Computational Statistics
732A90

Lab 5

Albin Västerlund
albva223

Eric Herwin
Erihe068

Contents

Assigment 1: Hypothesis testing **1**
1 . . . 1
2 . . . 1
3 . . . 2
4 . . . 4
5 . . . 7
a) . . . 7
b) . . . 8
c) . . . 9

Assigment 2: Bootstrap, jackknife and confidence intervals **12**
1 . . . 12
2 . . . 13
3 . . . 15
4 . . . 16

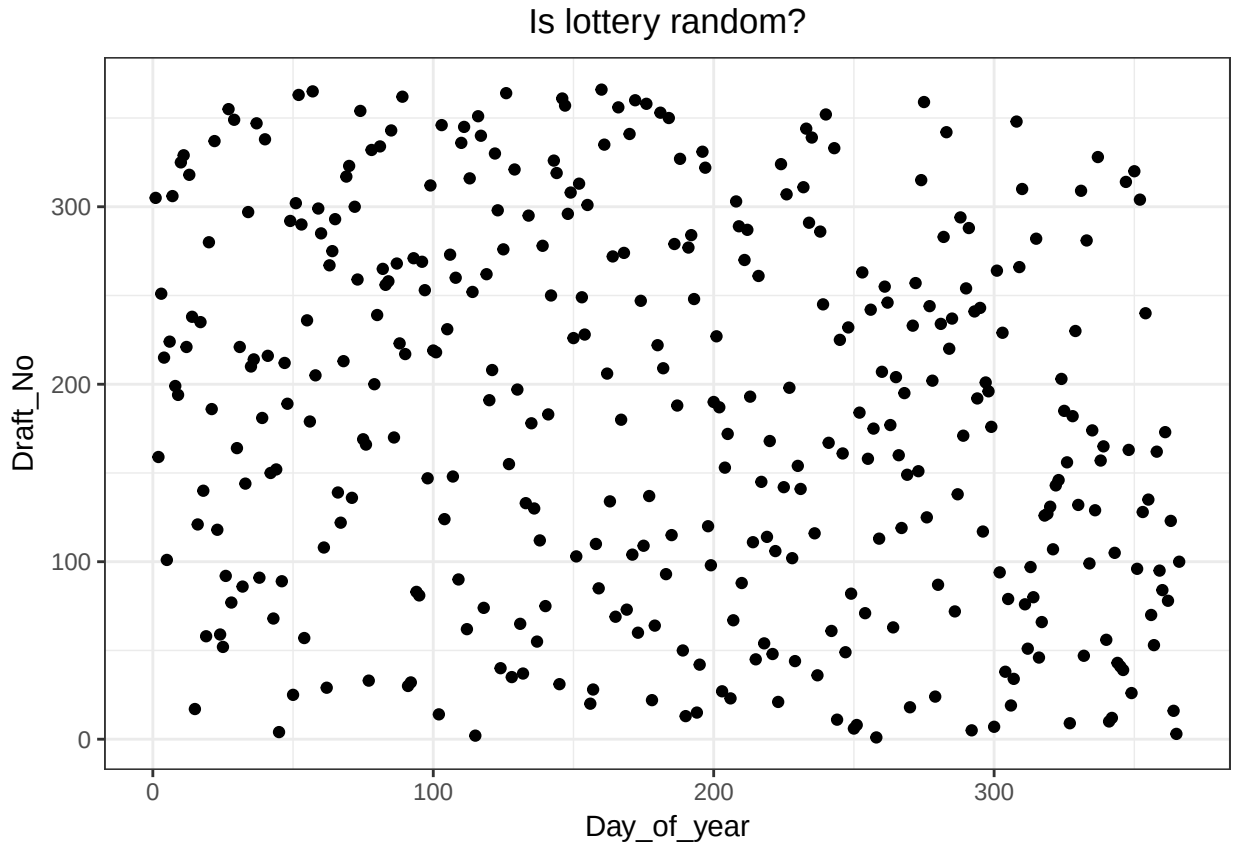
Appendix **17**
R-code . . . 17

Assigment 1: Hypothesis testing

1

In this task we will make a scatterplot of Y=Draft_No versus X=Day_of_year.

```
ggplot(lottery,aes(y=Draft_No,x=Day_of_year))+
  geom_point()+
  theme_bw()+
  ggtitle("Is lottery random?")+
  theme(plot.title = element_text(hjust = 0.5))
```



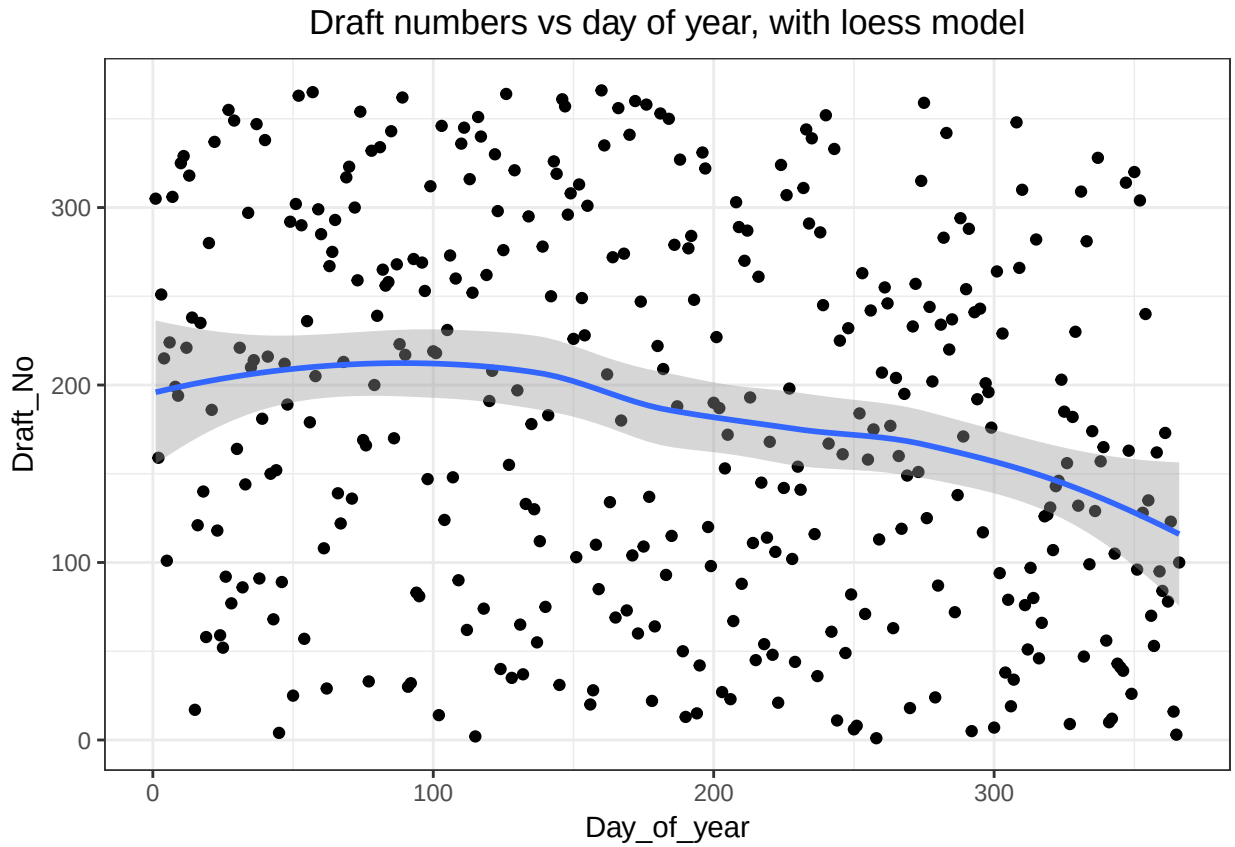
From looking at the plot we do only see randomness in the data.

2

In this task we will compute an estimate $\hat{Y}$ of the expected response as a function of X by using a loess smoother. We will plot the result of the `loess()` against the original data.

```
ggplot(lottery,aes(y=Draft_No,x=Day_of_year))+
  geom_point()+
  theme_bw()+
```

```
  ggtitle("Draft numbers vs day of year, with loess model")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_smooth(method = "loess")
```



We can see that the `loess()` estimation finds a visible trend that might indicate that the lottery is not random.

3

In this task we will use this test statistics:

$$T = \frac{\hat{Y}(X_b) - \hat{Y}(X_a)}{X_b - X_a}, \quad where\ X_b = argmax_x Y(X),\ X_a = argmin_y Y(X)$$

If the test statistic is very large its indicates on a positive trend while if its very small its indicats on a negative trend.

We will test the hypothesis:

$$H_0 : T \geq 0 \quad (i.e\ not\ a\ negative\ trend)$$
$$H_1 : T < 0\ (i.e\ a\ negative\ trend)$$

We are investigating in the right side of the distribution. Which means that we calculate the p-value in this way:

$$p-value = \frac{count(T_i \geq T)}{B} = 1 - \frac{count(T_i < T)}{B}$$

And in this case is T=0 and T_i is the observed test-statistics from for example a bootstrap.

So we will perform a non-parametric bootstrap with B = 2000.

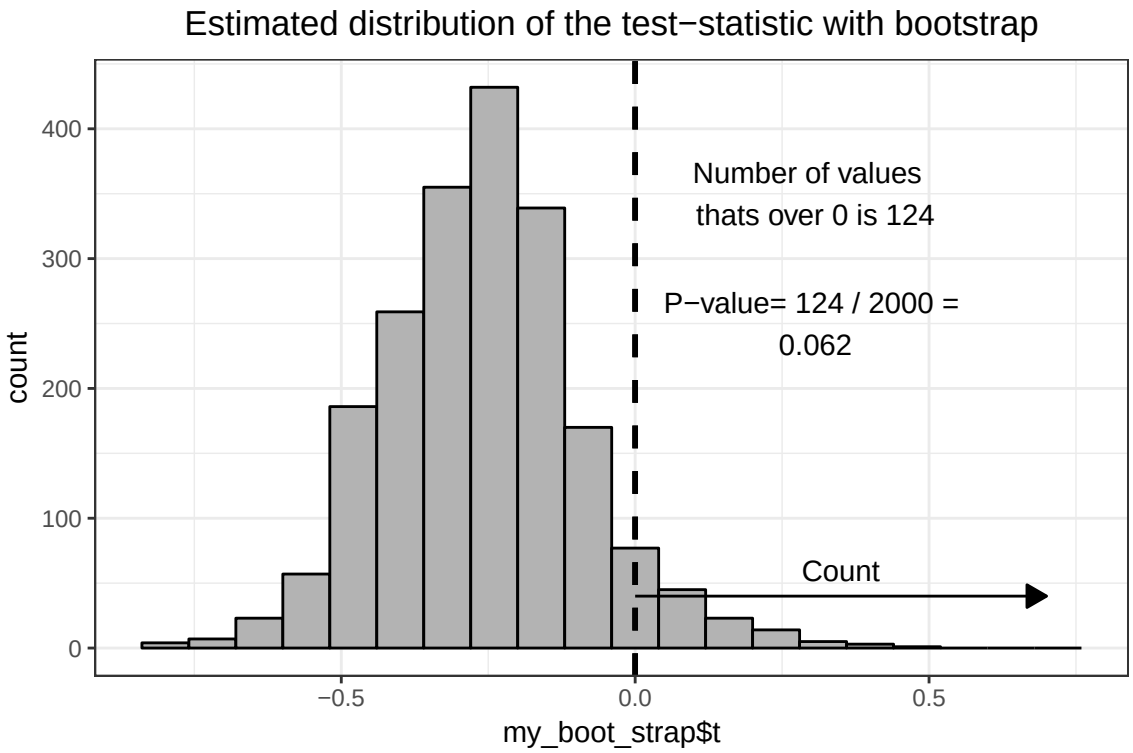
First we do a function that we will put into the `boot()` function.

```
my_bot_fun <- function(data, index) {
  new_data <- data[index, c(4, 5)]
  modelen <- loess(Draft_No ~ Day_of_year, new_data)

  Xa_pos <- which.min(new_data$Draft_No) #Which min on Y
  Xb_pos <- which.max(new_data$Draft_No) #Which max on Y

  Xa <- new_data$Day_of_year[Xa_pos] #The X value that gives the min Y
  Xb <- new_data$Day_of_year[Xb_pos] #The X value that gives the max Y

  Y_Xa <- modelen$fitted[Xa_pos]
  Y_Xb <- modelen$fitted[Xb_pos]

  T_value <- (Y_Xb - Y_Xa) / (Xb - Xa) #The test statistic

  return(T_value)
}
```

When we have the function we can perform a bootstrap and extract the p-value. We will also plot whats happens in a histogram.

```
B=2000
my_boot_strap <- boot(data = lottery, statistic = my_bot_fun, R = B) #Do the bootsprat

T_x<-0
p_value<-sum(my_boot_strap$t>=T_x)/B #get the p-value. Now its right

#1-sum(my_boot_strap$t<T_x)/B #We dident mix up p-value and power
                              #We mixed it up with this formula which gives the same results

text_nr1<-paste("Number of values \n thats over 0 is",sum(my_boot_strap$t>T_x))
text_nr2<-paste("P-value=",sum(my_boot_strap$t>T_x),"/",B,"=\n",sum(my_boot_strap$t>T_x)/B)

ggplot(mapping = aes(x=my_boot_strap$t))+
  geom_histogram(col="grey0",fill="grey70",bins=20)+
  theme_bw()+
  ggtitle("Estimated distribution of the test-statistic with bootstrap")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_vline(xintercept = 0,linetype="dashed",size=1)+
  annotate("text",0.3,350,label=text_nr1)+
  annotate("text",0.3,250,label=text_nr2)+
```

```
geom_vline(xintercept = T_x,linetype="dashed",size=1)+
geom_line(aes(x=c((T_x+0.7),T_x),y=40),
          arrow = arrow(length=unit(0.30,"cm"),
                        ends="last", type = "closed"))+
annotate("text",T_x+0.35,60,label="Count")
```



We can see that the p-value is 6.2%. The p-value is low big which indicats that we have a negative trend in or data.

4

In this task will implement a function depending on `data` and `B` that tests if the lottery is random or not. The hypothesis:

$$H_0 : T = 0 \quad (Lottery\ is\ random)$$
$$H_1 : T \neq 0 \quad (Lottery\ is\ not\ random)$$

We can see by looking at the hypothesis thats its a two-sided test (one side for positive trend and one for negativ trend).

We are investigating in the both side of the distribution. Which means that we calculate the p-value in this way:

$$p-value = \frac{count(|T_i| \geq |T|)}{B} = 1 - \frac{count(|T_i| < |T|)}{B}$$

These hyposesis will be tested with a permutation test with statistics T. This test will be performed with B = 2000.

We will start do the function.

```
tst <- function(data, B){

  tmp_data <- data #make a temporary data set
  samp_val <- c()  #We will save the test statistic in this variable

  # Get the distribution of T when X is "garbage" ####
  for(i in 1:B){
    ind <- sample(1:nrow(data), nrow(data))   #Suffle around the X
    tmp_data$X <- data$X[ind]                 #Suffle around the X

    modl <- loess(Y ~ X, tmp_data)            #Make a model when X is "garbage"

    x_b <- data$X[which.max(data$Y)]          #Make the test statistic
    x_a <- data$X[which.min(data$Y)]          #Make the test statistic

    y_x_b <- predict(modl, x_b)               #Make the test statistic
    y_x_a <- predict(modl, x_a)               #Make the test statistic

    samp_val[i] <- (y_x_b - y_x_a)/(x_b- x_a) #Save the test statistic
  }

  # Calc test statistc
  all_data_model <- loess(Y ~ X, data)     #Model when X is not "garbage"
  x_b <- data$X[which.max(data$Y)]         #Make the test statistic
  x_a <- data$X[which.min(data$Y)]         #Make the test statistic
  y_x_b <- predict(all_data_model, x_b)    #Make the test statistic
  y_x_a <- predict(all_data_model, x_a)    #Make the test statistic

  T_test_statistic <- (y_x_b - y_x_a)/(x_b- x_a) #Save our obtain test-statistic

  #This formula works if the disturbution is symmetric
  #Like for example in t-distribution and normal-distribution
  #So it dont work in our case
  #p_value <- (sum(samp_val < T_test_statistic) / B) * 2 #Get the p-value.
                                                         #We multiply it with 2
                                                         #because its a two-sided hypothesis.

  #The new way to calculate the p-value
  p_value <- (sum(abs(samp_val) > abs(T_test_statistic)) / B)

  ############ Down from here its just to make a histogram of the data. ########
  nr1_text<-paste("Number of values \n thats over |",
                  round(T_test_statistic,2),"| is ",
                  sum(abs(samp_val)>abs(T_test_statistic)),sep="")
```

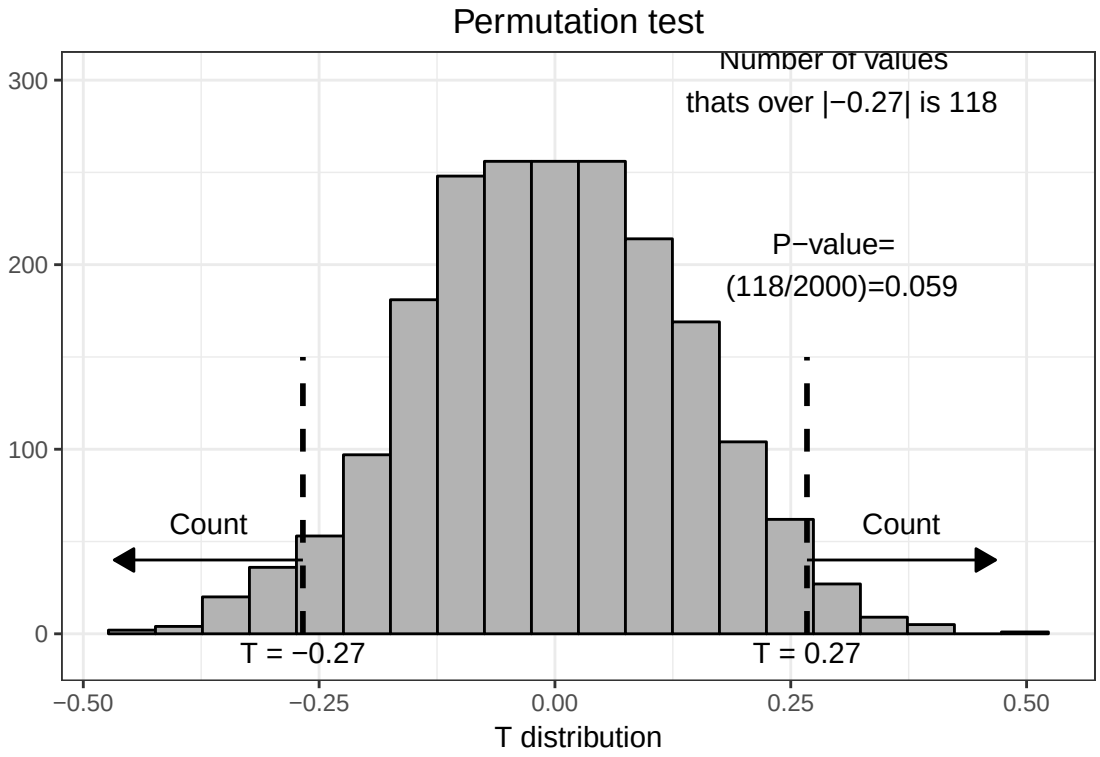
```
  nr2_text<-paste("P-value= \n (",
                  sum(abs(samp_val)>abs(T_test_statistic)),"/",B,")=",
                  p_value,sep="")

  ggplot_return<-ggplot()+
    geom_histogram(aes(x=samp_val),col="grey0",fill="grey70",bins=20)+
    #geom_vline(xintercept = T_test_statistic,linetype="dashed",size=1,y=0)+
    geom_segment(aes(x=T_test_statistic,y=0,xend=T_test_statistic,yend=150),linetype=8,size=1)+
    geom_segment(aes(x=-T_test_statistic,y=0,xend=-T_test_statistic,yend=150),linetype=8,size=1)+
    labs(x="T distribution",y="",title="Permutation test")+
    theme_bw()+
    theme(plot.title = element_text(hjust = 0.5))+
    annotate("text",T_test_statistic ,-10,label=paste("T =",round(T_test_statistic,2)))+
    annotate("text",-T_test_statistic,-10,label=paste("T =",-round(T_test_statistic,2)))+

    annotate("text",0.30,300,label=nr1_text)+
    annotate("text",0.30,200,label=nr2_text)+
    geom_line(aes(x=c(T_test_statistic-0.2,T_test_statistic),y=40),
            arrow = arrow(length=unit(0.30,"cm"),
                          ends="first", type = "closed"))+
    geom_line(aes(x=c(-T_test_statistic+0.2,-T_test_statistic),y=40),
            arrow = arrow(length=unit(0.30,"cm"),
                          ends="last", type = "closed"))+
    annotate("text",T_test_statistic-0.1,60,label="Count")+
    annotate("text",-T_test_statistic+0.1,60,label="Count")
  ########################

  svar <- list(p_value, ggplot_return, samp_val, T_test_statistic) #Save a list that the

  return(svar)    #function return
}
```

When we have a function that can make the permutation test we appley it on our data.

```
lottery2 <- data.frame(Y = lottery$Draft_No, X = lottery$Day_of_year) #Make a data.frame
Permutation_tests <- tst(lottery2, 2000)
Permutation_tests_1.4<-Permutation_tests  #save for assigment 1.5 power test
```

The p-value and histogram:

```
Permutation_tests[[2]] #Histogram
```



We can see that the p-value is 5.9%. If we had a significance level 10% we would reject H_0 and say that the data is not random. Which indicats that draft number is depending on the day of the year. In our case if its a day in the end of the year the draft number will decrease.

5

a)

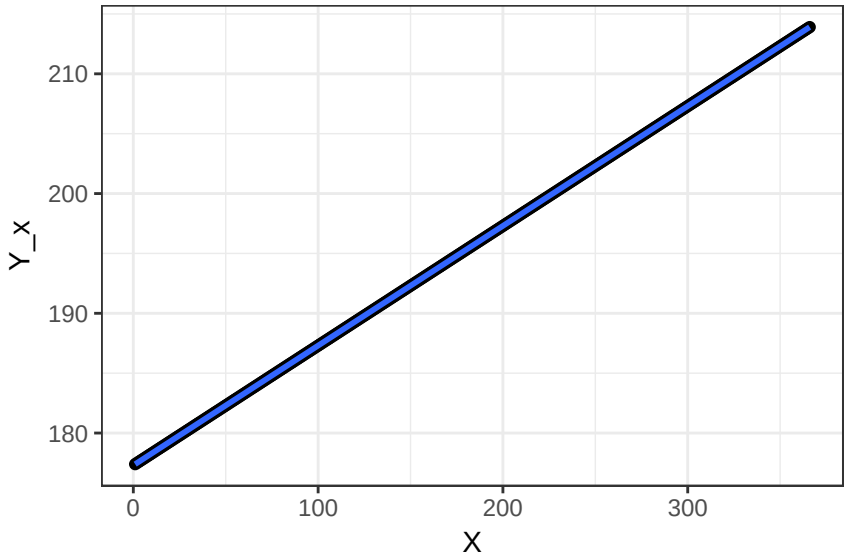
Here we will use the same data X as before and and simulate new Y values:

$$Y(x) = max(0, min(\alpha x + \beta, n)), \quad n = 366,\ \alpha = 0.1,\ \beta \sim N(183, sd = 10)$$

Code for simulate the new Y values

```
X <- lottery$Day_of_year                #Or X values
beta <- rnorm(1, 183, 10)               #Beta value
alpha <- 0.1                            #Alpha value
alpha_beta_value <- alpha * X + beta    #inner, inner values



Y_x <- NULL                               #We save our Y(x) in this variable

for (i in seq_along(alpha_beta_value)) {  #For i in inner,inner values
  a<-min(alpha_beta_value[i],366)         #Get the inner values
  Y_x[i] <- max(0, a)                     #Get the values=Y
```

```
}


ggplot(mapping = aes(x=X,y=Y_x))+          #Plot the obtain data
  geom_point()+
  theme_bw()+
  geom_smooth(method = "loess")
```



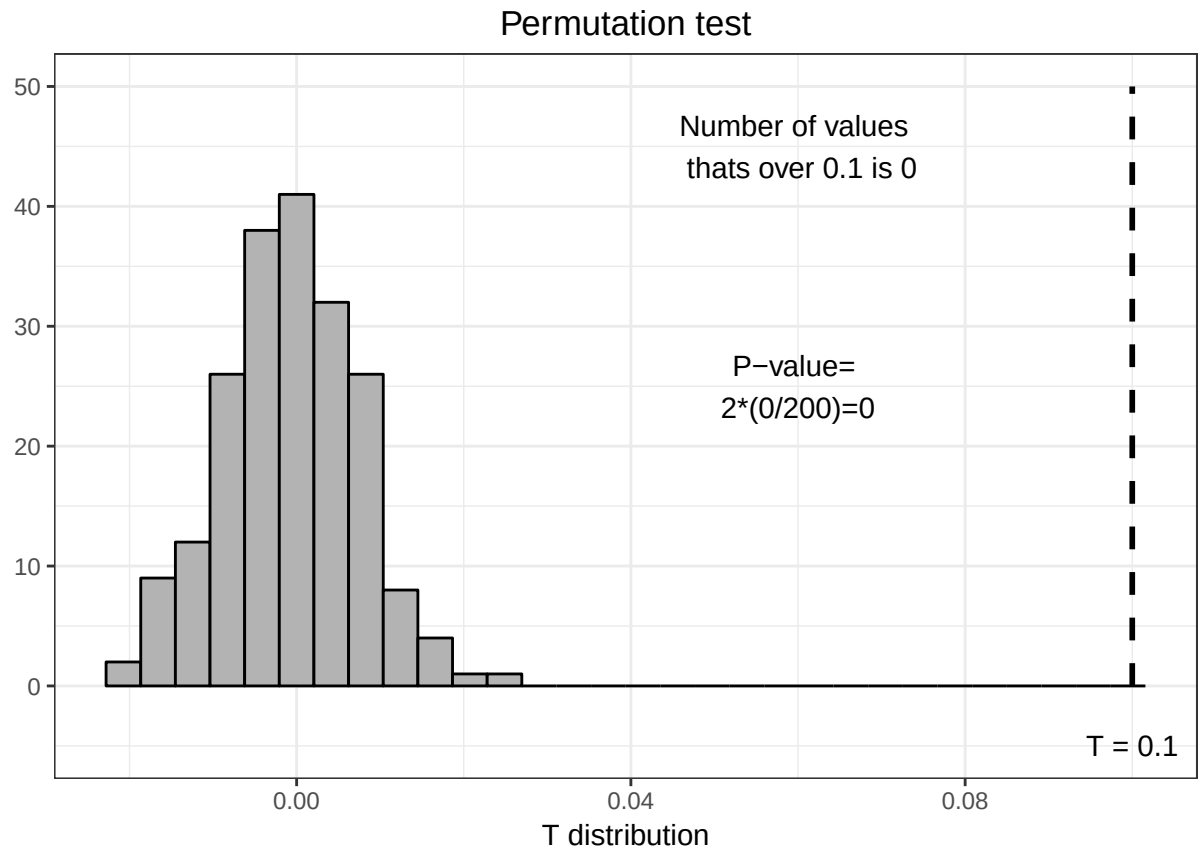
Here is the new simulated Y values vs the old X values.

b)

We will now perform the permutation test with B=200 and see if the test statistic get rejected or not (trend or not in the data).

We will do this with a function called `tst2()`. Very simular to the function `tst()` in assigment 1.4. See appendix for code.

```
# Make the Permutation tests ####
data <- data.frame(Y = Y_x, X = X)    #set the data in a data.frame
Permutation_tests <- tst2(data, 200) #Do permutation tests, look in appendix for
                                      #code for tst2 (simular to tst)

Permutation_tests[[2]] #histogram
```



As we saw in the graph in 1.5a there was a clear positive trend in the data and therefore gets a large T-statistic, which gives us a small p-value. This indicats that $Y(x)$ is depended on x (which is quite clear).

c)

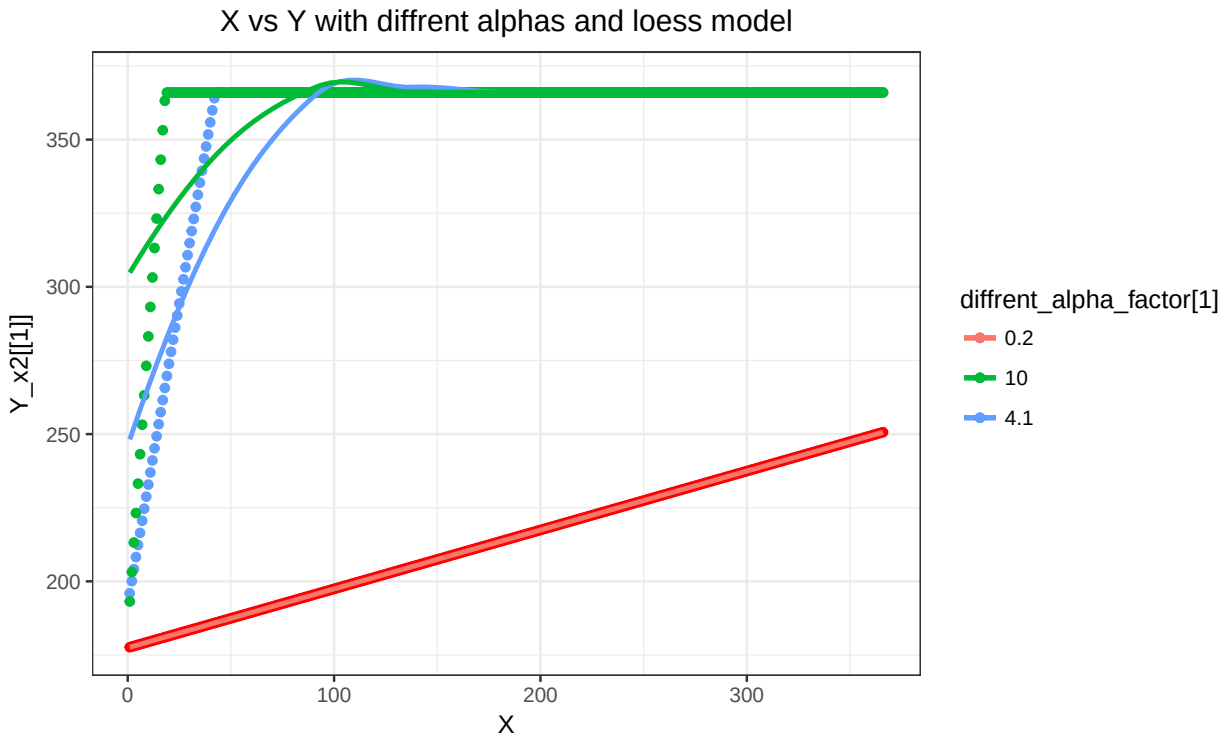
In this step we will repeat steps 5a-5b with

$$\alpha = 0.2, 0.3, ..., 10$$

.

```
# Fix parameters
diffrent_alpha <- seq(0.2, 10, 0.1) #diffrent alpha
X <- lottery$Day_of_year            #set our X values
Y_x2 <- list()                      #Save our Y(x) with diffrent alphas

# Calc the Y-values
for (j in seq_along(diffrent_alpha)) { #Loop over all alphas (its 99 diffrent alphas)
  alpha <- diffrent_alpha[j]           #Set the current alpha
  beta <- rnorm(1, 183, 10)            #Sample a beta

  alpha_beta_value <- alpha * X + beta #inner, inner values

  temp_y_x <- NULL                     #tempurary save Y(x) given a alpha in this varaiable

  for (i in seq_along(alpha_beta_value)) { #Loop over all inner,inner values
```

```
    a <- min(alpha_beta_value[i], 366)      #Get the inner values
    temp_y_x[i] <- max(0, a)                #Get Y(x) values
  }

  Y_x2[[j]] <- temp_y_x      #Save the current Y(x) in a list
}

diffrent_alpha_factor<-as.factor(diffrent_alpha)
ggplot(mapping = aes(x=X))+
  geom_point(aes(y=Y_x2[[1]],col=diffrent_alpha_factor[1]),col="red")+
  geom_point(aes(y=Y_x2[[40]],col=diffrent_alpha_factor[40]))+
  geom_point(aes(y=Y_x2[[99]],col=diffrent_alpha_factor[99]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[1]],
                            col=diffrent_alpha_factor[1]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[40]],
                            col=diffrent_alpha_factor[40]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[99]],
                            col=diffrent_alpha_factor[99]))+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(title="X vs Y with diffrent alphas and loess model")
```

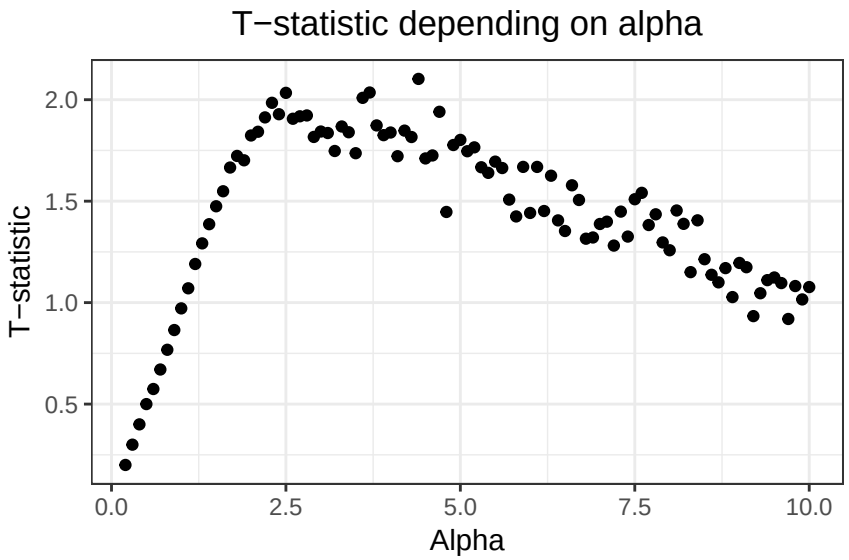


Here is three sets of Y with diffrent alphas ($\alpha = 0.2,\ 4.1,\ 10$). We can see that in all cases that there is a positive trend in the data. When we have a large alpha there is a lot of Y values that is set to 366. We can expect that we will get really low p-values in all permutation test and probably reject all H_0 as well.

```
p_values_ass_1.5 <- c() #variable to save p-values
test_statistic_1.5<-c() #variable to save T-statistic

for (i in 1:length(Y_x2)) {                    #Loop for every Y(x) (its 99 of them)
  data <- data.frame(Y = Y_x2[[i]], X = X)     #Put the current values in a data.frame
  perm_test<-tst2(data, 200)                   #Make a permutation test. (It return a list)

  p_values_ass_1.5[i]  <- perm_test[[1]]       #Save the p-value
  test_statistic_1.5[i] <- perm_test[[3]]      #Save the T-statistic
}
p_values_ass_1.5
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
test_statistic_dep_alpha<-ggplot(mapping = aes(x=diffrent_alpha,y=test_statistic_1.5))+
  geom_point()+
  theme_bw()+
  labs(x="Alpha",y="T-statistic",title="T-statistic depending on alpha")+
  theme(plot.title = element_text(hjust = 0.5))
test_statistic_dep_alpha
```



In all cases the p-value is set to 0. The test statistic start to increase up to around $\alpha = 3$. After that the T-value start to decrease again.

A bigger T-value does not necessarily mean a smaller p-value because the distribution of T-statistic is diffrent. But still, it indicates on a smaller p-value.

We calculate the power through Power=1-*type II* error.

```
#Significns level
alpha<-0.1

#numer of type 2 errors
type2_error<-sum((p_values_ass_1.5>alpha))
```

```
#proportion of type
type2_error<-type2_error/length(p_values_ass_1.5)

# The power
1-type2_error
```

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

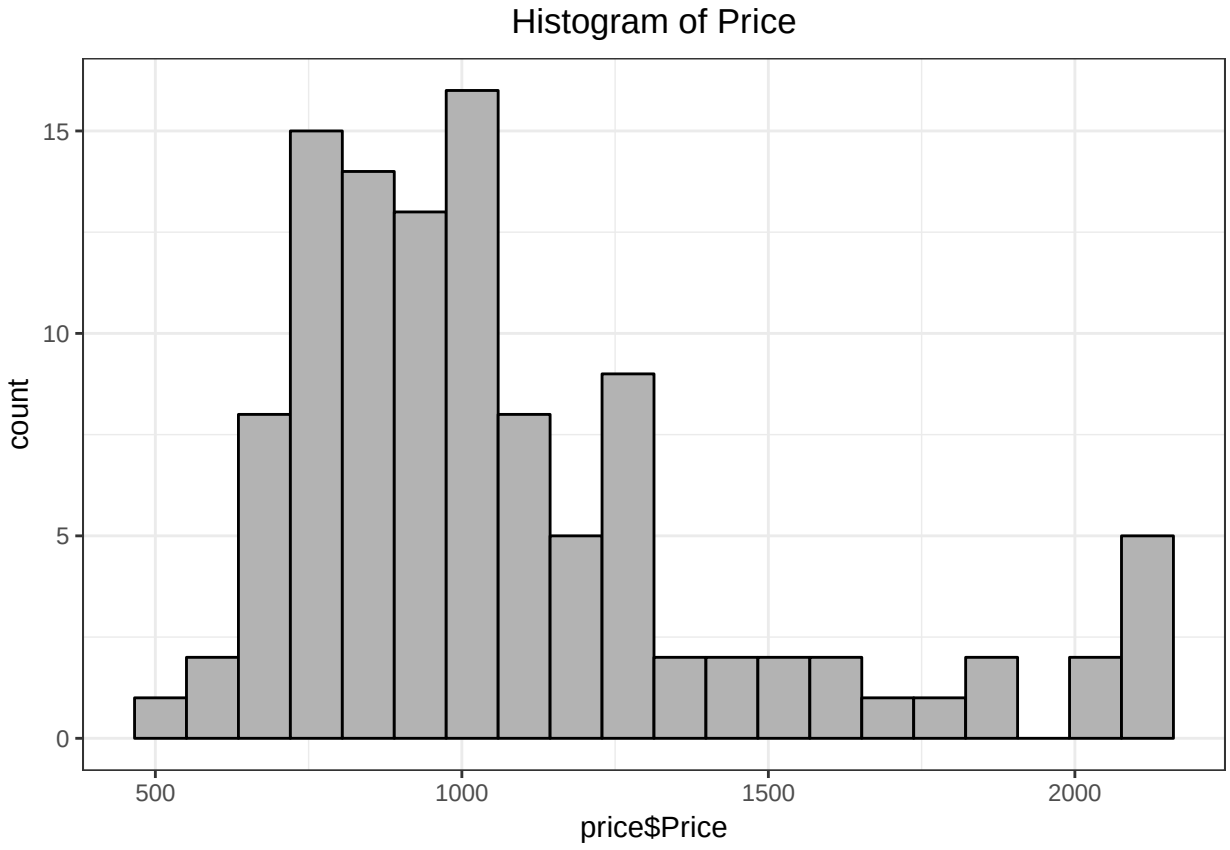
Se can see that the power is 1.

Assigment 2: Bootstrap, jackknife and confidence intervals

1

Here we will plot histogram of `Price` and compute the mean of the `Price`.

```
plot2_1 <- ggplot(mapping = aes(x = price$Price)) +
  geom_histogram(col = "grey0", fill = "grey70", bins = 20) +
  theme_bw() +
  ggtitle("Histogram of Price") +
  theme(plot.title = element_text(hjust = 0.5))
plot2_1
```




```
men <- mean(price$Price)
men
```

```
## [1] 1080
```

The distibution reminds a bit about a gamma distribution, but in the same time it is skewed.

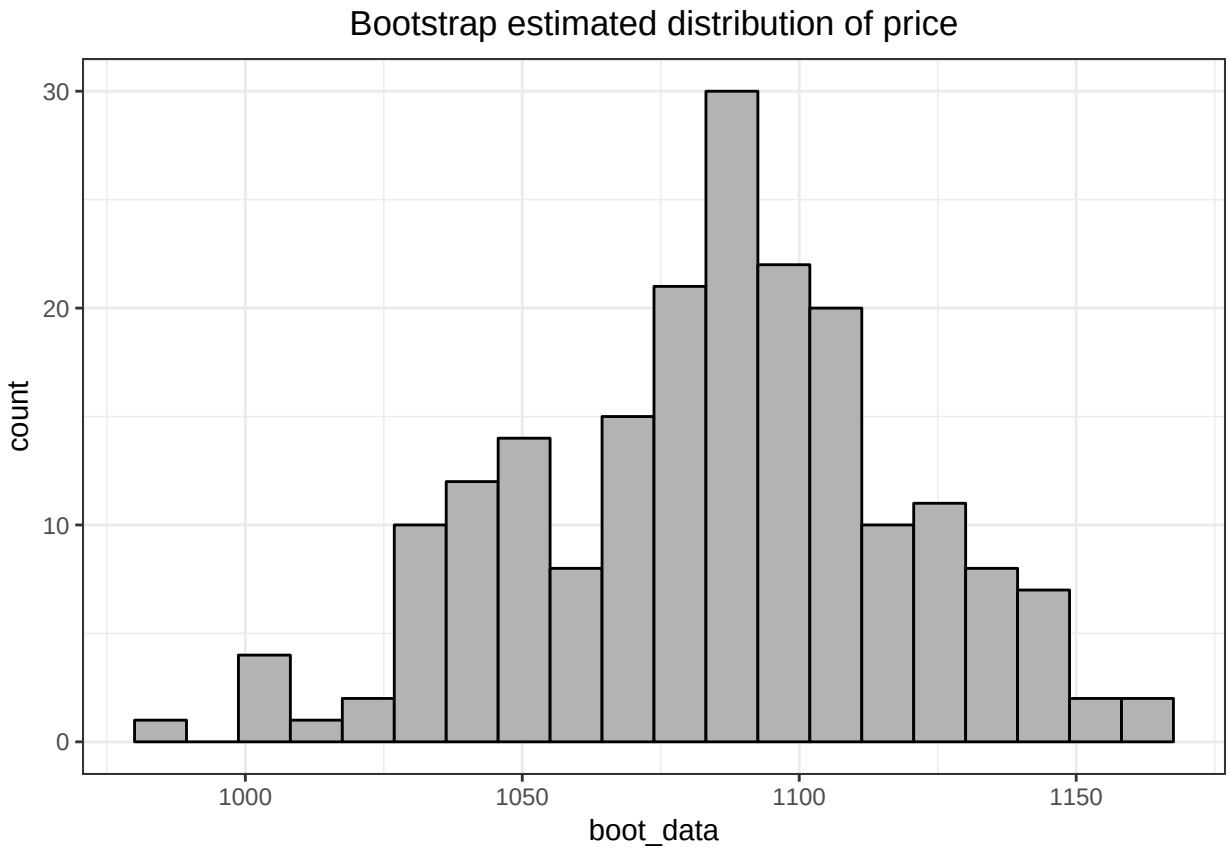
2

In this task we will compute a 95% confidence interval using bootstrap percentile, bootstrap BCa, and first–order normal approximation. We will also determine the bootstrap bias–correction and the variance of the mean price.

```
set.seed(12345)
#Make a function for the bootstrap
stat1 <- function(data, vn){
  data <- as.data.frame(data[vn,])

  return(mean(data$Price))
}
B <- 200
boot_2 <- boot(price, stat1, R = B) #bootstrap
boot_data <- boot_2$t               #Ectract the means from the bootstrap
```

The histogram of the bootstrap:

```
ggplot(mapping = aes(x=boot_data))+
  geom_histogram(col="grey0",fill="grey70",bins=20)+
  theme_bw()+
  ggtitle("Bootstrap estimated distribution of price")+
  theme(plot.title = element_text(hjust = 0.5))
```



The bias correction and boot variance:

```
#Calculate the bias of the mean
bias_estimator<-mean(price$Price)-(2*mean(price$Price)-sum(boot_data)/B)
bias_estimator
```

```
## [1] 3.83
```

```
#The mean bias correction
mean(price$Price)-bias_estimator
```

```
## [1] 1077
```

```
#Same formula as above (formula from slide 18 L5)
2*mean(price$Price)-(sum(boot_data)/B)
```

```
## [1] 1077
```

```
# Extract the bootstrap vaiance
boot_variance<-c(var(boot_data)) #boot variance
```

```
boot_variance
```

```
## [1] 1178
```

```
#Here we can compare with the results above
boot_2
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = price, statistic = stat1, R = B)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*     1080    3.83       34.32
```

The intervals:

```
perc<-boot.ci(boot_2,type = "perc") #bootstrap percentile
bca<-boot.ci(boot_2,type = "bca")   #bca (adjusted bootstrap percentile)
norm<-boot.ci(boot_2,type = "norm") #Normal approx

boot.ci(boot_2,type=c("perc","bca","norm"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 200 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_2, type = c("perc", "bca", "norm"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (1009, 1144 )   (1007, 1146 )   (1002, 1142 )
## Calculations and Intervals on Original Scale
## Some percentile intervals may be unstable
## Some BCa intervals may be unstable
```

3

In this task we will estimate the variance of the mean price using the jackknife and compare it with the bootstrap estimate.

```
store_2 <- c()
dat_tmp <- price$Price
n <- length(dat_tmp)
B <- n
```

```
for(i in 1:B){
  store_2[i] <- mean(dat_tmp[-i]) #exclude i in each itteration
}

J_T <- mean(store_2)
T_D <- mean(price$Price)

T_star <- n*T_D - (n - 1)*store_2

var_jack <- (1/(n*(n-1))) * sum((T_star- J_T)^2) #variance
```

The mean and variance for jackknife and bootstrap:

```
#Var
paste("Jack variance:", var_jack)      #Jack variance
```

```
## [1] "Jack variance: 1320.91104405187"
```

```
paste("Boot variance:", boot_variance) #Boot variance
```

```
## [1] "Boot variance: 1177.69574782798"
```

```
# Mean
paste("Jack mean",mean(store_2))   #Jack mean
```

```
## [1] "Jack mean 1080.47272727273"
```

```
paste("Boot mean", mean(boot_data)) #Boot mean
```

```
## [1] "Boot mean 1084.30313636364"
```

We can see that variance forthe bootstrap is smaller then the jackknife. The means are quite simmilar, but the jackknife has a lower mean then the bootstrap. The jackknife mean are closer to the mean from the sample.

4

In this task we will compare the confidence intervals obtained.

```
under <- c()
over <- c()
length_int <- c()
middle_of_intervall <- c()
namn <- c("perc", "bca", "norm")

#perc
under[1] <- perc$percent[, 4]; over[1] <- perc$percent[, 5]
length_int[1] <- over[1] - under[1]
middle_of_intervall[1] <- mean(c(over[1], under[1]))
```

```r
#bca
under[2] <- bca$bca[, 4]; over[2] <- bca$bca[, 5]
length_int[2] <- over[2] - under[2]
middle_of_intervall[2] <- mean(c(over[2], under[2]))


#norm
under[3] <- norm$normal[, 2]; over[3] <- norm$normal[, 3]
length_int[3] <- over[3] - under[3]
middle_of_intervall[3] <- mean(c(over[3], under[3]))

# print the values
knitr::kable(
  data.frame(
  "Type" = namn,
  "Under" = under,
  "Middle" = middle_of_intervall,
  "Over" = over,
  "Length of intervall" = length_int
  ))
```

Type	Under	Middle	Over	Length.of.intervall
perc	1007	1077	1146	138.6
bca	1002	1072	1142	139.6
norm	1009	1077	1144	134.5

We can see from this output that the interval that is shortest are the first–order normal approximation and has the middle point closest to the sample mean: 1080.473. Also from the output we can see that the broader the interval, the more the middle of the interval is off from the true mean.

Appendix

R-code

```r
# Library ####
library(boot)
library(tidyverse)
options(digits = 4)

# Library ####
library(boot)
library(tidyverse)
options(digits = 4)

lottery <- read.csv2("lottery.csv")
ggplot(lottery,aes(y=Draft_No,x=Day_of_year))+
  geom_point()+
  theme_bw()+
  ggtitle("Is lottery random?")+
  theme(plot.title = element_text(hjust = 0.5))
```

```
ggplot(lottery,aes(y=Draft_No,x=Day_of_year))+
  geom_point()+
  theme_bw()+
  ggtitle("Draft numbers vs day of year, with loess model")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_smooth(method = "loess")
my_bot_fun <- function(data, index) {
  new_data <- data[index, c(4, 5)]
  modelen <- loess(Draft_No ~ Day_of_year, new_data)

  Xa_pos <- which.min(new_data$Draft_No) #Which min on Y
  Xb_pos <- which.max(new_data$Draft_No) #Which max on Y

  Xa <- new_data$Day_of_year[Xa_pos] #The X value that gives the min Y
  Xb <- new_data$Day_of_year[Xb_pos] #The X value that gives the max Y

  Y_Xa <- modelen$fitted[Xa_pos]
  Y_Xb <- modelen$fitted[Xb_pos]

  T_value <- (Y_Xb - Y_Xa) / (Xb - Xa) #The test statistic

  return(T_value)
}
B=2000
my_boot_strap <- boot(data = lottery, statistic = my_bot_fun, R = B) #Do the bootsprat

T_x<-0
p_value<-sum(my_boot_strap$t>=T_x)/B #get the p-value. Now its right

#1-sum(my_boot_strap$t<T_x)/B #We dident mix up p-value and power
                              #We mixed it up with this formula which gives the same results

text_nr1<-paste("Number of values \n thats over 0 is",sum(my_boot_strap$t>T_x))
text_nr2<-paste("P-value=",sum(my_boot_strap$t>T_x),"/",B,"=\n",sum(my_boot_strap$t>T_x)/B)

ggplot(mapping = aes(x=my_boot_strap$t))+
  geom_histogram(col="grey0",fill="grey70",bins=20)+
  theme_bw()+
  ggtitle("Estimated distribution of the test-statistic with bootstrap")+
  theme(plot.title = element_text(hjust = 0.5))+
  geom_vline(xintercept = 0,linetype="dashed",size=1)+
  annotate("text",0.3,350,label=text_nr1)+
  annotate("text",0.3,250,label=text_nr2)+
  geom_vline(xintercept = T_x,linetype="dashed",size=1)+
  geom_line(aes(x=c((T_x+0.7),T_x),y=40),
            arrow = arrow(length=unit(0.30,"cm"),
                          ends="last", type = "closed"))+
  annotate("text",T_x+0.35,60,label="Count")
tst <- function(data, B){

  tmp_data <- data #make a temporary data set
  samp_val <- c()  #We will save the test statistic in this variable
```

```r
# Get the distribution of T when X is "garbage" ####
for(i in 1:B){
  ind <- sample(1:nrow(data), nrow(data))   #Suffle around the X
  tmp_data$X <- data$X[ind]                 #Suffle around the X

  modl <- loess(Y ~ X, tmp_data)            #Make a model when X is "garbage"

  x_b <- data$X[which.max(data$Y)]          #Make the test statistic
  x_a <- data$X[which.min(data$Y)]          #Make the test statistic

  y_x_b <- predict(modl, x_b)               #Make the test statistic
  y_x_a <- predict(modl, x_a)               #Make the test statistic

  samp_val[i] <- (y_x_b - y_x_a)/(x_b- x_a) #Save the test statistic
}

# Calc test statistc
all_data_model <- loess(Y ~ X, data)     #Model when X is not "garbage"
x_b <- data$X[which.max(data$Y)]         #Make the test statistic
x_a <- data$X[which.min(data$Y)]         #Make the test statistic
y_x_b <- predict(all_data_model, x_b)    #Make the test statistic
y_x_a <- predict(all_data_model, x_a)    #Make the test statistic

T_test_statistic <- (y_x_b - y_x_a)/(x_b- x_a) #Save our obtain test-statistic

#This formula works if the disturbution is symmetric
#Like for example in t-distribution and normal-distribution
#So it dont work in our case
#p_value <- (sum(samp_val < T_test_statistic) / B) * 2 #Get the p-value.
                                                      #We multiply it with 2
                                                      #because its a two-sided hypothesis.

#The new way to calculate the p-value
p_value <- (sum(abs(samp_val) > abs(T_test_statistic)) / B)

############ Down from here its just to make a histogram of the data. ########
nr1_text<-paste("Number of values \n thats over |",
                round(T_test_statistic,2),"| is ",
                sum(abs(samp_val)>abs(T_test_statistic)),sep="")
nr2_text<-paste("P-value= \n (",
                sum(abs(samp_val)>abs(T_test_statistic)),"/",B,")=",
                p_value,sep="")

ggplot_return<-ggplot()+
  geom_histogram(aes(x=samp_val),col="grey0",fill="grey70",bins=20)+
  #geom_vline(xintercept = T_test_statistic,linetype="dashed",size=1,y=0)+
  geom_segment(aes(x=T_test_statistic,y=0,xend=T_test_statistic,yend=150),linetype=8,size=1)+
  geom_segment(aes(x=-T_test_statistic,y=0,xend=-T_test_statistic,yend=150),linetype=8,size=1)+
  labs(x="T distribution",y="",title="Permutation test")+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))+
  annotate("text",T_test_statistic ,-10,label=paste("T =",round(T_test_statistic,2)))+
  annotate("text",-T_test_statistic,-10,label=paste("T =",-round(T_test_statistic,2)))+
```

```
    annotate("text",0.30,300,label=nr1_text)+
    annotate("text",0.30,200,label=nr2_text)+
    geom_line(aes(x=c(T_test_statistic-0.2,T_test_statistic),y=40),
            arrow = arrow(length=unit(0.30,"cm"),
                          ends="first", type = "closed"))+
    geom_line(aes(x=c(-T_test_statistic+0.2,-T_test_statistic),y=40),
            arrow = arrow(length=unit(0.30,"cm"),
                          ends="last", type = "closed"))+
    annotate("text",T_test_statistic-0.1,60,label="Count")+
    annotate("text",-T_test_statistic+0.1,60,label="Count")
  ########################

  svar <- list(p_value, ggplot_return, samp_val, T_test_statistic) #Save a list that the

  return(svar)   #function return
}
lottery2 <- data.frame(Y = lottery$Draft_No, X = lottery$Day_of_year) #Make a data.frame
Permutation_tests <- tst(lottery2, 2000)
Permutation_tests_1.4<-Permutation_tests  #save for assigment 1.5 power test

Permutation_tests[[2]] #Histogram

tst2 <- function(data, B){

  tmp_data <- data
  samp_val <- c()

  # Get the distribution of T ####
  for(i in 1:B){
    ind <- sample(1:nrow(data), nrow(data))
    tmp_data$X <- data$X[ind]

    modl <- loess(Y ~ X, tmp_data)

    x_b <- data$X[which.max(data$Y)]
    x_a <- data$X[which.min(data$Y)]

    y_x_b <- predict(modl, x_b)
    y_x_a <- predict(modl, x_a)

    samp_val[i] <- (y_x_b - y_x_a)/(x_b- x_a)
  }

  # Calc test statistc
  all_data_model <- loess(Y ~ X, data)     #model with true data
  x_b <- data$X[which.max(data$Y)]         #X value there Y is biggest
  x_a <- data$X[which.min(data$Y)]         #X value there Y is samllest
  y_x_b <- predict(all_data_model, x_b)    #pred Y on biggest X
  y_x_a <- predict(all_data_model, x_a)    #pred Y on samllest X

  T_test_statistic <- (y_x_b - y_x_a)/(x_b- x_a)
```

```
  p_value <- (sum(samp_val > T_test_statistic) / B) * 2


  nr1_text<-paste("Number of values \n thats over",
                  round(T_test_statistic,2),"is",
                  sum(samp_val>T_test_statistic))
  nr2_text<-paste("P-value= \n",
                  "2*(",sum(samp_val>T_test_statistic),
                  "/",B,")=",
                  p_value,sep="")

  ggplot_return<-ggplot()+
    geom_histogram(aes(x=samp_val),col="grey0",fill="grey70",bins=30)+
    geom_segment(aes(x=round(T_test_statistic,2),y=0,
                     xend=round(T_test_statistic,2),yend=50),
                 linetype=8,size=1) +
    labs(x="T distribution",y="",title="Permutation test")+
    theme_bw()+
    theme(plot.title = element_text(hjust = 0.5))+
    annotate("text",round(T_test_statistic,2),-5,label=paste("T =",round(T_test_statistic,2)))+
    annotate("text",0.06,45,label=nr1_text)+
    annotate("text",0.06,25,label=nr2_text)


  svar <- list(p_value, ggplot_return,T_test_statistic)
  return(svar)
}
X <- lottery$Day_of_year               #Or X values
beta <- rnorm(1, 183, 10)              #Beta value
alpha <- 0.1                           #Alpha value
alpha_beta_value <- alpha * X + beta   #inner, inner values



Y_x <- NULL                               #We save our Y(x) in this variable

for (i in seq_along(alpha_beta_value)) {  #For i in inner,inner values
  a<-min(alpha_beta_value[i],366)         #Get the inner values
  Y_x[i] <- max(0, a)                     #Get the values=Y
}


ggplot(mapping = aes(x=X,y=Y_x))+         #Plot the obtain data
  geom_point()+
  theme_bw()+
  geom_smooth(method = "loess")
# Make the Permutation tests ####
data <- data.frame(Y = Y_x, X = X)   #set the data in a data.frame
Permutation_tests <- tst2(data, 200) #Do permutation tests, look in appendix for
                                     #code for tst2 (simular to tst)

Permutation_tests[[2]] #histogram
# Fix parameters
```

```r
diffrent_alpha <- seq(0.2, 10, 0.1) #diffrent alpha
X <- lottery$Day_of_year            #set our X values
Y_x2 <- list()                      #Save our Y(x) with diffrent alphas

# Calc the Y-values
for (j in seq_along(diffrent_alpha)) { #Loop over all alphas (its 99 diffrent alphas)
  alpha <- diffrent_alpha[j]           #Set the current alpha
  beta <- rnorm(1, 183, 10)            #Sample a beta

  alpha_beta_value <- alpha * X + beta #inner, inner values

  temp_y_x <- NULL                     #tempurary save Y(x) given a alpha in this varaiable

  for (i in seq_along(alpha_beta_value)) { #Loop over all inner,inner values
    a <- min(alpha_beta_value[i], 366)     #Get the inner values
    temp_y_x[i] <- max(0, a)               #Get Y(x) values
  }

  Y_x2[[j]] <- temp_y_x     #Save the current Y(x) in a list
}

diffrent_alpha_factor<-as.factor(diffrent_alpha)
ggplot(mapping = aes(x=X))+
  geom_point(aes(y=Y_x2[[1]],col=diffrent_alpha_factor[1]),col="red")+
  geom_point(aes(y=Y_x2[[40]],col=diffrent_alpha_factor[40]))+
  geom_point(aes(y=Y_x2[[99]],col=diffrent_alpha_factor[99]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[1]],
                            col=diffrent_alpha_factor[1]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[40]],
                            col=diffrent_alpha_factor[40]))+
  geom_smooth(method = "loess",se=FALSE,
              mapping = aes(y=Y_x2[[99]],
                            col=diffrent_alpha_factor[99]))+
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))+
  labs(title="X vs Y with diffrent alphas and loess model")


p_values_ass_1.5 <- c() #variable to save p-values
test_statistic_1.5<-c() #variable to save T-statistic

for (i in 1:length(Y_x2)) {                  #Loop for every Y(x) (its 99 of them)
  data <- data.frame(Y = Y_x2[[i]], X = X)   #Put the current values in a data.frame
  perm_test<-tst2(data, 200)                 #Make a permutation test. (It return a list)

  p_values_ass_1.5[i]  <- perm_test[[1]]     #Save the p-value
  test_statistic_1.5[i] <- perm_test[[3]]    #Save the T-statistic
}
p_values_ass_1.5

test_statistic_dep_alpha<-ggplot(mapping = aes(x=diffrent_alpha,y=test_statistic_1.5))+
```

```
  geom_point()+
  theme_bw()+
  labs(x="Alpha",y="T-statistic",title="T-statistic depending on alpha")+
  theme(plot.title = element_text(hjust = 0.5))
test_statistic_dep_alpha

#Significns level
alpha<-0.1

#numer of type 2 errors
type2_error<-sum((p_values_ass_1.5>alpha))

#proportion of type
type2_error<-type2_error/length(p_values_ass_1.5)

# The power
1-type2_error
price <- read.csv2("prices1.csv")
plot2_1 <- ggplot(mapping = aes(x = price$Price)) +
  geom_histogram(col = "grey0", fill = "grey70", bins = 20) +
  theme_bw() +
  ggtitle("Histogram of Price") +
  theme(plot.title = element_text(hjust = 0.5))
plot2_1

men <- mean(price$Price)
men

#Make a function for the bootstrap
stat1 <- function(data, vn){
  data <- as.data.frame(data[vn,])

  return(mean(data$Price))
}
B <- 200
boot_2 <- boot(price, stat1, R = B) #bootstrap
boot_data <- boot_2$t               #Ectract the means from the bootstrap
ggplot(mapping = aes(x=boot_data))+
  geom_histogram(col="grey0",fill="grey70",bins=20)+
  theme_bw()+
  ggtitle("Bootstrap estimated distribution of price")+
  theme(plot.title = element_text(hjust = 0.5))
bias_estimator<-mean(price$Price)-(2*mean(price$Price)-sum(boot_data)/B) #Calculate the bias estimator
boot_variance<-c(var(boot_data)) #boot variance

bias_estimator

boot_variance
perc<-boot.ci(boot_2,type = "perc") #bootstrap percentile
bca<-boot.ci(boot_2,type = "bca")   #bca (adjusted bootstrap percentile)
norm<-boot.ci(boot_2,type = "norm") #Normal approx

boot.ci(boot_2,type=c("perc","bca","norm"))
```

```r
store_2 <- c()
dat_tmp <- price$Price
n <- length(dat_tmp)
B <- n

for(i in 1:B){
  store_2[i] <- mean(dat_tmp[-i]) #exclude i in each itteration
}

J_T <- mean(store_2)
T_D <- mean(price$Price)

T_star <- n*T_D - (n - 1)*store_2

var_jack <- (1/(n*(n-1))) * sum((T_star- J_T)^2) #variance
#Var
paste("Jack variance:", var_jack)      #Jack variance
paste("Boot variance:", boot_variance) #Boot variance

# Mean
paste("Jack mean",mean(store_2))   #Jack mean
paste("Boot mean", mean(boot_data)) #Boot mean
under <- c()
over <- c()
length_int <- c()
middle_of_intervall <- c()
namn <- c("perc", "bca", "norm")

#perc
under[1] <- perc$percent[, 4]; over[1] <- perc$percent[, 5]
length_int[1] <- over[1] - under[1]
middle_of_intervall[1] <- mean(c(over[1], under[1]))

#bca
under[2] <- bca$bca[, 4]; over[2] <- bca$bca[, 5]
length_int[2] <- over[2] - under[2]
middle_of_intervall[2] <- mean(c(over[2], under[2]))


#norm
under[3] <- norm$normal[, 2]; over[3] <- norm$normal[, 3]
length_int[3] <- over[3] - under[3]
middle_of_intervall[3] <- mean(c(over[3], under[3]))

# print the values
knitr::kable(
  data.frame(
  "Type" = namn,
  "Under" = under,
  "Middle" = middle_of_intervall,
  "Over" = over,
  "Length of intervall" = length_int
  ))
```

```
##
## # code=readLines(knitr::purl('Lab 5 albin eric.Rmd', documentation = 0)), eval = FALSE
## # knitr::purl('Lab 5 albin eric.Rmd', documentation = 0)
##
```