975, df = 53)

serum_data %>% summarise(mean_serum = mean(serum),
                         se_serum = sd(serum)/sqrt(n()),
                         lower_CI = mean_serum - t_star * se_serum,
                         upper_CI = mean_serum + t_star * se_serum)
```

```
##   mean_serum  se_serum lower_CI upper_CI
## 1   2.318519 0.2078991 1.901526 2.735511
```

Or by using the `t.test()` function:

```
t.test(serum_data %>% pull(serum))
```

```
##
##  One Sample t-test
##
## data:  serum_data %>% pull(serum)
## t = 11.152, df = 53, p-value = 1.62e-15
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  1.901526 2.735511
## sample estimates:
## mean of x
##  2.318519
```

The 95% CI provides our best guess of where the true proportion lies. The 95% CI for μ is 1.90 to 2.74. We found this interval using a method that gives an interval that captures μ 19 times out of 20.

95% CI for the median

- There is a formula for getting the SE of the median, and the CLT will apply if sample size is big enough, but the standard error is complicated
- However, we can use the bootstrap and avoid complicated standard error calculations that come with its own estimation problem
  - Can itself be a poor estimate of the true standard deviation of the sampling distribution of the median

How the Bootstrap CI is made

- If we truly have an SRS from the underlying population, this means that the distribution of serum in the sample should *approximate* the distribution of serum in the population
- Graphically, this means the shape of the histogram for the sample data should approximate the shape of the density plot for the entire population
- **The key: If we take repeated samples (with replacement) from our sample, we can approximate the sampling distribution for any statistic we'd like**
- This is the process of bootstrapping
- Let's apply this method to calculate the 95% CI for the median using the serum data

Check your understanding!

Boostrap confidence interval for the median

1. Calculate the median for the original sample of size 54. Denote this value by m. This is our estimate of the median for the underlying population. We need to create a 95% CI around m.

```
median_serum <- serum_data %>%
  summarise(median_serum = median(serum))
```

```
median_serum
```

```
##   median_serum
## 1          2.35
```

2. Resample **with replacement** from the original sample a new sample, also of size 54.
3. Calculate the median based on resample #1. Call this median m_1^*.

Boostrap confidence interval for the median

4. Resample again. Calculate the median based on resample #2. Call this median m_2^*. … repeat this resampling procedure several thousand times.
5. Make a histogram of $m_1^*, m_2^*, \ldots, m_{1000}^*$. **This histogram approximates the sampling distribution for the median**
6. Calculate the bounds such that the middle 95% of the observations are between the lower and upper bounds. In R, we can do this using `quantile(sample_median, 0.025)` and `quantile(sample_median, 0.975)` to locate the 2.5th and 97.5th percentiles of the variable `sample_median`.

Boostrap confidence interval for the median

```r
# students, you don't need to know how this code works.

number_of_bootstraps <- 10000
sample_size <- 54

calc_sample_stats <- function(df) {
  df %>% summarize(sample_median = median(serum))
}

set.seed(1)

many_sample_medians <- replicate(number_of_bootstraps,
                                 sample_n(serum_data,
                                          sample_size,
                                          replace = T),
                                 simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
```

This code resamples the data 1000 times and calculates the median for each resample. It stores the median in a data frame called `many_sample_medians`.

```r
head(many_sample_medians)
```

```
##   sample_median sample.id
## 1           2.4         1
## 2           2.4         2
## 3           2.5         3
## 4           2.4         4
## 5           2.4         5
## 6           2.4         6
```

Calculate the lower and upper bounds of the 95% bootstrap CI

```r
# Understand this code. It takes the data frame containing the bootstrap
# sample medians and calculates the lower and upper CI using the quantile function.
```

```r
bounds <- many_sample_medians %>%
  summarise(lower_CI = quantile(sample_median, 0.025),
            upper_CI = quantile(sample_median, 0.975))

bounds
```

```
##   lower_CI upper_CI
## 1      1.9      2.5
```

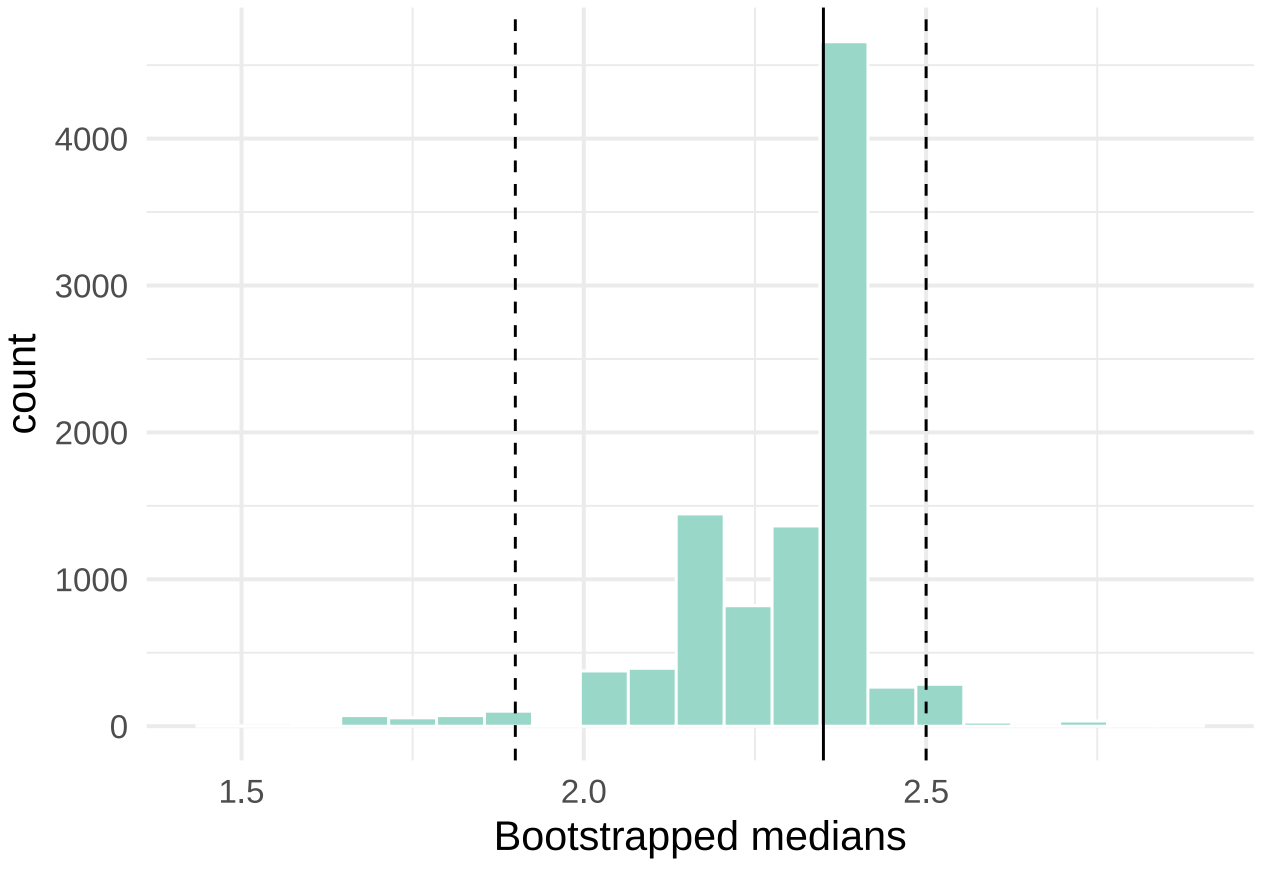
Thus, our best estimate of the median is 2.35. The bootstrapped 95% confidence interval for the median is 1.9 to 2.5.

How would the code change if we were interested in the 99% CI?

Plot the histogram of the bootstrapped medians and denote the 95% confidence interval

```r
fig <- ggplot(many_sample_medians, aes(x = sample_median)) +
  geom_histogram(binwidth = 0.07, col = "white", fill = "#99d8c9") +
  labs(x = "Bootstrapped medians") +
  geom_vline(aes(xintercept = median_serum %>% pull())) +
  theme_minimal(base_size = 15) +
  geom_vline(aes(xintercept = bounds %>% pull(lower_CI)), lty = 2) +
  geom_vline(aes(xintercept = bounds %>% pull(upper_CI)), lty = 2)
```

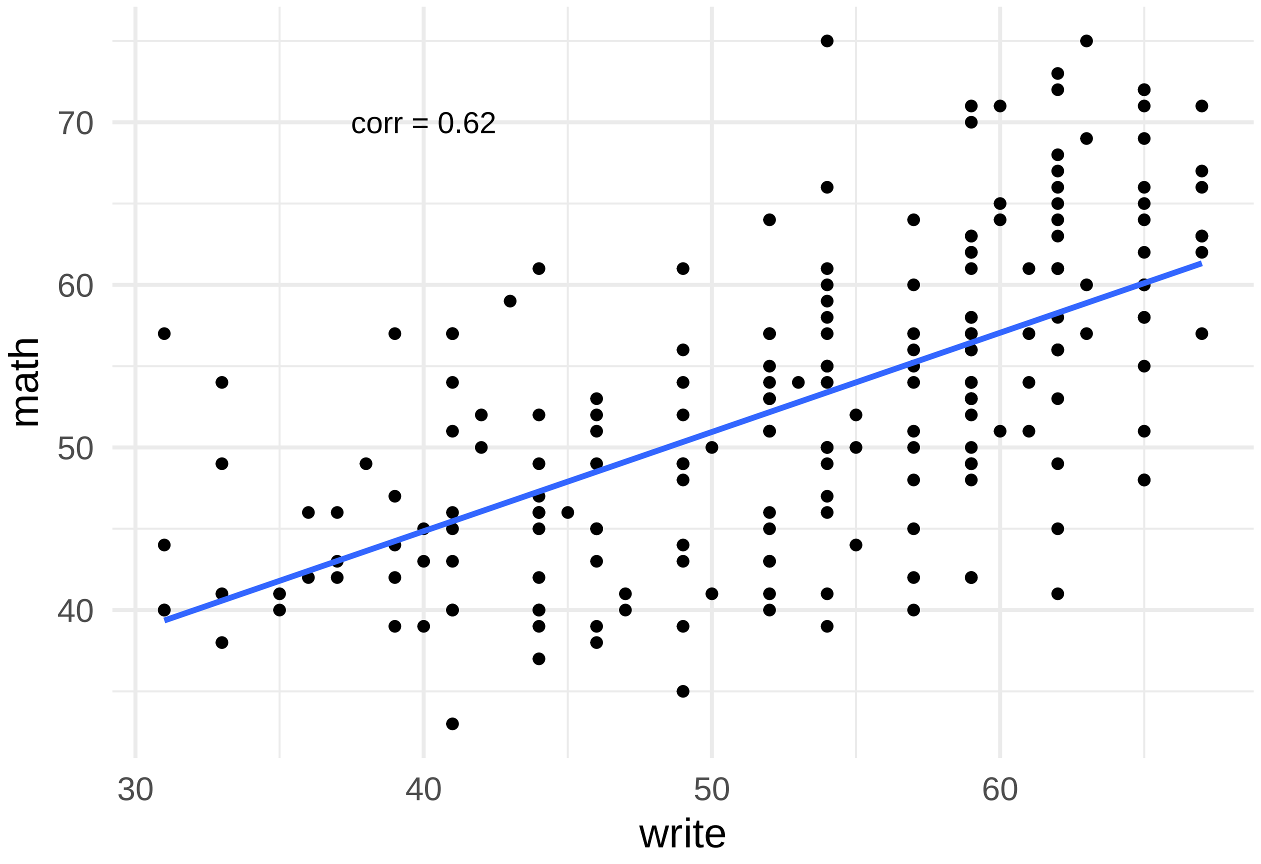
- Note that the sampling distribution for the median is not symmetric. It is skewed left (for these data).
- Note, the CI is not symmetric around our estimate of the sample median (2.35)
- How to examine whether this procedure would result in a CI with 95% coverage?



Another example: Calculate the bootstrap CI for the correlation coefficient

You have data on student demographics and scores across multiple subjects. You are interested in the correlation between the scores for writing (`write`) and mathematics (`math`).

```
hsb2 <- read.table("./data/hsb2.csv", sep=",", header=T)

fig <- ggplot(data = hsb2, aes(write, math)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = F) +
  annotate("text", x = 40, y = 70,
           label = "corr = 0.62", check_overlap = T) +
  theme_minimal(base_size = 15)
```



Calculate the bootstrapped CI for the correlation

1. Calculate the correlation coefficient for the original sample of size 200. Denote this value by r. This is our estimate of the correlation coefficient for the underlying population. We need to create a 95% CI around r.

```
corr_math_write <- hsb2 %>%
  summarise(correlation = cor(write,
                              math))
```

```
corr_math_write
```

```
##   correlation
## 1   0.6174493
```

2. Resample with replacement from the original sample a new sample, also of size 200.

3. Calculate the correlation coefficient based on resample #1. Call this correlation r_1^*.

Calculate the bootstrapped CI for the correlation

4. Resample again. Calculate the correlation coefficient based on resample #2. Call this correlation coefficient r_2^*. ... repeat this resampling procedure several thousand times.
5. Make a histogram of $r_1^*, r_2^*, \ldots, r_{10000}^*$. **This histogram approximates the sampling distribution for the correlation coefficient**
6. Calculate the bounds such that the middle 95% of the observations are between the lower and upper bounds. In R, we can do this using `quantile(sample_corr, 0.025)` and `quantile(sample_corr, 0.975)` to locate the 2.5th and 97.5th percentiles of the variable `sample_corr`.

Calculate the bootstrapped CI for the correlation, continued

```r
# students, you don't need to know how this code works.

number_of_bootstraps <- 1000
sample_size <- 200

calc_sample_stats <- function(df) {
  df %>% summarize(sample_corr = cor(math, write))
}

set.seed(1)

many_sample_correlations <- replicate(number_of_bootstraps,
                                      sample_n(hsb2,
                                               sample_size,
                                               replace = T),
                                      simplify = F) %>%
  lapply(., calc_sample_stats) %>%
  bind_rows() %>%
  mutate(sample.id = 1:n())
```

```r
head(many_sample_correlations)
```

```
##   sample_corr sample.id
## 1   0.6032971         1
## 2   0.6020023         2
## 3   0.6171369         3
## 4   0.6083906         4
## 5   0.7292170         5
## 6   0.6268646         6
```

Calculate the lower and upper bounds of the 95% bootstrap CI for the correlation coefficient

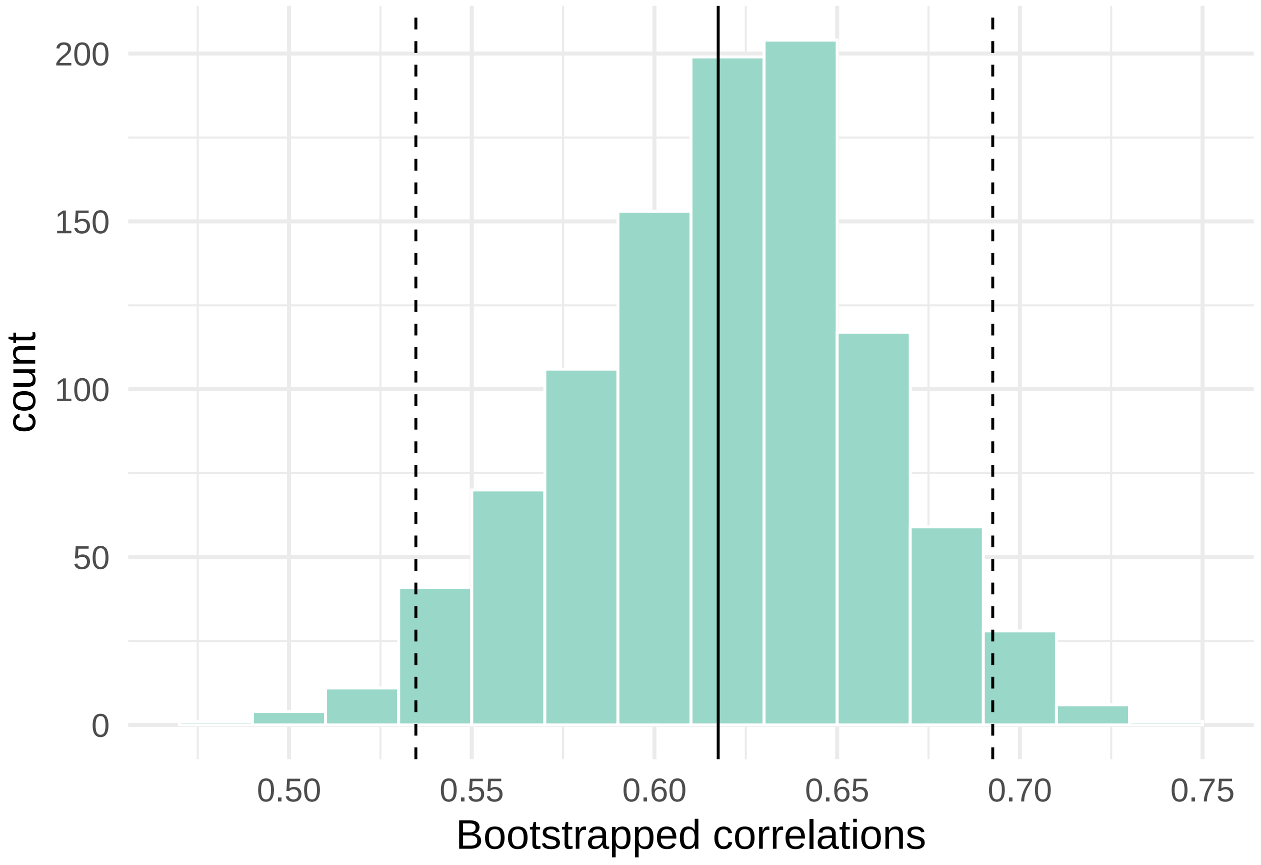
```r
# Understand this code. It takes the data frame containing the bootstrap
# sample medians and calculates the lower and upper CI using the quantile function.

bounds <- many_sample_correlations %>%
  summarise(lower_CI = quantile(sample_corr, 0.025),
            upper_CI = quantile(sample_corr, 0.975))

bounds
```

```
##    lower_CI  upper_CI
## 1 0.5347083 0.6925803
```

How would the code change if we were interested in the 99% CI?

Plot the histogram of the bootstrapped correlations and show the 95% CI

```
fig <- ggplot(many_sample_correlations, aes(x = sample_corr)) +
  geom_histogram(binwidth = 0.02, col = "white", fill = "#99d8c9") +
  labs(x = "Bootstrapped correlations") +
  geom_vline(aes(xintercept = corr_math_write %>% pull())) +
  theme_minimal(base_size = 15) +
  geom_vline(aes(xintercept = bounds %>% pull(lower_CI)), lty = 2) +
  geom_vline(aes(xintercept = bounds %>% pull(upper_CI)), lty = 2)
```



Summary on Bootstrap CIs

- The bootstrap is a method that we use to calculate confidence intervals
- It is particularly useful when:
  - We don't know what the formula is
  - We don't have a nice formula to calculate the CI
  - The formula is a biased estimator
  - The sample size is too small to apply the CLT
  - A Normal distribution might not fit the true sampling distribution well
- We can make bootstrap CIs around **any** statistic we've learned about: the median, the quartiles, the correlation coefficient, etc.

Extra information (not tested; for future reference)

- The type of bootstrap I've shown you is called the **percentile bootstrap**
- It is the most intuitive method but has some drawbacks that we don't have time to discuss
- Two other methods are known as:
  i) the **basic or empirical bootstrap**, and
  ii) the **bias-corrected and accelerated bootstrap**

Formulas for basic bootstrap (not tested; for future reference)

Here is the code for the basic/empirical bootstrap. This method also depends on the sample estimate for the correlation coefficient `corr_math_write`:

```r
#lower bound of 95% CI
#note that it "seems" flipped because we take the 97.5th percentile to get the lower bound
2*corr_math_write - quantile(many_sample_correlations$sample_corr, 0.975)
```

```
##   correlation
## 1   0.5423182
```

```r
#upper bound of 95% CI
2*corr_math_write - quantile(many_sample_correlations$sample_corr, 0.025)
```

```
##   correlation
## 1   0.7001902
```

How does the CI compare to the one using the percentile method?

Code for bias-corrected and accelerated (BCA) bootstrap (not tested; for future reference)

```r
#install.packages("coxed")
library(coxed)
bca(many_sample_correlations$sample_corr)
```

```
## [1] 0.5309652 0.6903872
```

How does the CI compare to the one using the percentile method?

Check your understanding!