Bootstrapping and Permutation Tests

Week 17

Winnie Xia

Get Started

What is the difference between parametric or non-parametric?

Bootstrap

- Definition: It usually refers to a self-starting process that is to proceed without external input.
- Applied to statistics: We sample with replace from the sample

Bootstrap

Bootstrap is a desirable approach when:

- the distribution of a statistic is unknown or complicated.
- Reason: bootstrap is a non-parametric and does not ask for specific distributions.
- the sample size is too small to draw a valid inference.
- Reason: it is a resampling method with replacement and recreates any number of resamples.

Let's break down "bootstrap"

Bootstrap breaks down into the following steps:

- decide how many bootstrap samples to perform.
- what is the sample size?
- for each bootstrap sample:
  - draw a sample with replacement with the chosen size
  - calculate the statistic of interest for that sample
- calculate the mean of the calculated sample statistics.

Bootstrapping Illustration in R

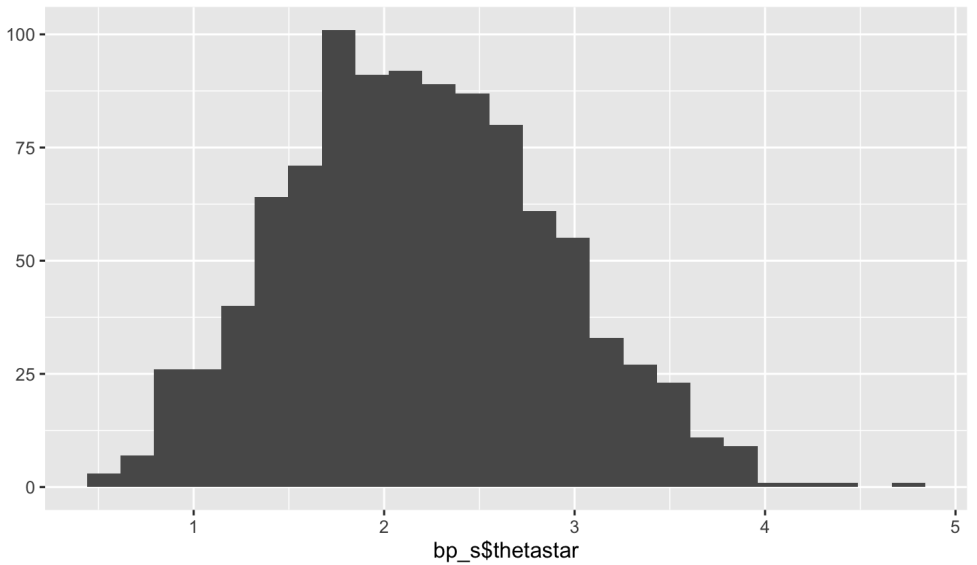
```
set.seed(2021)
n <- 20
s <- rnorm(n = n,
           mean = 5.5,
           sd = 1.4)
bp_s <- bootstrap::bootstrap(s, 1000, var)
str(bp_s)
```

```
## List of 5
##  $ thetastar     : num [1:1000] 1.77 3 1.52 1.23 2.47 ...
##  $ func.thetastar: NULL
##  $ jack.boot.val : NULL
##  $ jack.boot.se  : NULL
##  $ call          : language bootstrap::bootstrap(x = s, nboot = 1000, theta
```

Bootstrapping Illustration in R

The list item thetastar (a vector) contains each of the bootrap estimates of the statistic of interest (variance in the present example). It's always good to plot a histogram of the bootstrap distribution.

```
quickplot(bp_s$thetastar,
          geom = "histogram",
          bins = 25)
```

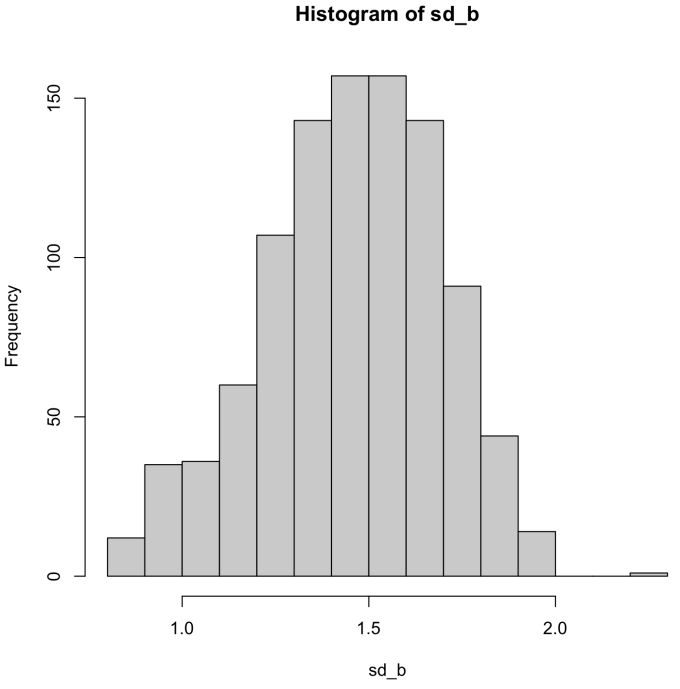


Or we could write the bootstrap() manually

```
B <- matrix(0, nrow = 1000,
            ncol = n)

for (i in 1: 1000){
  B[i, ] <- sample(s, size = n,
                   replace = TRUE
}

sd_b <- apply(B, 1, sd) # compute
hist(sd_b) # inspect the distribu
```



Other Sampling Approaches

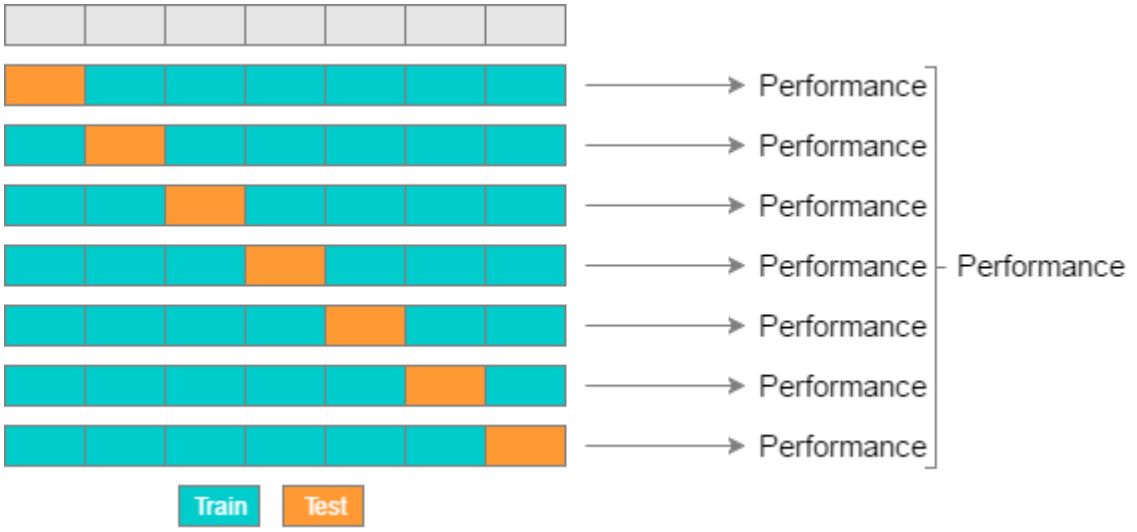
Jackknife

It is a leave-one-out procedure. It means:

- We copy the existing sample *n* times, and each time, we delete one but different observation.
- Then, we calculate the statistics of interests.

K-cross Validation

- Divide the data into k parts and predict one left out segment based on a model of the remaining k − 1 segments;
- Then assess distribution of prediction error.



Permutation Tests

- To compare outcomes in experiments, we often do a two-sample t-test.
- It assumes that data are randomly selected from the population, arrived in large samples (>30), or normally distributed with equal variances between groups.
- But we could also o a permutation test, without any distributional assumptions.

Permutation Tests in R

```
set.seed(2021)
dat <- data.frame(group = c(rep("t", 10),
                            rep("c", 10)),
                  mark = c(rnorm(10, 69, 10),
                           rnorm(10, 57, 10)))
head(dat)
```

```
##   group     mark
## 1     t 67.77540
## 2     t 74.52457
## 3     t 72.48650
## 4     t 72.59632
## 5     t 77.98054
## 6     t 49.77430
```

Permutation Test in R

- First, we compute the difference in means.

```
obs <- mean(dat$mark[dat$group == "t"]) -
  mean(dat$mark[dat$group == "c"])
obs
```

```
## [1] 10.18717
```

- Then, we perform the *t* test.

```
t.test(dat$mark[dat$group == "t"], dat$mark[dat$group == "c"])
```

```
##
##     Welch Two Sample t-test
##
## data:  dat$mark[dat$group == "t"] and dat$mark[dat$group == "c"]
## t = 2.0439, df = 16.694, p-value = 0.05708
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3434257 20.7177593
## sample estimates:
## mean of x mean of y
##  72.03481  61.84764
```

Permutation Test in R

```
d <- numeric(1000)
for (i in 1:1000){
  dat2 <- dat
  dat$group <- sample(dat2$group,
                      replace = F
  d[i] <- mean(dat$mark[dat$group
  mean(dat$mark[dat$group == "c"]
}
# Here, we re-assign groups label
# and then re-compute the differe
# means again;
# we repeat these steps.
# Eventually, this yields a distr

hist(d)
abline(v = quantile(d, 0.95),
       col = "blue",
       lwd = 3)
abline(v = obs,
       col = "red",
       lwd = 3)
```