

# The bootstrap revisited

## Packages for this section

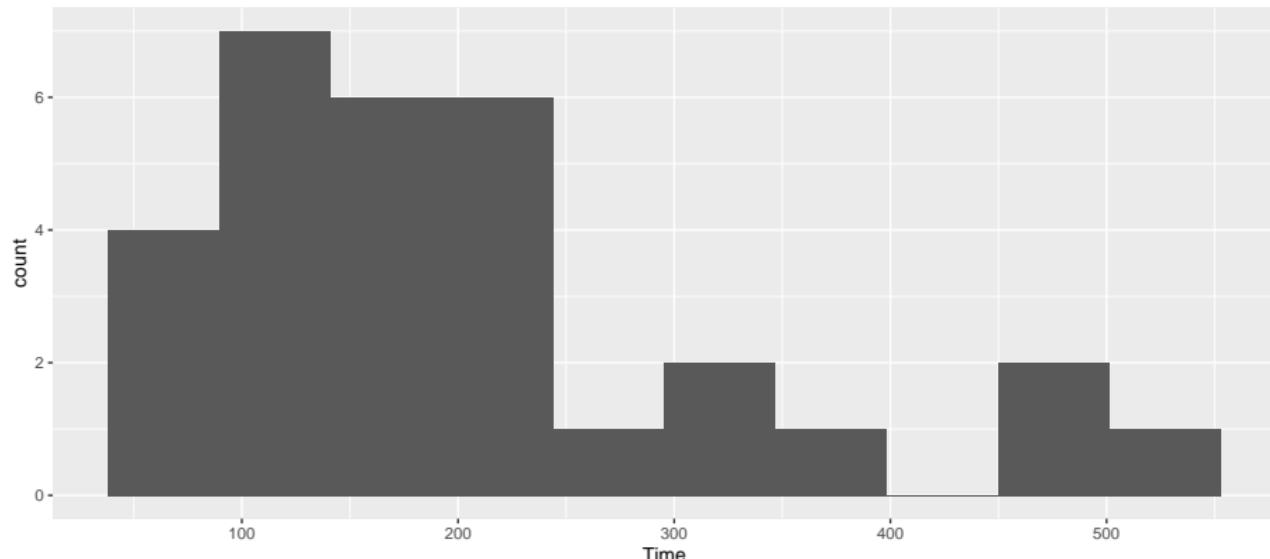
```
library(tidyverse)
library(bootstrap)
```

Source: [Hesterberg et al](#)

## Is my sampling distribution normal enough?

- Recall IRS data (used as a motivation for the sign test) :

```
ggplot(irs, aes(x=Time))+geom_histogram(bins=10)
```



- $t$  procedure for the mean would not be a good idea because the distribution is skewed.

## What *actually* matters

- It's not the distribution of the *data* that has to be approx normal (for a *t* procedure).
- What matters is the *sampling distribution of the sample mean*.
- If the sample size is large enough, the sampling distribution will be normal enough even if the data distribution is not.
  - ▶ This is why we had to consider the sample size as well as the shape.
- But how do we know whether this is the case or not? We only have *one* sample.

## The (nonparametric) bootstrap

- Typically, our sample will be reasonably representative of the population.
- Idea: pretend the sample *is* the population, and sample from it *with replacement*.
- Calculate test statistic, and repeat many times.
- This gives an idea of how our statistic might vary in repeated samples: that is, its sampling distribution.
- Called the **bootstrap distribution** of the test statistic.
- If the bootstrap distribution is approx normal, infer that the true sampling distribution also approx normal, therefore inference about the mean such as  $t$  is good enough.
- If not, we should be more careful.

## Why it works

- We typically estimate population parameters by using the corresponding sample thing: eg. estimate population mean using sample mean.
- This called **plug-in principle**.
- The fraction of sample values less than a value  $x$  called the **empirical distribution function** (as a function of  $x$ ).
- By plug-in principle, the empirical distribution function is an estimate of the population CDF.
- In this sense, the sample *is* an estimate of the population, and so sampling from it is an estimate of sampling from the population.

## Bootstrapping the IRS data

- Sampling with replacement is done like this (the default sample size is as long as the original data):

```
boot <- sample(irs$Time, replace=T)  
mean(boot)
```

[1] 201.7333

- That's one bootstrapped mean. We need a whole bunch.

## A whole bunch

- Use the same idea as for simulating power:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(boot_sample = list(sample(irs$Time, replace = TRUE)))
```

```
# A tibble: 1,000 x 2
```

```
# Rowwise:
```

```
    sim boot_sample
  <int> <list>
 1     1 <dbl [30]>
 2     2 <dbl [30]>
 3     3 <dbl [30]>
 4     4 <dbl [30]>
 5     5 <dbl [30]>
 6     6 <dbl [30]>
 7     7 <dbl [30]>
 8     8 <dbl [30]>
 9     9 <dbl [30]>
10    10 <dbl [30]>
```

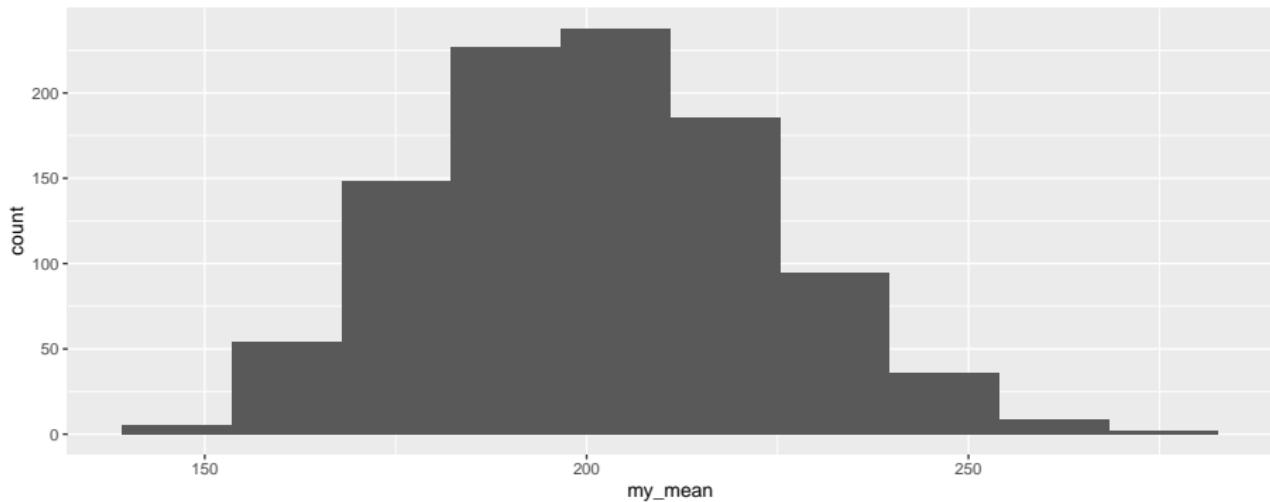
## Get the mean of each of those

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(boot_sample = list(sample(irs$Time, replace = TRUE))) %>%
  mutate(my_mean = mean(boot_sample)) -> samples
samples
```

```
# A tibble: 1,000 x 3
# Rowwise:
#   sim boot_sample my_mean
#   <int> <list>      <dbl>
# 1     1 <dbl [30]>    196
# 2     2 <dbl [30]>    202.
# 3     3 <dbl [30]>    263.
# 4     4 <dbl [30]>    173.
# 5     5 <dbl [30]>    204.
# 6     6 <dbl [30]>    197.
# 7     7 <dbl [30]>    210.
# 8     8 <dbl [30]>    160.
# 9     9 <dbl [30]>    198.
# 10   10 <dbl [30]>    178.
```

# Sampling distribution of sample mean

```
ggplot(samples, aes(x=my_mean)) + geom_histogram(bins=10)
```

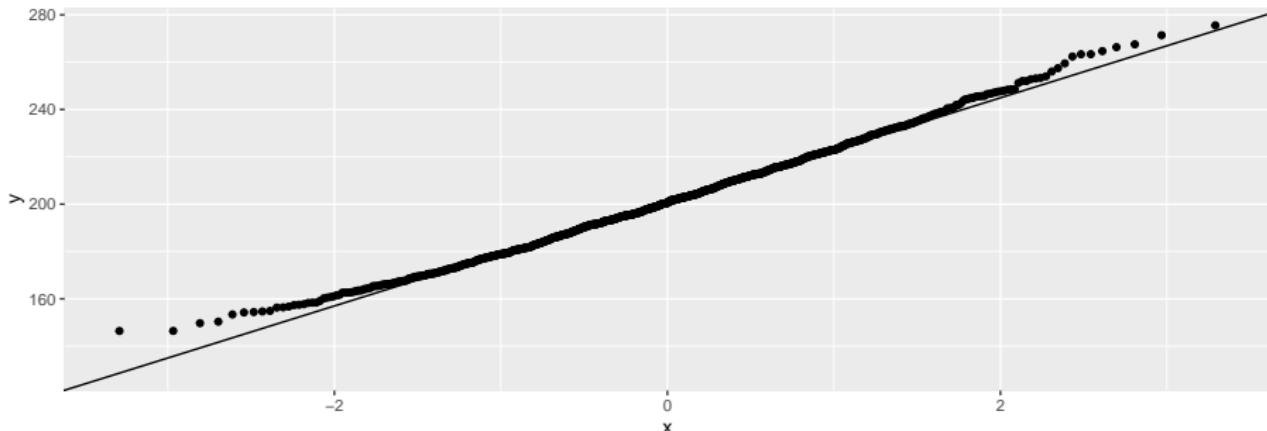


- Is that a slightly long right tail?

## Normal quantile plot

might be better than a histogram:

```
ggplot(samples, aes(sample = my_mean)) +  
  stat_qq() + stat_qq_line()
```



- a very very slight right-skewness, but very close to normal.

## Confidence interval from the bootstrap distribution

There are two ways (at least):

- percentile bootstrap interval: take the 2.5 and 97.5 percentiles (to get the middle 95%). This is easy, but not always the best:

```
(b_p=quantile(samples$my_mean, c(0.025, 0.975)))
```

```
2.5%    97.5%
162.5775 246.9092
```

- bootstrap  $t$ : use the SD of the bootstrapped sampling distribution as the SE of the estimator of the mean and make a  $t$  interval:

```
n <- length(irs$Time)
t_star <- qt(0.975, n-1)
b_t <- with(samples, mean(my_mean)+c(-1, 1)*t_star*sd(my_mean))
b_t
```

```
[1] 156.5070 246.4032
```

# Comparing

- get ordinary  $t$  interval:

```
my_names=c("LCL", "UCL")
o_t <- t.test(irs$Time)$conf.int
```

- Compare the 2 bootstrap intervals with the ordinary  $t$ -interval:

```
tibble(limit=my_names, o_t, b_t, b_p)
```

```
# A tibble: 2 x 4
  limit    o_t    b_t    b_p
  <chr> <dbl> <dbl> <dbl>
1 LCL     155.   157.   163.
2 UCL     247.   246.   247.
```

- The bootstrap  $t$  and the ordinary  $t$  are very close
- The percentile bootstrap interval is noticeably shorter (common) and higher (skewness).

## Which to prefer?

- If the intervals agree, then they are all good.
- If they disagree, they are all bad!
- In that case, use BCA interval (over).

## Bias correction and acceleration

- this from “An introduction to the bootstrap”, by Brad Efron and Robert J. Tibshirani.
- there is way of correcting the CI for skewness in the bootstrap distribution, called the BC<sub>a</sub> method
- complicated (see the Efron and Tibshirani book), but implemented in bootstrap package.

Run this on the IRS data:

```
bca=bcanon(irs$Time, 1000, mean)  
bca$confpoints
```

	alpha	bca	point
[1, ]	0.025	161.8333	
[2, ]	0.050	168.0667	
[3, ]	0.100	174.8333	
[4, ]	0.160	180.7667	
[5, ]	0.840	224.1333	
[6, ]	0.900	232.3000	
[7, ]	0.950	241.9333	
[8, ]	0.975	253.7333	

use 2.5% and 97.5% points for CI

```
bca$confpoints %>% as_tibble() %>%
  filter(alpha %in% c(0.025, 0.975)) %>%
  pull(`bca point`) -> b_bca
b_bca
```

```
[1] 161.8333 253.7333
```

# Comparing

```
tibble(limit=my_names, o_t, b_t, b_p, b_bca)
```

```
# A tibble: 2 x 5
  limit    o_t    b_t    b_p b_bca
  <chr> <dbl> <dbl> <dbl> <dbl>
1 LCL     155.   157.   163.   162.
2 UCL     247.   246.   247.   254.
```

- The BCA interval says that the mean should be estimated even higher than the bootstrap percentile interval does.
- The BCA interval is the one to trust.

## Bootstrapping the correlation

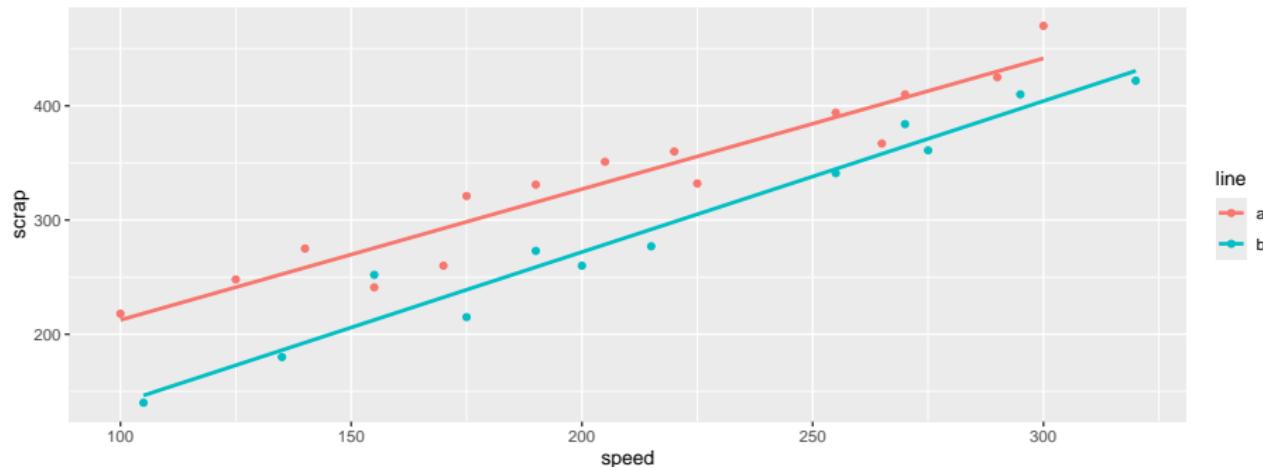
Recall the soap data:

```
url <- "http://ritsokiguess.site/datafiles/soap.txt"
soap <- read_delim(url, " ")
soap
```

```
# A tibble: 27 x 4
  case scrap speed line
  <dbl> <dbl> <dbl> <chr>
1     1    218    100 a
2     2    248    125 a
3     3    360    220 a
4     4    351    205 a
5     5    470    300 a
6     6    394    255 a
7     7    332    225 a
8     8    321    175 a
9     9    410    270 a
10   10    260    170 a
# i 17 more rows
```

# Scatterplot

```
ggplot(soap, aes(x=speed, y=scrap, colour=line))+  
  geom_point() + geom_smooth(method="lm", se=F)
```



## Comments

- Line B produces less scrap for any given speed.
- For line B, estimate the correlation between speed and scrap (with a confidence interval.)

## Extract the line B data; standard correlation test

```
soap %>% filter(line=="b") -> line_b  
with(line_b, cor.test(speed, scrap))
```

Pearson's product-moment correlation

```
data: speed and scrap  
t = 15.829, df = 10, p-value = 2.083e-08  
alternative hypothesis: true correlation is not equal to 0  
95 percent confidence interval:  
 0.9302445 0.9947166  
sample estimates:  
 cor  
 0.9806224
```

## Bootstrapping a correlation 1/2

- This illustrates a different technique: we need to keep the  $x$  and  $y$  values *together*.
- Sample *rows* of the data frame rather than individual values of speed and scrap:

```
line_b %>% sample_frac(replace=T)
```

```
# A tibble: 12 x 4
  case scrap speed line
  <dbl> <dbl> <dbl> <chr>
1     24    252   155 b
2     22    260   200 b
3     16    140   105 b
4     25    422   320 b
5     16    140   105 b
6     19    341   255 b
7     19    341   255 b
8     19    341   255 b
9     17    277   215 b
10    16    140   105 b
11    20    215   175 b
12    18    384   270 b
```

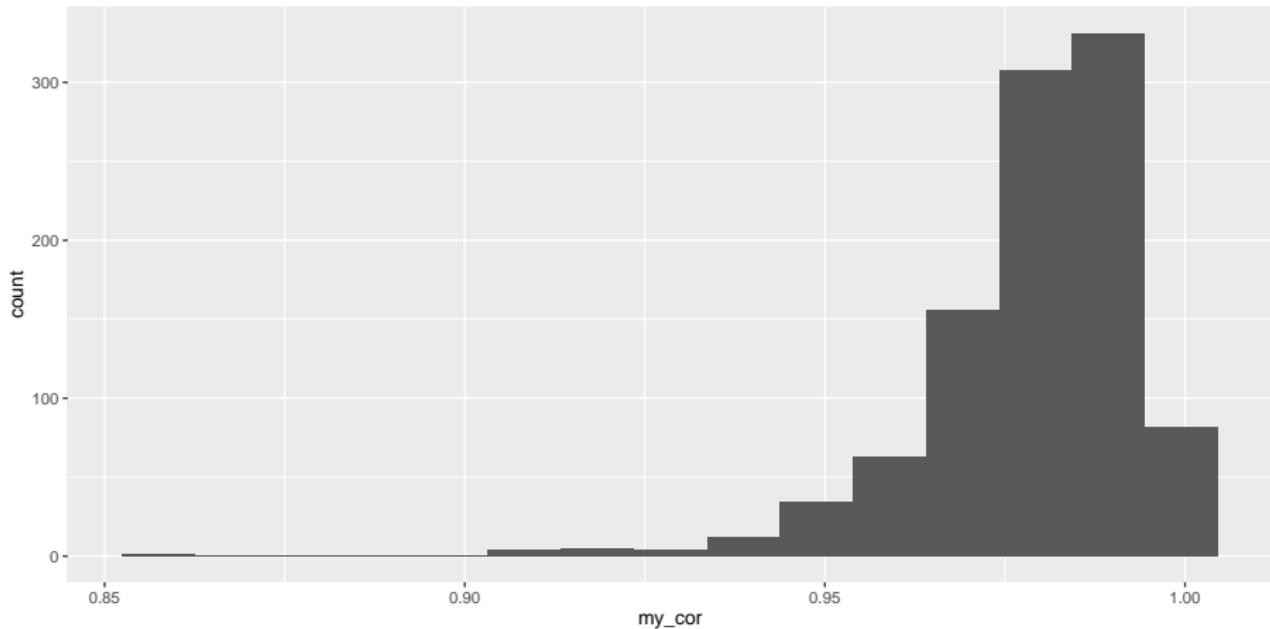
## Bootstrapping a correlation 2/2

1000 times:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(boot_df = list(sample_frac(line_b, replace = TRUE)))
  mutate(my_cor = with(boot_df, cor(speed, scrap))) -> cors
```

## A picture of this

```
ggplot(cors, aes(x=my_cor))+geom_histogram(bins=15)
```



## Comments and next steps

- This is very left-skewed.
- Bootstrap percentile interval is:

```
(b_p=quantile(cors$my_cor, c(0.025, 0.975)))
```

2.5% 97.5%  
0.9415748 0.9962462

- We probably need the BCA interval instead.

## Getting the BCA interval 1/2

- To use bcanon, write a function that takes a vector of row numbers and returns the correlation between speed and scrap for those rows:

```
theta=function(rows, d) {  
  d %>% slice(rows) %>% with(., cor(speed, scrap))  
}  
theta(1:3, line_b)
```

```
[1] 0.9928971
```

```
line_b %>% slice(1:3)
```

```
# A tibble: 3 x 4  
  case scrap speed line  
  <dbl> <dbl> <dbl> <chr>  
1     16    140    105 b  
2     17    277    215 b  
3     18    384    270 b
```

- That looks about right.

## Getting the BCA interval 2/2

- Inputs to bcanon are now:

- ▶ row numbers (1 through 12 in our case: 12 rows in line\_b)
- ▶ number of bootstrap samples
- ▶ the function we just wrote
- ▶ the data frame:

```
points=bcanon(1:12, 1000, theta, line_b)$confpoints  
points %>% as_tibble() %>%  
  filter(alpha %in% c(0.025, 0.975)) %>%  
  pull(`bca point`) -> b_bca  
b_bca
```

```
[1] 0.9314334 0.9947799
```

## Comparing the results

```
tibble(limit=my_names, o_c, b_p, b_bca)
```

```
# A tibble: 2 x 4
  limit    o_c    b_p   b_bca
  <chr>  <dbl>  <dbl>  <dbl>
1 LCL     0.930  0.942  0.931
2 UCL     0.995  0.996  0.995
```

- The bootstrap percentile interval doesn't go down far enough.
- The BCA interval seems to do a better job in capturing the skewness of the distribution.
- The ordinary confidence interval for the correlation is very similar to the BCA one, and thus seems to be trustworthy here even though the correlation has a very skewed distribution. (`cor.test` uses the Fisher  $z$  transformation which "spreads out" correlations close to 1).

# The $z$ -transformed bootstrapped correlations

```
cors %>%
  mutate(z = 0.5 * log((1+my_cor)/(1-my_cor))) %>%
  ggplot(aes(sample=z)) + stat_qq() + stat_qq_line()
```

