Exercice n°5

Sampling Intervals for Models

Grégoire de Lambertye

2022-11-22

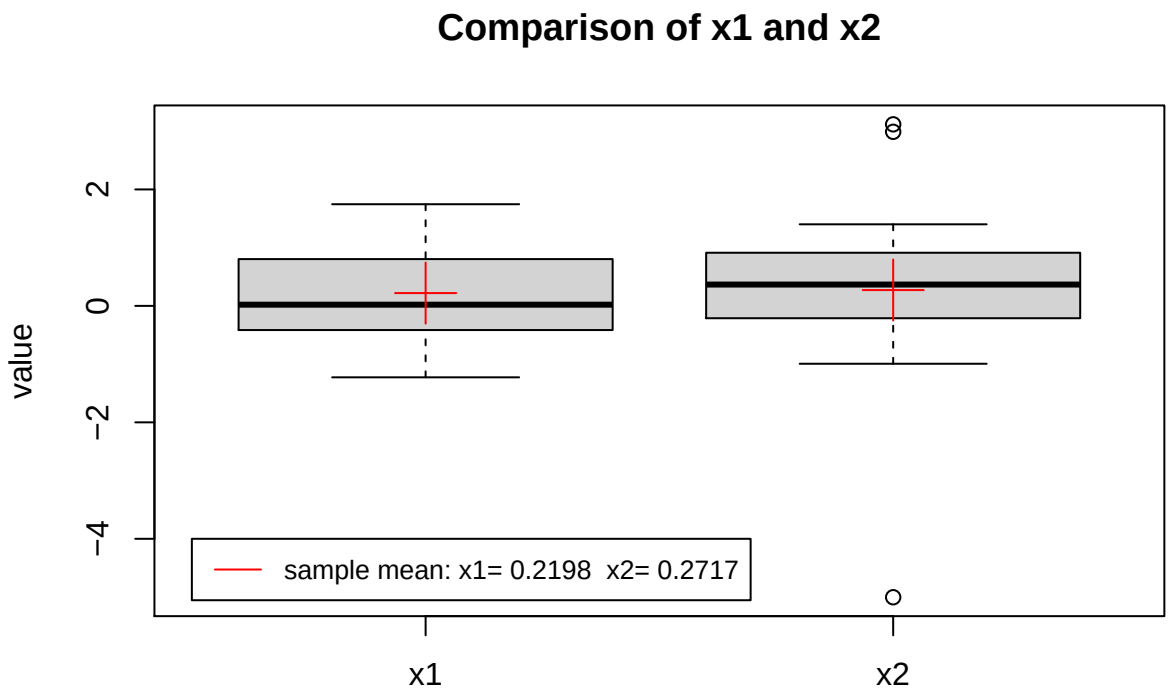
Task n°1

Consider a two sample problem and the hypothesis $H_0 : \mu 1 = \mu 2 vs H_1 : \mu 1 \neq \mu 2$, where $\mu 1$ and $\mu 2$ are the corresponding sample locations. The two samples are:

```
matricul.number <- 12202211
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
set.seed(matricul.number)
```

I.1 Plot the data in a way which visualises the comparison of means appropriately.

```
library(ggplot2)
par(mfrow=c(1,1))
boxplot(x1,x2, main = "Comparison of x1 and x2", ylab = "value", names = c("x1", "x2"))
points(x=1, y = mean(x1), col='red', pch=3, cex = 3)
points(x=2, y = mean(x2), col='red', pch=3, cex = 3)
legend(.5,-4 , legend=c(paste("sample mean: x1=",round(mean(x1),digits=4), " x2=",round(mean(x2),digits=
       col=c("red"), lty=1:2, cex=0.8)
```



I.2 Consider different sampling schemes

- Sampling with replacement from each group
- Centering both samples and then resample from the combined samples x1 and x2 for n1 and n2 times.

```r
#Sampling with replacement from each group
sampling_1 <- function(data,n, size=length(data)){
  samples = numeric(n)
  return(replicate(n, sample(data, size=size, replace=TRUE)))
}

#Centering both samples and then resample from the combined samples x1 and x2 for n1 and n2 times.
sampling_2 <- function(data1,data2,n1,n2, size1=length(data1),size2=length(data2)){
  #centreing
  data1 <- data1-mean(data1)
  data2 <- data2-mean(data2)
  combined <- c(data1,data2)
  sampl_x1 <- replicate(10000, sample(combined, size=size1, replace=TRUE))
  sampl_x2 <- replicate(10000, sample(combined, size=size2, replace=TRUE))
  return(list("sampl_x1" = sampl_x1, "sampl_x2" = sampl_x2))
}
```

Argue for choice what is more natural and which advantages or disadvantages may apply.
Our goal is to determine the whether the sample locations are the same. In the first case, we have the most common and the most natural way to do it using bootstrapping. We can estimate the parameter of x1 and

x2 with their bootstrap and use test statistic and compare their value to a test statistic computed with the original samples. With those comparison, we can obtain a p-value and conclude on our hypothesis. The second method is less common, we will use the exact same process as with method one but the main difference is that our test statistic values will contain less information on the mean since the bootsrap come from the same combined values. But the comparison with the original test statistic value will enable us to obtain a similar result as befor. The second method may give better results if the samples size are significantly different but the first one is easier to understand.

I.3 Bootstrap using both strategies mentioned above using the t-test statistic. Calculate the bootstrap p-value based on 10'000 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.

```
test_stat <- function(x1,x2){
  return((mean(x1) - mean(x2))/sd(x1))
}
```

```
#Generating bootstraps
n=10000
mean_x1 = mean(x1)
mean_x2 = mean(x2)
boot1_x1<- sampling_1(x1, n=n)
boot1_x2<- sampling_1(x2, n=n)

boot2 <- sampling_2(x1,x2,n,n)
boot2_x1 <- boot2[[1]]
boot2_x2 <- boot2[[2]]

tests_x1 <- test_stat(x1,x2) #H0: x1 = x2
```

```
#Methode 1
#test statistic
bootstap_tests_x1 <- numeric(n)
p_value_x1 <- 1

for(i in 1:n){
  bootstap_tests_x1[i] <- test_stat(boot1_x2[,i],x2)
  if(abs(bootstap_tests_x1[i]) > abs(tests_x1)){
    p_value_x1 <- p_value_x1 + 1}
}
p_value_x1 <- p_value_x1/(n+1)

# CI
a = .025
b = 1-a

ci_1_x1 <-  quantile(bootstap_tests_x1,c(a,b))

a = .005
b = 1-a

ci_1_x1_99 <-  quantile(bootstap_tests_x1,c(a,b))
```

Method n°1

hypothese	H0:x1=x2	
p_value	0.8060194	
CI	2.5%	97.5%
95%	-0.4009562	0.5834701
CI	.5%	99.5%
99%	-0.5095706	0.7340501

The p_value we obtained is very high, based on that we can not reject the null hypothesis, x1 and x2 might have the same location.

```
#Method 2
# h0: x1=x2
p_value2_x1 <- 1
t_values1 = numeric(n)

for(i in 1:n){
    t_values1[i] <- test_stat(boot2_x1[,i],boot2_x2[,i])  # bootstrap test statistic
    if( (abs(t_values1[i]) > abs(tests_x1)) ){
      p_value2_x1 <- p_value2_x1 + 1}
}

p_value2_x1 <- p_value2_x1/(n+1)


# CI
a = .025
b = 1-a

ci_2_x1 <-  quantile(t_values1,c(a,b))

a = .005
b = 1-a

ci_2_x1_99 <-  quantile(t_values1,c(a,b))
```

Method n°2

hypotheses	H0:x1=x2	
p_value	0.8453155	
CI	2.5%	97.5%
95%	-0.6176585	0.812572
CI	.5%	99.5%
99%	-0.8629164	1.0921075

The p_value we obtained is very high, based on that we cannot reject the null hypothesis, x1 and x2 might have the same location.

I.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.

A permutation version of the test would be a permutation of the values between x1 and x2.

```
indices <- seq(from = 1, to = length(x1) + length(x2), by = 1)
combined <- c(x1, x2)

t_stat_x1 <- c()

for (i in 1:n) {
  x1_indices <- sample(indices, size = length(x1), replace = FALSE)
  boot_x1 <- combined[x1_indices]
  t_stat_x1 <- c(t_stat_x1, test_stat(boot_x1,x2))
}

p_value_x1 <- 1

for(i in 1:n){
  if(abs(t_stat_x1[i]) > abs(tests_x1)){
    p_value_x1 <- p_value_x1 + 1}
}
p_value3_x1 <- p_value_x1/(n+1)

# CI
a = .025
b = 1-a

ci_3_x1 <-  quantile(t_stat_x1,c(a,b))

a = .005
b = 1-a

ci_3_x1_99 <-  quantile(t_values1,c(a,b))
```

Permutation

hypotheses	H0:x1=x2	
p_value	0.6977302	
CI	2.5%	97.5%
95%	-0.2964934	0.3178029
CI	.5%	99.5%
99%	-0.8629164	1.0921075

Our p_value is once again very high, we still cannot reject H0.

I.5 The Wilxocon 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.

```
# First method
Wilcox_boot_x1 <- c()
for(i in 1:n){
  Wilcox_boot_x1 <- c(Wilcox_boot_x1, wilcox.test(boot1_x1[,i], x2, paired = FALSE)$statistic)
}
will_original <- wilcox.test(x1, x2, paired = FALSE)$statistic

p_value_x1 <- 1
for(i in 1:n){
  if(abs(Wilcox_boot_x1[i]) > abs(will_original)){
    p_value_x1 <- p_value_x1 + 1}
}
p_value_x1 <- p_value_x1/(n+1)

# Saecond method
Wilcox_boot_x1_2 <- c()
for(i in 1:n){
  Wilcox_boot_x1_2 <- c(Wilcox_boot_x1_2, wilcox.test(boot2_x1[,i], x2, paired = FALSE)$statistic)
}
p_value_x1_2 <- 1
for(i in 1:n){
  if(abs(Wilcox_boot_x1_2[i]) > abs(will_original)){
    p_value_x1_2 <- p_value_x1_2 + 1}
}
p_value_x1_2 <- p_value_x1_2/(n+1)

# CI
a = .025
b = 1-a
ci_Wilcox_boot_x1 <-  quantile(Wilcox_boot_x1,c(a,b))
Wilcox_boot_x1_2 <-  quantile(Wilcox_boot_x1_2,c(a,b))
a = .005
b = 1-a
ci_Wilcox_boot_x1_99 <-  quantile(Wilcox_boot_x1,c(a,b))
Wilcox_boot_x1_2_99 <-  quantile(Wilcox_boot_x1_2,c(a,b))
```

Wilcox

Methode	sampling 1		sampling 2	
p_value	0.4966503		0.1794821	
CI	2.5%	97.5%	2.5%	97.5%
95%	136	228	115	207
CI	.5%	99.5%	.5%	99.5%
99%	122	244	115.46	206.54

I.6 Compare your results to the results using t.test and wilcox.test.

```
wilcox <- wilcox.test(x1, x2, paired = FALSE)
wilcox
```

```
##
##  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
```

```
ttest <- t.test(x1,x2, paired = FALSE)
ttest
```

```
##
##  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
```

We can make the same conclusion with the boostraps we did and the result from the tests.

Task n°2

Consider the model $y = 3 + 2 * x1 + x2 + e$ where x1 comes from a normal distribution with mean 2 and variance 3, x2 comes from a uniform distribution between 2 and 4 and e 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.

```
model <- function(n){
  x1 <-  rnorm(n, mean = 2, sd = 3)
  x2 <- runif(n, min = 2, max = 4)
  eps <- rt(n, df = 5)
  x3 <- runif(n, min = -2, max = 2)
  y <- 3+ 2*x1 + x2 + eps
  df <- data.frame(y,x1,x2,x3)
  return(df)
}

x3  <- function(){
  return(runif(1,-2,2))
}
```

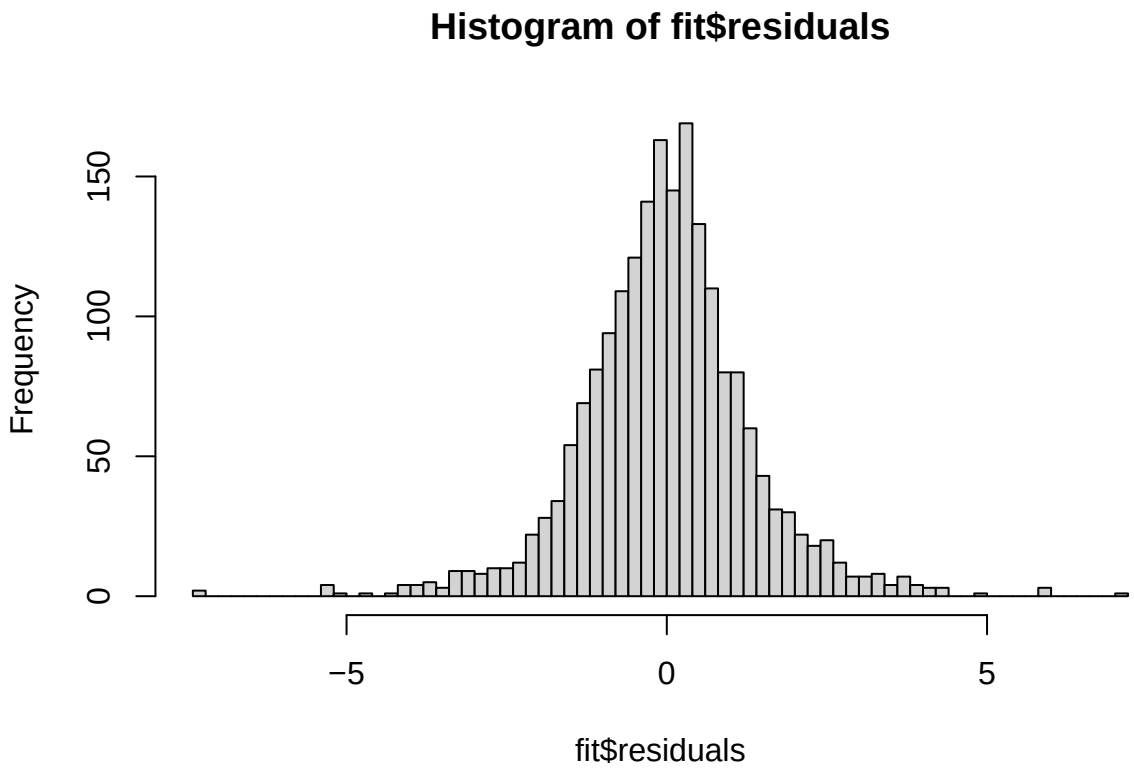
II.1 Create a sample of size 200 from the model above and for the independent predictor x3.

```
n <- 2000
data <- model(n)
```

II.2 Do residual bootstrap for linear regression and fit the model y follow x1+x2+x3 . Get the percentile CI for the coefficients. Can you exclude x3 ?

```
library(boot)

fit <- lm(y~., data=data)
hist(fit$residuals, breaks = 100)
```



The residual distribution seems to follow a normal distribution, we will use a parametric bootsrap.

```
reg.fun <- function(x){
  return(coef(lm(y ~ x1+x2+x3, data = x)))
}
reg.sim <- function(x, resi){
  x$y <- (resi$yhat + sample(resi$res, replace = TRUE))
  return(x)
}
```

```
res <- resid(fit)
yhat <- fitted(fit)
reg2 <- data.frame(yhat, res)

fit.boot <- boot(data, reg.fun, R=1000, sim="parametric", ran.gen=reg.sim, mle = reg2)
x1_coef_ci <- quantile(fit.boot$t[,2], c(0.005,0.025,0.975, 0.995))
x2_coef_ci <- quantile(fit.boot$t[,3], c(0.005,0.025,0.975, 0.995))
x3_coef_ci <- quantile(fit.boot$t[,4],  c(0.005,0.025,0.975, 0.995))


df <- data.frame(
  x1 = x1_coef_ci,
  x2 = x2_coef_ci,
  x3 = x3_coef_ci
)

knitr::kable(df, caption="CI for paramters")
```

Table 5: CI for paramters

	x1	x2	x3
0.5%	1.977732	0.8011341	-0.0614037
2.5%	1.984048	0.8509686	-0.0508927
97.5%	2.021565	1.0487025	0.0532977
99.5%	2.027482	1.0748493	0.0601089

Thanks to the confidence interval in the previous graph we can assume that x3 is not significant and we can exclude x3.

II.3 Do pairs bootstrap for linear regression and fit the model yfollowx1+x2+x3 . Get the percentile CI for the coefficients. Can you exclude x3 ?

```
fit <- lm(y ~ x1+x2+x3, data = data)
res <- resid(fit)
yhat <- fitted(fit)
reg_model <- data.frame(yhat, res)
reg_fun <- function(x, idx){
  return(coef(lm(y ~ x1+x2+x3, data = x[idx, ])))
}

fit.boot_pairs <- boot(data, reg_fun, R = 1000)

bot_pair_sol <- fit.boot_pairs$t
x1_coef_pairs <- quantile(bot_pair_sol[,2], c(0.005,0.025,0.975, 0.995))
x2_coef_pairs <- quantile(bot_pair_sol[,3], c(0.005,0.025,0.975, 0.995))
x3_coef_pairs <- quantile(bot_pair_sol[,4], c(0.005,0.025,0.975, 0.995))

df <- data.frame(
  x1 = x1_coef_ci,
```

```
  x2 = x2_coef_ci,
  x3 = x3_coef_ci
)

knitr::kable(df, caption="CI for paramters")
```

Table 6: CI for paramters

	x1	x2	x3
0.5%	1.977732	0.8011341	-0.0614037
2.5%	1.984048	0.8509686	-0.0508927
97.5%	2.021565	1.0487025	0.0532977
99.5%	2.027482	1.0748493	0.0601089

Our result are very similar to what we obtained with residual bootstrap, we can also conclude that x3 is non significant.

II.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.

Th main difference between this 2 approach is the assumption in the first case that our residuals follow a normal distribution. In the second case, no assumption is made. In the residual bootrap, the new sample are generated wheras in the pair bootstrap, they are picked in the computed residuals. Asymptotically the 2 methods are equivalent when the model is correctly specified. They perform quite differently for small samples and for the correct model the residual version is more efficient. The pairs strategy is often considered more robust when the model is misspecified.

Task n°3 :Summaries the bootstrapping methodology, its advantages and disadvantages based on your exercises for constructing parametric and non-paramteric confidence bounds for estimators, test statistics or model estimates.

To put it in a nutshell, bootstrapping consist of approaching a statistic magnitude (estimators, test statistics or model estimates) with samples. These samples called bootstrap are computed by randomly sampling the original dataset with replacement or they are generated under the hypothesis the values used to compute this magnitude follow a normal distribution. The method offers a very convenient way to compute confidence intervals. Bootstrap can give good point of view on the magnitude as long as the calcul are not to long.