Assignment 5

Sampling Intervals for Models

Roman Grebnev

2022-11-21

Contents

```
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.1.3
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

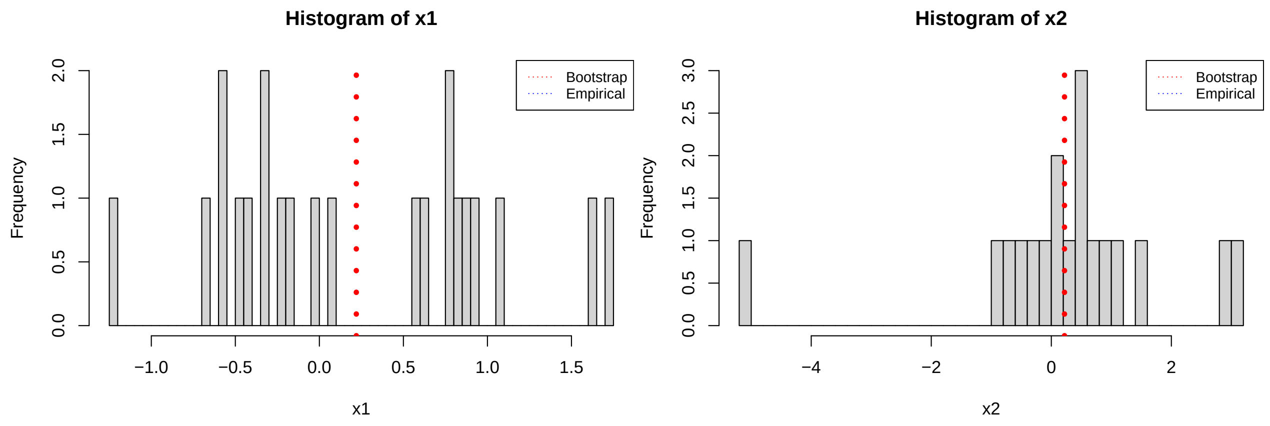
Task 1

```
x1 <- c(-0.673, -0.584, 0.572, -0.341, -0.218, 0.603, -0.415, -0.013, 0.763, 0.804, 0.054, 1.746, -0.47

x2 <- c(0.913, -0.639, 2.99, -5.004, 3.118, 0.1, 1.128, 0.579, 0.32, -0.488, -0.994, -0.212, 0.413, 1.4
```

Task 1.1

Plot the data in a way which visualizes the comparison of means appropriately.

```
hist(x1, breaks = 50)
abline(v = mean(x1), col = 'red', lty = 3, lwd = 5)
legend(x = "topright", legend=c("Bootstrap", "Empirical"),
       col=c("red", "blue"), lty=3, cex=0.8)

hist(x2, breaks = 50)
abline(v = mean(x1), col = 'red', lty = 3, lwd = 5)
legend(x = "topright", legend=c("Bootstrap", "Empirical"),
       col=c("red", "blue"), lty=3, cex=0.8)
```



```
mean(x1)
```

```
## [1] 0.2198182
```

```
sd(x1)
```

```
## [1] 0.7980614
```

```
mean(x2)
```

```
## [1] 0.2717222
```

```
sd(x2)
```

```
## [1] 1.707192
```

Both samples have similar mean, but different variance

Task 1.2

Consider different sampling schemes, such as:

- Sampling with replacement from each group.
- Centering both samples and then resample from the combined samples X1 and X2 for n1 and n2 times. Argue for choice what is more natural and which advantages or disadvantages may apply.

Sampling with replacement:

Generate bootstrap samples and compute t-statistics based on each pair of bootstrap samples.

```
two_sample_t_test_boot <- function(x1_boot, x2_boot) {

mean_x1_boot <- mean(x1_boot)
mean_x2_boot <- mean(x2_boot)

std_x1_boot <- sd(x1_boot)
```

```r
std_x2_boot <- sd(x2_boot)

n1 <- length(x1_boot)
n2 <- length(x2_boot)

t_stat <- (mean_x1_boot - mean_x2_boot)/(sqrt(std_x1_boot^2/n1 + std_x2_boot^2/n2))
}
```

Compute confidence intervals

```r
conf_boundaries <- function(x1_boot, x2_boot, alpha) {

  mean_x1_boot <- mean(x1_boot)
  mean_x2_boot <- mean(x2_boot)

  std_x1_boot <- sd(x1_boot)
  std_x2_boot <- sd(x2_boot)

  n1 <- length(x1_boot)
  n2 <- length(x2_boot)

  df_two_sample = min(c(n1-1,n2-1))

  upper_ci <- (mean_x1_boot - mean_x2_boot) +  qt(p=alpha/2, df=df_two_sample, lower.tail=FALSE) * sqrt
  lower_ci <- (mean_x1_boot - mean_x2_boot) - qt(p=alpha/2, df=df_two_sample, lower.tail=FALSE) * sqrt(

  c(lower_ci, upper_ci)
}
```

Compute p-value as the proportion of samples, which exceed the critical value of t-statistic

```r
compute_p_val <- function(x1, x2, t_stat_boot, alpha) {

  n1 <- length(x1)
  n2 <- length(x2)
  df_two_sample <- min(n1-1, n2-1)
  ttest_crit_val <- (mean(x1) - mean(x2))/(sqrt(sd(x1)^2/n1 + sd(x2)^2/n2))
  m <- length(t_stat_boot)
  cnt_crit <- 0
  for (i in 1:length(t_stat_boot)) {
    if (abs(t_stat_boot[i]) >= abs(ttest_crit_val)) {
      cnt_crit <- cnt_crit + 1
    }
  }
  p_val <- (cnt_crit + 1)/(m +1)
  p_val
}
```

```r
confidence_intervals <- function(x1, x2, alpha, m){
  ci_bounds <- replicate(m, conf_boundaries(sample(x1, replace = TRUE), sample(x2, replace = TRUE), alp
  ci_df <- data.frame(cbind(ci_bounds[1,], ci_bounds[2,]))
  colnames(ci_df) <- c('ci_lower', 'ci_upper')
  c(mean(ci_df$ci_lower), mean(ci_df$ci_upper))
}
```

Sampling from centered combined samples x1 and x2 for n1 and n2 times.

For bootstrap sampling from centered data, approach is slightly different. The main difference compared to the previous approach is that we need to center data before applying bootstrap sampling.

Task 1.3

Bootstrap using both strategies mentioned above using the t-test statistic. Calculate the bootstrap p-value based on 10000 bootstrap samples and 0.95 as well as 0.99 confidence intervals. Make your decision at the significance level 0.05 or 0.01, respectively.

Sampling with replacement of t-statistics.

Compute t-statistics based on bootstrap samples:

```
t_test_vals <- replicate(10000, two_sample_t_test_boot(
  sample(x1, replace = TRUE),
  sample(x2, replace = TRUE)
))
```

```
p_val <- compute_p_val(x1, x2, t_test_vals, alpha = 0.01)
```

```
p_val <- compute_p_val(x1, x2, t_test_vals, alpha = 0.05)
```

Decision about mean equivalence:

For both significance levels, the inference about the equivalence of the sample means of samples x1 and x2 is the same. As p-value is 0.9058, there is no sufficient evidence to reject H0.

Evaluate confidence intervals based on bootstrap samples:

For significance level 0.05:

```
confidence_intervals(x1, x2, 0.05, 10000)
```

```
## [1] -0.9265560  0.8160128
```

For significance level 0.01:

```
confidence_intervals(x1, x2, 0.01, 10000)
```

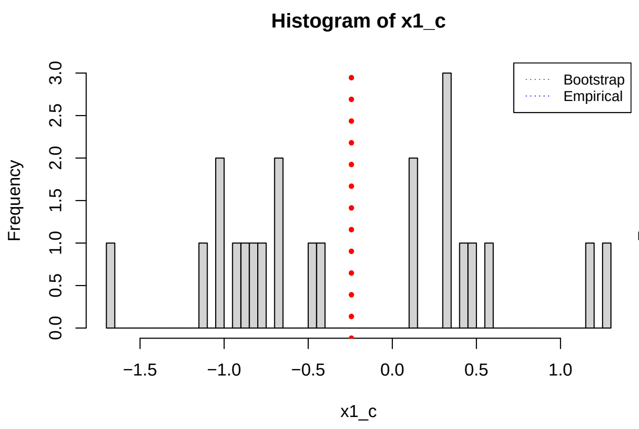
```
## [1] -1.246795  1.139923
```

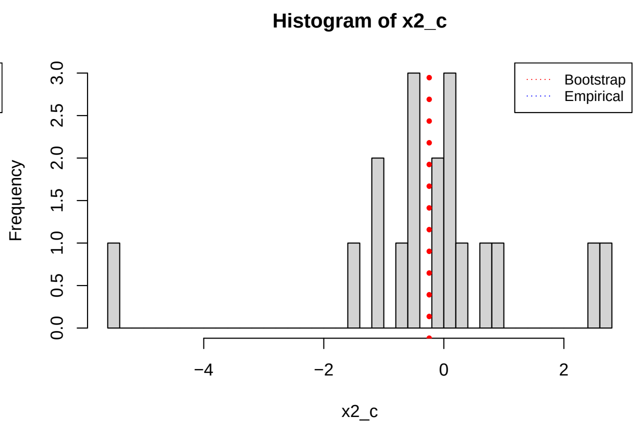
Sampling from centered combined samples x1 and x2 for n1 and n2 times:

As a first step we center the samples.

```
x_pooled <- c(x1, x2)
x1_c <- x1 - mean(x1) - mean(x_pooled)
x2_c <- x2 - mean(x2) - mean(x_pooled)
```

```
hist(x1_c, breaks = 50)
abline(v = mean(x1_c), col = 'red', lty = 3, lwd = 5)
legend(x = "topright", legend=c("Bootstrap", "Empirical"),
       col=c("red", "blue"), lty=3, cex=0.8)

hist(x2_c, breaks = 50)
abline(v = mean(x2_c), col = 'red', lty = 3, lwd = 5)
legend(x = "topright", legend=c("Bootstrap", "Empirical"),
       col=c("red", "blue"), lty=3, cex=0.8)
```





```r
t_test_vals_centered <- replicate(10000, two_sample_t_test_boot(
  sample(x1_c, replace = TRUE),
  sample(x2_c, replace = TRUE)
))
```

```r
confidence_intervals(x1_c, x2_c, 0.05, 10000)
```

```
## [1] -0.8727175  0.8700903
```

```r
confidence_intervals(x1_c, x2_c, 0.01, 10000)
```

```
## [1] -1.196529  1.196119
```

```r
compute_p_val(x1, x2, t_test_vals_centered, alpha = 0.05)
```

```
## [1] 0.9112089
```

```r
compute_p_val(x1, x2, t_test_vals_centered, alpha = 0.01)
```

```
## [1] 0.9112089
```

As in the t-test above, there is no sufficient evidence to reject H0 as p-value is 0.908. Beside that, confidence intervals of the second approach compared to the first one have the same width for significance levels 0.05 and 0.01 correspondignly. CIs of the second approach are shifted because of the centering of the data.

Task 1.4

What would be a permutation version of the test? Implement the corresponding permutation test and obtain p-value and confidence intervals as in 3. to get a corresponding test decision at the same significance levels.

Permutation t-test have the following sampling approach: Values are sampled from the pooled sample. Then first n1 elements of the pooled bootstrap sample are attributed to the first sample, and n2 remaining samples are attributed to the second sample. Then statistics are computed based on obtained bootstrap samples.

Implementatiom of the permutation test:

```r
two_sample_t_test_boot_permutation <- function(x1, x2, indices, alpha) {

  n1 <- length(x1)
  n2 <- length(x2)

  x_pooled <- c(x1, x2)
  x1_perm <- x_pooled[indices][1:n1]
  x2_perm <- x_pooled[indices][-c(1:n2)]

  mean_x1_perm <- mean(x1_perm)
  mean_x2_perm <- mean(x2_perm)

  std_x1_perm <- sd(x1_perm)
  std_x2_perm <- sd(x2_perm)

  df_two_sample <- n1+n2-2

  t_stat_perm <- (mean_x1_perm - mean_x2_perm)/(sqrt(std_x1_perm^2/n1 + std_x2_perm^2/n2))
  ci_lower <- (mean_x1_perm - mean_x2_perm) - qt(p=alpha/2, df=df_two_sample, lower.tail=FALSE) * sqrt(
  ci_upper <- (mean_x1_perm - mean_x2_perm) + qt(p=alpha/2, df=df_two_sample, lower.tail=FALSE) * sqrt(
  c(t_stat_perm, ci_lower, ci_upper)
}
```

Obtained confidence intervals for significance level of 0.05:

```
t_test_vals_permutation <- replicate(10000, two_sample_t_test_boot_permutation(
  x1,
  x2,
  sample(c(1:length(c(x1, x2))), replace = TRUE),
  alpha = 0.05
))

perm_bootstrap_95 <- data.frame(cbind(t_test_vals_permutation[1,], t_test_vals_permutation[2,], t_test_
colnames(perm_bootstrap_95) <- c('tstatistic', 'ci_lower', 'ci_upper')
mean(perm_bootstrap_95$ci_lower)
```

```
## [1] -0.7922424
```

```
mean(perm_bootstrap_95$ci_upper)
```

```
## [1] 0.7753856
```

Obtained confidence intervals for significance level of 0.01:

```
t_test_vals_permutation <- replicate(10000, two_sample_t_test_boot_permutation(
  x1,
  x2,
  sample(c(1:length(c(x1, x2))), replace = TRUE),
  alpha = 0.01
))

perm_bootstrap_99 <- data.frame(cbind(t_test_vals_permutation[1,], t_test_vals_permutation[2,], t_test_
colnames(perm_bootstrap_99) <- c('tstatistic', 'ci_lower', 'ci_upper')
mean(perm_bootstrap_99$ci_lower)
```

```
## [1] -1.052544
```

```
mean(perm_bootstrap_99$ci_upper)
```

```
## [1] 1.0562
```

Obtained p-values:

```
compute_p_val(x1, x2, perm_bootstrap_95$tstatistic, alpha = 0.05)
```

```
## [1] 0.8962104
```

```
compute_p_val(x1, x2, perm_bootstrap_99$tstatistic, alpha = 0.01)
```

```
## [1] 0.8967103
```

Based on the obtained p-values for permutation test we can conclude that there is not enough evidence to reject H0.

Task 1.5

The Wilcoxon rank sum test statistic is the sum of ranks of the observations of sample 1 computed in the combined sample. Use bootstrapping with both strategies mentioned above and obtain p-value and confidence intervals as in 3. to get a corresponding test decision at the same significance levels.

Implementation of bootstrap version of Wilcoxon rank sum test:

```
wilcoxon_statistic <- replicate(10000,
  sample(wilcox.test(x1, x2, alternative = "two.sided")$statistic, replace = TRUE)
)[1,]
```

Computation of the p-value based on the critical value of Wilcoxon distribution with significance level alpha.

```
compute_p_val_wilcoxon_boot <- function(x1, x2, w_stat, alpha) {


  w_stat_crit_val <- qsignrank(alpha/2, length(x1))
  m <- length(w_stat)

  cnt_crit <- 0

  for (i in 1:length(w_stat)) {
    if (abs(w_stat[i]) >= abs(w_stat_crit_val)) {
      cnt_crit <- cnt_crit + 1
    }
  }
  p_val <- (cnt_crit + 1)/(m +1)
  p_val
}
```

Obtain p-value for 0.05 significance level:

```
compute_p_val_wilcoxon_boot(x1, x2, wilcoxon_statistic, 0.05)
```

```
## [1] 0.6349365
```

Compute bootstrapped Wilcoxon statistic based on the centered data.

```
x_pooled <- c(x1, x2)
x1_c_w <- x1 - mean(x1) - mean(x_pooled)
x2_c_w <- x2 - mean(x2) - mean(x_pooled)

wilcoxon_statistic_centered <- replicate(10000,
  sample(wilcox.test(x1_c_w, x2_c_w)$statistic, replace = TRUE)
)[1,]

compute_p_val_wilcoxon_boot(x1, x2, wilcoxon_statistic_centered, 0.05)
```

```
## [1] 0.6471353
```

```
quantile(wilcoxon_statistic, c(0.025, 1-0.025))
```

```
##  2.5% 97.5%
##     5   177
```

```
quantile(wilcoxon_statistic, c(0.005, 1-0.005))
```

```
##  0.5% 99.5%
##     1   181
```

```
quantile(wilcoxon_statistic_centered, c(0.025, 1-0.025))
```

```
##  2.5% 97.5%
##     5   183
```

```
quantile(wilcoxon_statistic_centered, c(0.005, 1-0.005))
```

```
##  0.5% 99.5%
##     1   187
```

Conclusion:

For both both approaches Wilcoxon test provides similar results and brings to the same conclusions about the hypothesis: we don't have enough evidence to reject H0.

Task 1.6

Compare your results to the results using t.test and wilcox.test.

```
t.test(x1, x2, alternative = "two.sided", var.equal = FALSE, conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  x1 and x2
## t = -0.11881, df = 23.027, p-value = 0.9065
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9556081  0.8518000
## sample estimates:
## mean of x mean of y
## 0.2198182 0.2717222
```

```
t.test(x1, x2, alternative = "two.sided", var.equal = FALSE, conf.level = 0.99)
```

```
##
##  Welch Two Sample t-test
##
## data:  x1 and x2
## t = -0.11881, df = 23.027, p-value = 0.9065
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -1.278256  1.174448
## sample estimates:
## mean of x mean of y
## 0.2198182 0.2717222
```

```
t.test(x1_c, x2_c, alternative = "two.sided", var.equal = FALSE, conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  x1_c and x2_c
## t = 0, df = 23.027, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.9037041  0.9037041
## sample estimates:
## mean of x mean of y
## -0.243175 -0.243175
```

```r
t.test(x1_c, x2_c, alternative = "two.sided", var.equal = FALSE, conf.level = 0.99)
```

```
##
##  Welch Two Sample t-test
##
## data:  x1_c and x2_c
## t = 0, df = 23.027, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -1.226352  1.226352
## sample estimates:
## mean of x mean of y
## -0.243175 -0.243175
```

```r
wilcox.test(x1, x2, alternative = "two.sided", conf.level = 0.95)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  x1 and x2
## W = 181, p-value = 0.6572
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(x1, x2, alternative = "two.sided", conf.level = 0.99)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  x1 and x2
## W = 181, p-value = 0.6572
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(x1_c, x2_c, alternative = "two.sided", conf.level = 0.95)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  x1_c and x2_c
## W = 188, p-value = 0.7984
## alternative hypothesis: true location shift is not equal to 0
```

```r
wilcox.test(x1_c, x2_c, alternative = "two.sided", conf.level = 0.99)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  x1_c and x2_c
## W = 188, p-value = 0.7984
## alternative hypothesis: true location shift is not equal to 0
```

We draw the same conclusion for 5 variations of the test: there is not enough evidence to reject H0.

- T-test based on the bootstrapped samples
- T-test based on the centered samples
- Permutation T-test based on the bootstrapped samples
- Permutation T-test based on the centered samples
- Wilcoxon test based on the bootstrapped samples
- Wilcoxon test based on the centered samples

Task 2

Consider the model $y=3+2X1+X2+\epsilon$ where X1 comes from a normal distribution with mean 2 and variance 3, X2 comes from a uniform distribution between 2 and 4 and ϵ from a student's t distribution with 5 degrees of freedom. In addition, there is a predictor X3 coming from a uniform distribution between -2 and 2.

Task 2.1

Create a sample of size 200 from the model above and for the independent predictor X3 .

```r
x1_var <- rnorm(200, 2, 3)
x2_var <- runif(200, 2, 4)
x3_var <- runif(200, -2, 2)
epsilon <- rt(200, 5)

y <- 3 + 2 * x1_var + x2_var + epsilon
```

```r
data_synth <- data.frame(cbind(y, x1_var, x2_var, x3_var))
```

Task 2.2

Do residual bootstrap for linear regression and fit the model y~X1+X2+X3 . Get the percentile CI for the coefficients. Can you exclude X3?

```r
fit <- lm(y ~ x1_var+x2_var+x3_var, data = data_synth)

regression.func <- function(x){coef(lm(y ~ x1_var+x2_var+x3_var, data = x))}
regression.sim <- function(x, res){
  x$y <- (res$yhat + sample(res$res, replace = TRUE))
  return(x)
}

res <- resid(fit)
yhat <- fitted(fit)

regr_yhat_resid <- data.frame(yhat, res)
fit.boot <- boot(data_synth, regression.func, R=10000, sim="parametric", ran.gen=regression.sim, mle = 

x1_beta_ci <- quantile(fit.boot$t[,2], c(0.005,0.025,0.975, 0.995))
x2_beta_ci <- quantile(fit.boot$t[,3], c(0.005,0.025,0.975, 0.995))
x3_beta_ci <- quantile(fit.boot$t[,4],  c(0.005,0.025,0.975, 0.995))

df_beta_conf_ints_resid <- data.frame(x1 = x1_beta_ci,x2 = x2_beta_ci,x3 = x3_beta_ci)

knitr::kable(df_beta_conf_ints_resid, caption="CI for residuals bootstrap estimates of model coefficien
```

Table 1: CI for residuals bootstrap estimates of model coefficients

	x1	x2	x3
0.5%	1.940560	0.7115734	-0.3076832
2.5%	1.956752	0.8044790	-0.2631461
97.5%	2.068640	1.4016152	0.0438302
99.5%	2.085959	1.4915102	0.0886854

Yes, X3 is insignificant, thus we can remove it safely from model specification.

Task 2.3

Do pairs bootstrap for linear regression and fit the model y~X1+X2+X3. Get the percentile CI for the coefficients. Can you exclude x3 ?

```r
fit <- lm(y ~ x1_var+x2_var+x3_var, data = data_synth)
res <- resid(fit)
yhat <- fitted(fit)

regr_yhat_resid <- data.frame(yhat, res)

reg_fun <- function(x, ids){
  return(coef(lm(y ~ x1_var + x2_var + x3_var, data = x[ids, ])))
}

fit.boot_pairs <- boot(data_synth, reg_fun, R = 10000)

bot_pair_sol <- fit.boot_pairs$t

x1_beta_ci <- quantile(bot_pair_sol[,2], c(0.005,0.025,0.975, 0.995))
x2_beta_ci <- quantile(bot_pair_sol[,3], c(0.005,0.025,0.975, 0.995))
x3_beta_ci <- quantile(bot_pair_sol[,4], c(0.005,0.025,0.975, 0.995))

df_beta_conf_ints_pairs <- data.frame(
  x1 = x1_beta_ci,
  x2 = x2_beta_ci,
  x3 = x3_beta_ci
)

knitr::kable(df_beta_conf_ints_pairs, caption="CI for pairs bootstrap estimates of model coefficients")
```

Table 2: CI for pairs bootstrap estimates of model coefficients

	x1	x2	x3
0.5%	1.931407	0.7068012	-0.3212088
2.5%	1.949516	0.8048534	-0.2725225
97.5%	2.070824	1.4107235	0.0561072
99.5%	2.088294	1.5155393	0.1164842

Yes, X3 is insignificant and we can remove it from model specification.

Task 2.4

Compare the two approaches in 2. and 3. and explain the differences in the sampling approach and how this (might) affect(s) the results.

The principal difference between approaches of residuals bootstrap and pairs bootstrap is that for residuals one we make the assumption of the normal distribution of the residuals and thus this is the parametric bootstrap approach. For the pairs bootstrap we don't make such assumptions and draw samples directly from the distributions of the residuals. Conceptual difference between the approaches is that residuals bootstrap samples the residuals of the original model, while the pairs bootstrap samples the pairs of y and x from the original dataset. Both approaches are asymptotically equivalent, when the model is correctly specified. They performed differently for small samples. Pairs strategy is considered more robust, when the model is specifies correctly.

Task 3

Summarize the bootstrapping methodology, its advantages and disadvantages based on your exercises for constructing parametric and non-parametric confidence bounds for estimators, test statistics or model estimates.

Bootstrapping technique can be used in various contexts for evaluation of estimates of random variables and for hypothesis testing. Bootstrapping is performed by sampling from the empirical distribution of random variables with replacements and thus obtaining the bootstrap samples. Bootstrapping utilizes central limit theorem and asymptotic theory as its basis for estimation of the parameters as when the number of bootstrap samples is high, the distribution of examined estimates θ, evaluated on bootstrap samples, starts following the normal distribution.

Bootstrap is a convenient tool for evaluation of confidence intervals. Advantage of non-parametric bootstrap is that we don't make any assumptions about the population's distribution and hence we can construct confidence intervals very easy. Disadvantage of bootstrap approach is its computational intensity as in order to obtain bootstrap estimates of parameters we need to compute the amount of samples reaching 10000 samples.