Introduction to Simulation with Variance Estimation

Bosse Behrens, st.id 12347333

2024W

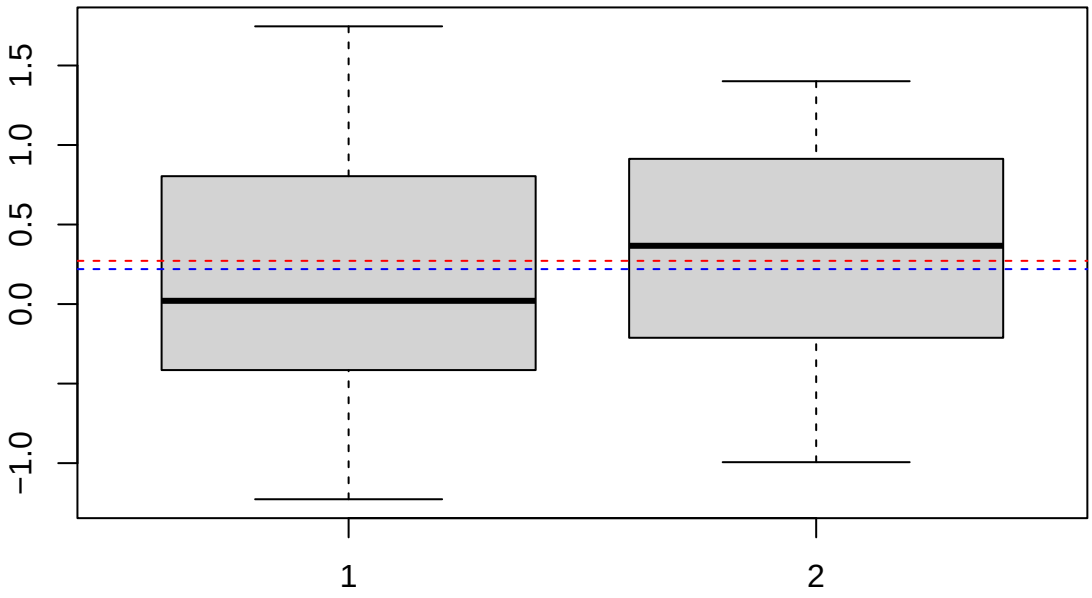
Task 1

part 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, -0.472, 1.638, -0.578, 0.947, -0.329, -0.188,
        0.794, 0.894, -1.227, 1.059)
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, 1.401, 0.007, 0.568, -0.005, 0.696)
```

We will plot both samples as boxplots. We do not show the outliers for a better overview and add the means of both x1 and x2.

```
x1_mean <- mean(x1)
x2_mean <- mean(x2)
boxplot(x1,x2, outline = FALSE)
abline(h = x1_mean, col = "blue", lty = 2)
abline(h = x2_mean, col = "red", lty = 2)
```



As we can observe both means are very close to each other.

part 2

An advantage of resampling each sample individually is that it preserves the individual structures and variability. This is good when not assuming any relationship between the two samples.
Disadvantages are that the sample sizes may be too small which can lead to mayn similar data points and also assumes independence between both samples.
For the combined centered samples advantages are a bigger sample size and also being better for assuming similar distribution or properties like the mean. This is useful when assuming the same distribution and testing on for example $\mu_1 = \mu_2$ since it removes group specific location effects.
Since we want to test for exactly that example the combined sample bootstrapping is in this case the prefered method.

part 3

Method 1: individual resampling of the samples:

```
set.seed(12347333)
n1 <- length(x1)
n2 <- length(x2)


t_stat <- function(x1, x2) {
  n1 <- length(x1)
  n2 <- length(x2)
  (mean(x1)-mean(x2))/sqrt(var(x1)/n1 + var(x2)/n2)
}
```

```r
t_x1x2 <- t_stat(x1, x2)

t_val_ind <- c()
for (i in 1:10000) {
 x1_boot <- sample(x1, n1, replace = TRUE)
 x2_boot <- sample(x2, n2, replace = TRUE)
 t_val_ind <- append(t_val_ind, t_stat(x1_boot, x2_boot))
}

p_val_ind <- mean(abs(t_val_ind) >= abs(t_x1x2))
ci_ind_95 <- quantile(t_val_ind, c(0.025, 0.975))
ci_ind_99 <- quantile(t_val_ind, c(0.005, 0.995))

cat("individual bootstrap p-value:", p_val_ind)
```

```
## individual bootstrap p-value: 0.9071
```

```r
cat("individual 95% CI:", ci_ind_95)
```

```
## individual 95% CI: -2.553482 1.630161
```

```r
cat("individual 99% CI:", ci_ind_99)
```

```
## individual 99% CI: -3.309845 2.170712
```

Method 2: Combining the samples and resample the combination:

```r
x1_cen <- x1 - mean(x1)
x2_cen <- x2 - mean(x2)
x1x2 <- c(x1_cen, x2_cen)
n <- n1 + n2

t_val_comb <- c()
for (i in 1:10000) {
  comb_sample <- sample(x1x2, n , replace = TRUE)
  x1_comb <- comb_sample[1:n1]
  x2_comb <- comb_sample[(n1+1):n]
  t_val_comb <- append(t_val_comb, t_stat(x1_comb, x2_comb))
}

p_val_comb <- mean(abs(t_val_comb) >= abs(t_x1x2))
ci_comb_95 <- quantile(t_val_comb, c(0.025, 0.975))
ci_comb_99 <- quantile(t_val_comb, c(0.005, 0.995))

cat("combined bootstrap p-value:", p_val_comb)
```

```
## combined bootstrap p-value: 0.9067
```

```r
cat("combined 95% CI:", ci_comb_95)
```

```
## combined 95% CI: -2.036275 2.007107
```

```r
cat("combined 99% CI:", ci_comb_99)
```

```
## combined 99% CI: -2.577079 2.583854
```

For both bootstrapping methods the p-value is > 0.9. This means we cannot reject the null-Hypothesis at either 0.05 or 0.01 significance levels.

part 4

```
comb <- c(x1, x2)
t_val_perm <- c()
for (i in 1:10000) {
  perm <- sample(comb)
  x1_perm <- perm[1:n1]
  x2_perm <- perm[(n1+1):n]
  t_val_perm <- append(t_val_perm, t_stat(x1_perm, x2_perm))
}

p_val_perm <- mean(abs(t_val_perm) >= abs(t_x1x2))
ci_perm_95 <- quantile(t_val_perm, c(0.025, 0.975))
ci_perm_99 <- quantile(t_val_perm, c(0.005, 0.995))

cat("permutation test p-value:", p_val_perm)
```

```
## permutation test p-value: 0.914
```

```
cat("permutation 95% CI:", ci_perm_95)
```

```
## permutation 95% CI: -1.999859 1.936013
```

```
cat("permutation 99% CI:", ci_perm_99)
```

```
## permutation 99% CI: -2.545791 2.444727
```

For the permutation version of the test the p-value is also > 0.9 and we clearly cannot reject the null-Hypothesis.

part 5

```
wilcoxon_stat <- function(x1, x2) {
  sum(rank(c(x1,x2))[1:length(x1)])
}

wilcoxon_orig <- wilcoxon_stat(x1, x2)

wilcoxon_vals_ind <- c()
for (i in 1:10000) {
  x1_wilcox <- sample(x1, n1, replace = TRUE)
  x2_wilcox <- sample(x2, n2, replace = TRUE)
  wilcoxon_vals_ind <- append(wilcoxon_vals_ind, wilcoxon_stat(x1_wilcox, x2_wilcox))
}

p_val_w_ind <- mean(wilcoxon_vals_ind >= wilcoxon_orig)
ci_ind_95 <- quantile(wilcoxon_vals_ind, c(0.025, 0.975))
ci_ind_99 <- quantile(wilcoxon_vals_ind, c(0.005, 0.995))

cat("individual bootstrap Wilcoxon p-value:", p_val_w_ind)
```

```
## individual bootstrap Wilcoxon p-value: 0.5102
```

```
cat("individual 95% CI:", ci_ind_95)
```

```
## individual 95% CI: 362 509
```

```r
cat("individual 99% CI:", ci_ind_99)
```

```
## individual 99% CI: 340 532
```

```r
wilcoxon_vals_comb <- c()
for (i in 1:10000) {
  combined_w <- sample(x1x2, n, replace = TRUE)
  x1_w_comb <- combined_w[1:n1]
  x2_w_comb <- combined_w[(n1 + 1):(n1 + n2)]
  wilcoxon_vals_comb <- append(wilcoxon_vals_comb, wilcoxon_stat(x1_w_comb, x2_w_comb))
}

p_val_w_comb <- mean(wilcoxon_vals_comb >= wilcoxon_orig)
ci_comb_w_95 <- quantile(wilcoxon_vals_comb, c(0.025, 0.975))
ci_comb_w_99 <- quantile(wilcoxon_vals_comb, c(0.005, 0.995))

cat("combined bootstrap Wilcoxon p-value:", p_val_w_comb)
```

```
## combined bootstrap Wilcoxon p-value: 0.6876
```

```r
cat("combined 95% CI:", ci_comb_w_95)
```

```
## combined 95% CI: 378.5 522.5
```

```r
cat("combined 99% CI:", ci_comb_w_99)
```

```
## combined 99% CI: 356.9975 543.5075
```

For both the individual bootstrap and the combined bootstrap the Wolcoxon-test p-value is > 0.5, meaning we cannot reject the null-Hypothesis.

part 6

```r
t_true <- t.test(x1,x2)
w_true <- wilcox.test(x1,x2)
print(t_true)
```

```
##
##  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
```

```r
print(w_true)
```

```
##
##  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
```

Using R's built-in *t.test()* and *wilcox.test()* yields similar results in terms of significance. This is due to the random sampling and different bootstrapping methods used. Still using any test we can certainly say that the null-Hypothesis cannot be rejected and $\mu_1 = \mu_2$ holds true.

Task 2

part 1

```
n <- 200
x1 <- rnorm(n, mean = 2, sd = sqrt(3))
x2 <- runif(n, min = 2, max = 4)
epsilon <- rt(n, df = 5)

y <- 3 + 2 * x1 + x2 + epsilon

x3 <- runif(n, min =-2, max = 2)
```

part 2

```
set.seed(12347333)
model <- lm(y ~ x1 + x2 + x3)

b_coef <- matrix(NA, nrow = 1000, ncol = 4)
x3_check <- c()

for (i in 1:1000) {
  samp_res <- sample(residuals(model), replace = TRUE)
  y_b <- fitted(model) + samp_res
  model_b <- lm(y_b ~ x1 + x2 + x3)

  b_coef[i, ] <- coef(model_b)

  x3_p <- summary(model_b)$coefficients["x3", "Pr(>|t|)"]
  x3_check <- append(x3_check, x3_p < 0.05)
}

ci_lower <- apply(b_coef, 2, quantile, 0.025)
ci_upper <- apply(b_coef, 2, quantile, 0.975)
ci <- cbind(ci_lower, ci_upper)
colnames(ci) <- c("lower", "upper")
rownames(ci) <- c("(Intercept)", "x1", "x2", "x3")

cat("ratio where x3 was marked as significant:", mean(x3_check))
```

```
## ratio where x3 was marked as significant: 0.069
```

```
print(ci)
```

```
##                 lower     upper
## (Intercept)  2.7944505 4.4291053
## x1           1.8551666 2.0295941
## x2           0.5883877 1.1124182
## x3          -0.1648752 0.1058175
```

As we can see, only in 6.9% of the refitted residual bootstrap models $x3$ was marked as significant. Furthermore

the 95% confidence interval contains zero. Therefore we can conclude that x3 can be excluded from the model, which is unsurprising since $x3$ is not incldued in the original y as well.

part 3

```
data <- data.frame(x1 = x1, x2 = x2, x3 = x3, y = y)

bp_coef <- matrix(NA, nrow = 1000, ncol = 4)
x3p_check <- c()
for (i in 1:1000) {
  bp_sample <- data[sample(1:200, replace = TRUE), ]
  model_bp <- lm(y ~ x1 + x2 + x3, data = bp_sample)

  bp_coef[i, ] <- coef(model_bp)

  x3_bp <- summary(model_bp)$coefficients["x3", "Pr(>|t|)"]
  x3p_check <- append(x3p_check, x3_bp < 0.05)
}

ci_lower_p <- apply(bp_coef, 2, quantile, 0.025)
ci_upper_p <- apply(bp_coef, 2, quantile, 0.975)
ci_p <- cbind(ci_lower_p, ci_upper_p)

colnames(ci_p) <- c("lower", "upper")
rownames(ci_p) <- c("(Intercept)", "x1", "x2", "x3")

cat("ratio where x3 was marked as significant:", mean(x3p_check))
```

```
## ratio where x3 was marked as significant: 0.063
```

```
print(ci_p)
```

```
##                 lower     upper
## (Intercept)  2.8365302 4.4091373
## x1           1.8429072 2.0298081
## x2           0.5973338 1.1152105
## x3          -0.1632048 0.1173853
```

This methods yields similar results as residual bootstrapping and for the same reasons we can exclude $x3$ from the model.

part 4

In the residual bootstrap we first calculated the linear regression model for the predicted values y and the predictors $x1, x2, x3$. Then the fitted values and residuals are calculated. In this method the predictors are kept the same over all iterations and only the residuals e are resampled to create new y-values by $\hat{y} + e$. These are then used with the same predicotrs to estimate a new model in every iteration. This preserves the model structure and is a good approach if the original model is already well-suited and specified and the residual errors are randomly/independently and identically distributed. Therefore this method has a lower variability and creates narrower confidence intervals.
In the second method we resampled from the original whole pairs of observations $(y, x1, x2, x3)$ and created in every iteration a new model using the resampled observations. This method makes no assumptions about the residuals and only assumes the observations are independently distributed. This approach can also be used when we don't know how well the model is actually specified. Due to this is has a greater variability but produces more robust estimates. The confidence interval tend to be larger than in the residual bootstrapping.

Since the model was in this case already well-specified adn the observations come from the actual predictors, except $x3$, both methods correctly can exclude $x3$.

Task 3

Bootstrapping is a resampling technique used to estimate the sampling distribution of a statistic like an estimator, a test statistic, or model coefficients by redrawing samples from the observed data. It can be applied both in parametric and non-parametric ways. Parametric bootstrapping assumes a specific model or distribution for the data. New samples are drawn from the assumed model by using the estimated parameters and used to generate a distribution for the desired statistic. In non-parametric bootstrapping no assumption is made about the underlying distribution of the data. Instead new samples are drawn directly from the observed data, creating a distribution of the statistic.
In bootstrapping first the model is fit or statistic computed. Then the data is resampled as emntioned before. In the next step the mdoel or statistic are recalculated for the resmapled data and a distribution is created. In the end, confidence intervals for the model/statistic can be computed.
The advantages of bootstrapping are that (in non-parametric bootstrapping) it does not rely on the data following a particular distribution, making it applicable in a wide variety of settings where traditional parametric methods might not be appropriate due to small sample sizes or non-normal distributed data. Additionally bootstrapping can be applied to estimators or statistics for which standard theoretical methods to derive confidence intervals like the CLT may not apply. Bootstrapping is in general a vrey simple method that also does not need any model assumptions to compute confidence intervals and is robust to misspecification as well.
Disadvantages are that it can be very computationally intensive since for large datasets samples have to be withdrawn mayn times and statistics/models recalcualted for each of these samples. Another disadvantage can be if the sample is not representative of the data. Since bootstrapping is resampling the sample and this contains outleirs or bias, the estimators of the bootstarpping will reflect this. Also in parametric bootstrapping the right model assumptions have to be made or results might be misleading. Furthermore if using non-parametric bootstrap it can sometiems lead to large confidence intervals since variabilty can be large.