Bootstrap again

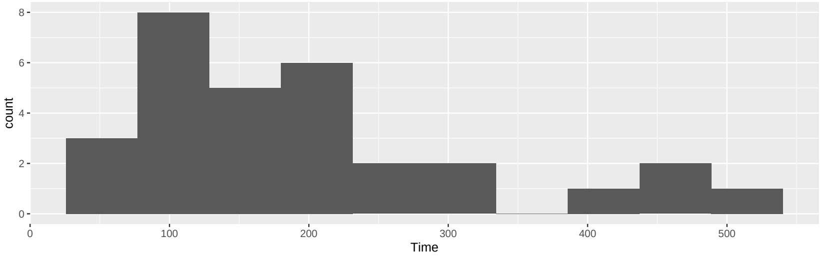
packages

```r
library(tidyverse)
library(bootstrap)
library(rsample)
library(conflicted)
conflict_prefer("filter", "dplyr")
```

Is my sampling distribution normal enough?

- Recall the IRS data that we used as a motivation for the sign test:

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



- We said that a 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)
```

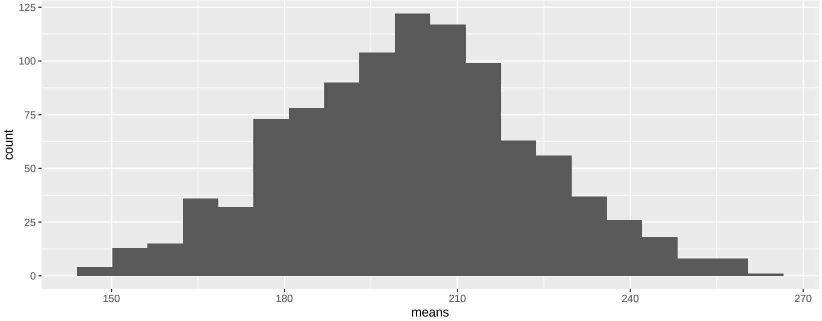
```
[1] 182.5667
```

- That's one bootstrapped mean. We need a whole bunch.
- Use the same idea as for simulating power:

```
rerun(1000, sample(irs$Time, replace=T)) %>%
  map_dbl(~mean(.)) -> means
```

Sampling distribution of sample mean

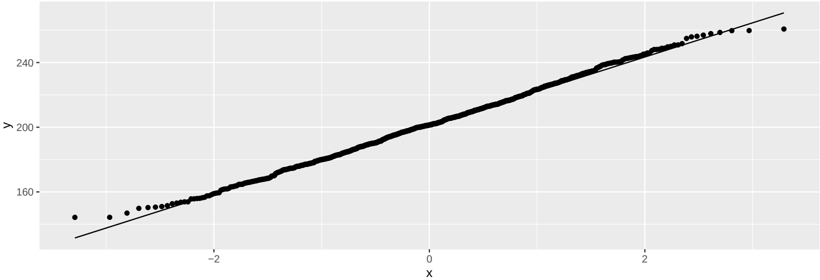
```
ggplot(tibble(means), aes(x=means))+geom_histogram(bins=20)
```



Comments

This is not so bad: a long right tail, maybe:

```
ggplot(tibble(means), aes(sample=means))+
  stat_qq()+stat_qq_line()
```



or not so much.

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(means, c(0.025, 0.975)))
```

```
    2.5%    97.5%
159.5292 244.1075
```

- 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=mean(means)+c(-1, 1)*t_star*sd(means))
```

```
[1] 157.9020 245.4334
```

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.  158.  160.
2 UCL    247.  245.  244.
```

- 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 BCa 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  162.8000
[2,] 0.050  166.8000
[3,] 0.100  174.4667
[4,] 0.160  179.5667
[5,] 0.840  224.6000
[6,] 0.900  234.3000
[7,] 0.950  247.4333
[8,] 0.975  256.3000
```

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] 162.8 256.3
```

Comparing

```r
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.  158.  160.  163.
2 UCL    247.  245.  244.  256.
```

- 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.

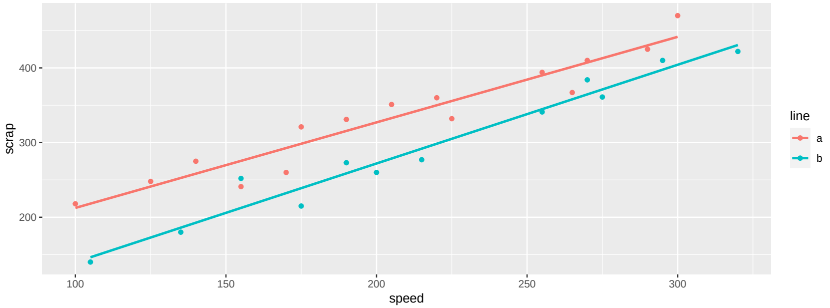
Bootstrapping the correlation

Recall the soap data:

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

The data

```
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

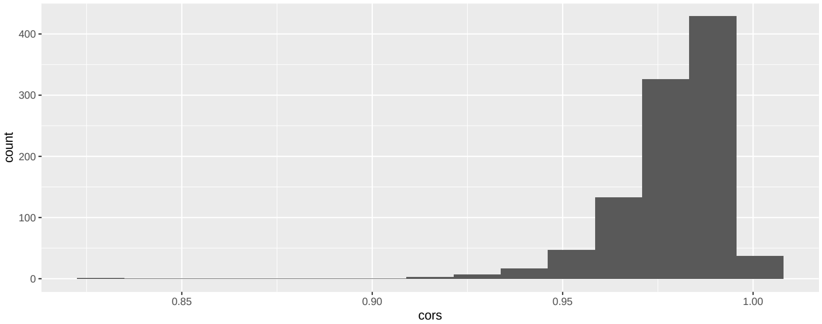
- Sample from data with replacement, but have to keep the `speed-scrap` *pairs* together
- Sample *rows* at random, then take the variable values that belong to those rows:

*** REDO THIS ***

```
rerun(1000, sample(1:nrow(line_b), replace=T)) %>%
  map(~slice(line_b, .)) %>%
  map_dbl(~with(.,cor(speed, scrap))) -> cors
```

A picture of this

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



Comments and next steps

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

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

```
     2.5%     97.5%
0.9443861 0.9960207
```

- 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
```

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] 0.9281762 0.9945016
```

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 0.944 0.928
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 than the ordinary `cor.test` interval in capturing the skewness of the distribution.

A problem

Consider this example: samples of UK and Ontario (Canada) children, and their journey times to school, in minutes:

```
my_url="http://ritsokiguess.site/datafiles/to-school.csv"
to_school=read_csv(my_url)
to_school
```

```
# A tibble: 80 x 2
   traveltime location
        <dbl> <chr>
 1         30 Ontario
 2         10 Ontario
 3          8 Ontario
 4         30 Ontario
 5          5 Ontario
 6          8 Ontario
 7          7 Ontario
 8         15 Ontario
 9         10 Ontario
```

Automating the bootstrap

Let's go back to our IRS data for a moment:

```
irs
```

```
# A tibble: 30 x 1
    Time
   <dbl>
 1    91
 2    64
 3   243
 4   167
 5   123
 6    65
 7    71
 8   204
 9   110
10   178
# i 20 more rows
```

What happens if we use `rsample` to resample from these? Let's