Computational Statistics (732A90) Lab5

Anubhav Dikshit(anudi287) and Thijs Quast(thiqu264)

12 Feburary 2019

Contents

Question 1: Hypothesis testing **2**

1. Make a scatterplot of Y(draft_no) versus X(day_of_year) and conclude whether the lottery looks random. 2
2. Compute an estimate Y(hat) of the expected response as a function of X by using a loess smoother (use loess()), put the curve Y(hat) versus X in the previous graph and state again whether the lottery looks random. 2
3. To check whether the lottery is random, it is reasonable to use test statistics : 4
4. Implement a *function* depending on data and B that tests the hypothesis 6
5. Make a crude estimate of the power of the test constructed in Step4 : 8

Question 2: Bootstrap, jackknife and confidence intervals **10**

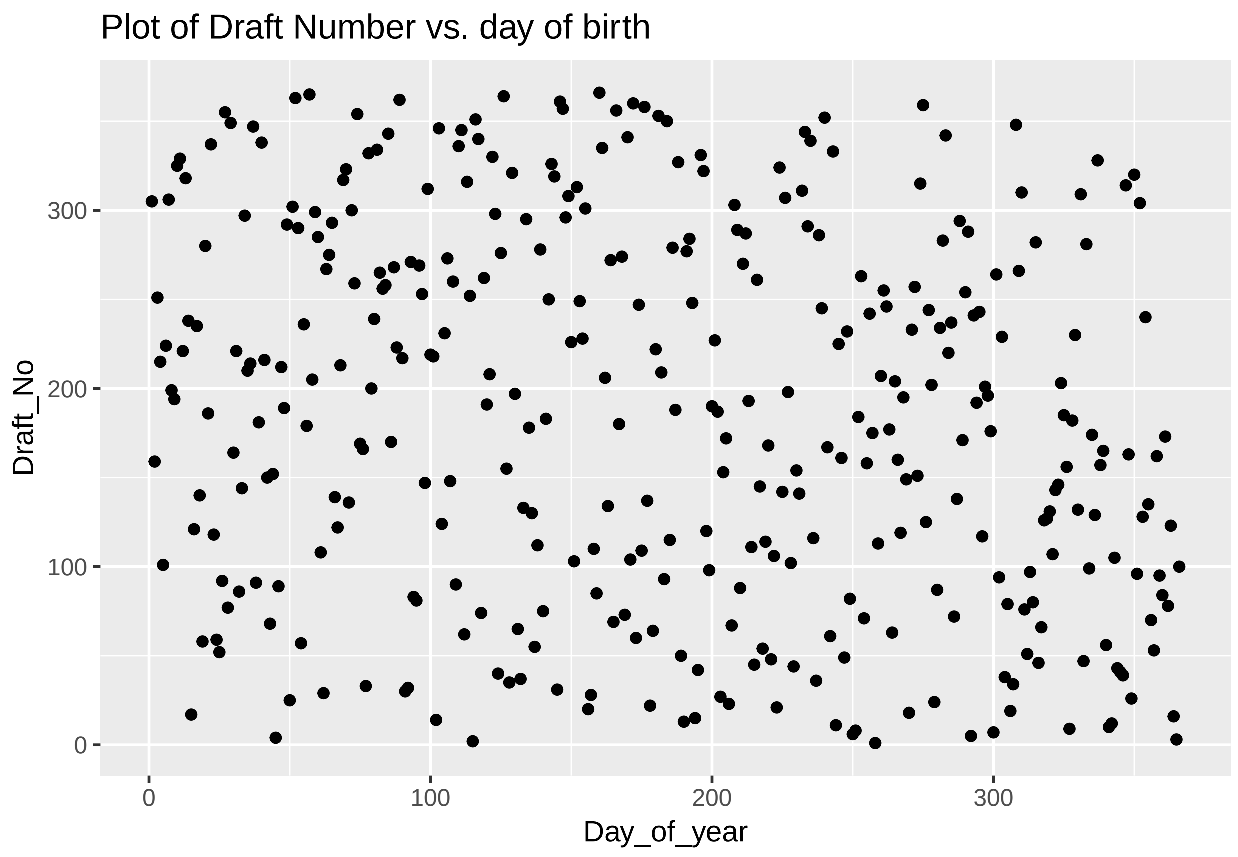
1. Plot the histogram of Price. Does it remind any conventional distribution? Compute the mean price. 10
2. Estimate the distribution of the mean price of the house using bootstrap. Determine the bootstrap bias-correction and the variance of the mean price. Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first-order normal approximation 11
3. Estimate the variance of the mean price using the jackknife and compare it with the bootstrap estimate 14
4. Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals. 15

Appendix **16**

Question 1: Hypothesis testing

1. Make a scatterplot of Y(draft_no) versus X(day_of_year) and conclude whether the lottery looks random.

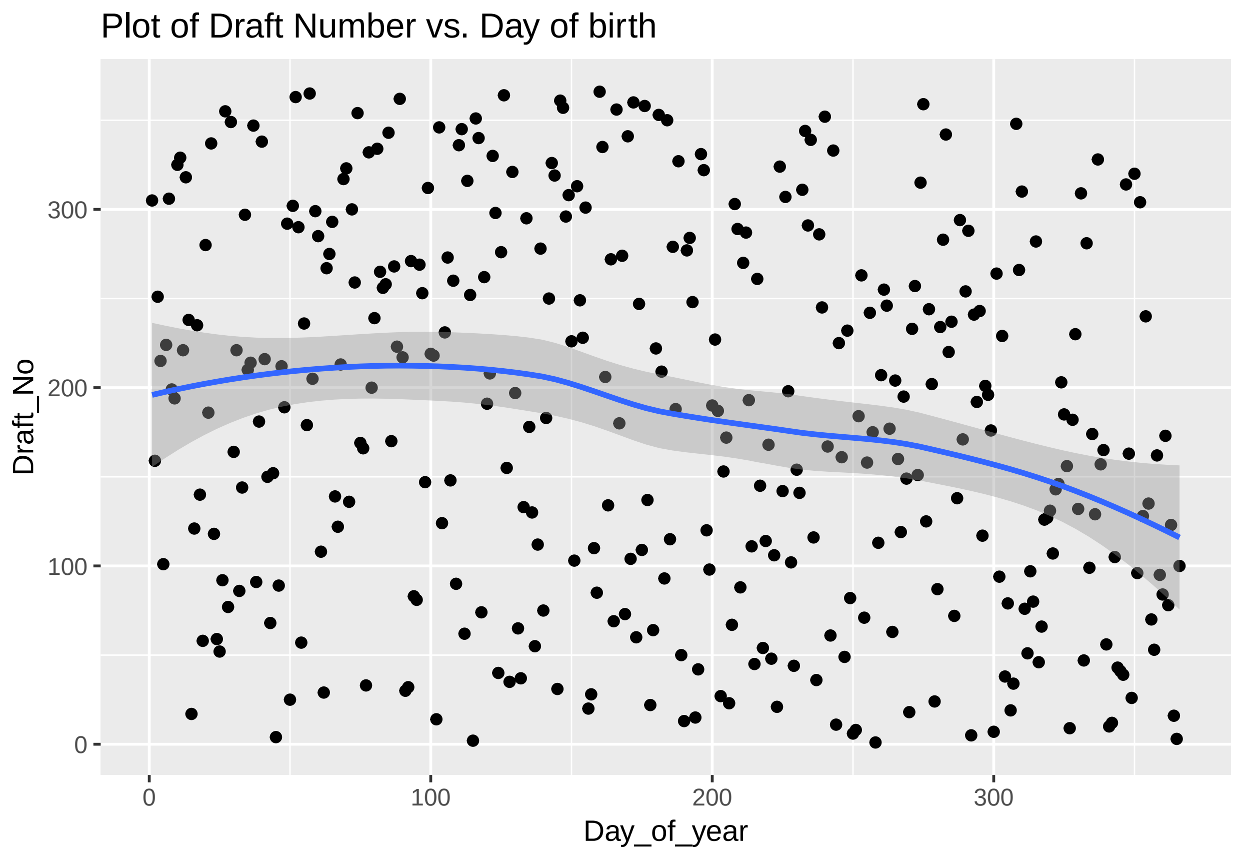
```
data <- read.csv("lottery.csv", sep=";")

ggplot(data, aes(x=Day_of_year, y = Draft_No)) + geom_point() +
  ggtitle("Plot of Draft Number vs. day of birth")
```



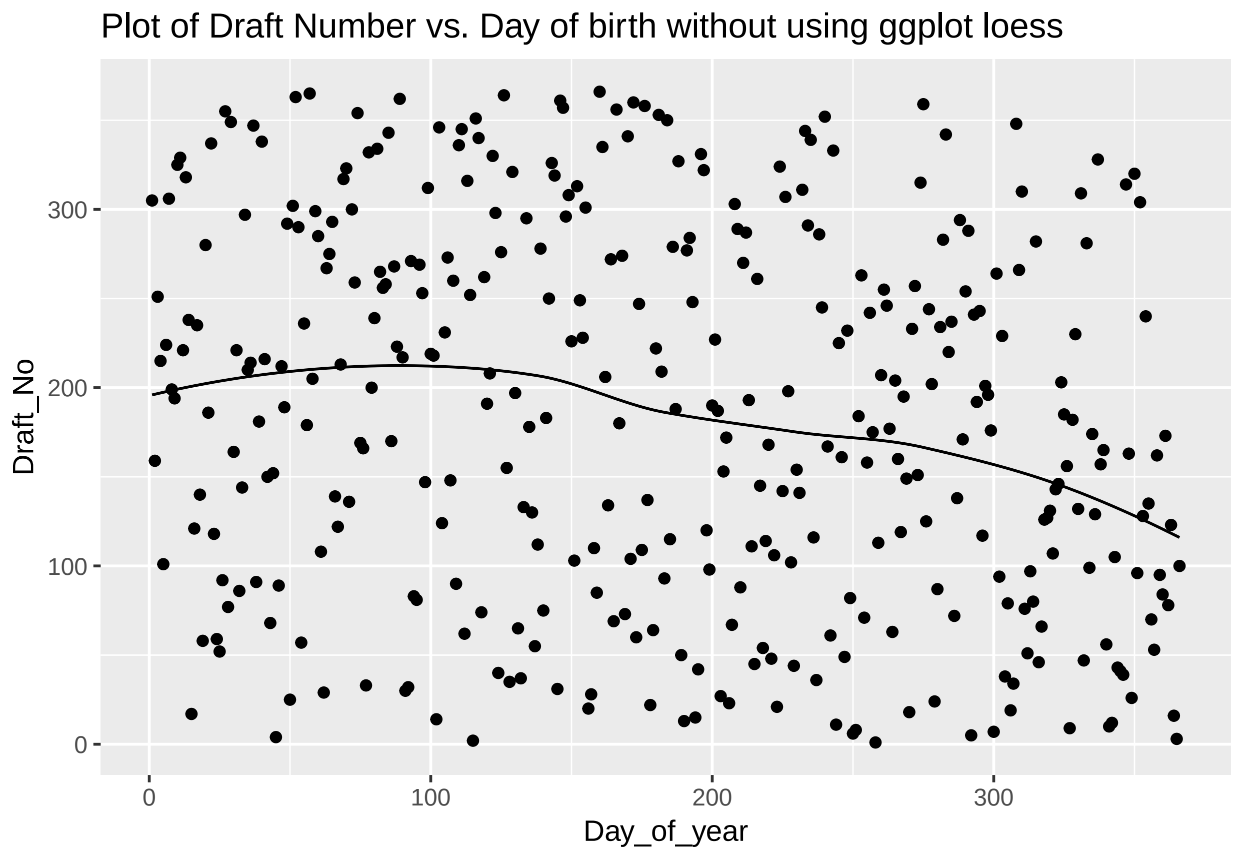
Analysis: The plot seems random and its very difficult to judge any sort of trend

2. Compute an estimate Y(hat) of the expected response as a function of X by using a loess smoother (use loess()), put the curve Y(hat) versus X in the previous graph and state again whether the lottery looks random.

```
ggplot(data, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_smooth(method = loess) +
  ggtitle("Plot of Draft Number vs. Day of birth")
```



```r
model <- loess(Draft_No ~ Day_of_year, data)
data$Y_hat <- predict(model, data)

ggplot(data, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_line(aes(y = Y_hat)) +
  ggtitle("Plot of Draft Number vs. Day of birth without using ggplot loess")
```



Analysis: One can see that the overall trend is downward, the implies the more people were born in the first half of the year than the later half, however the distribution still seems random.

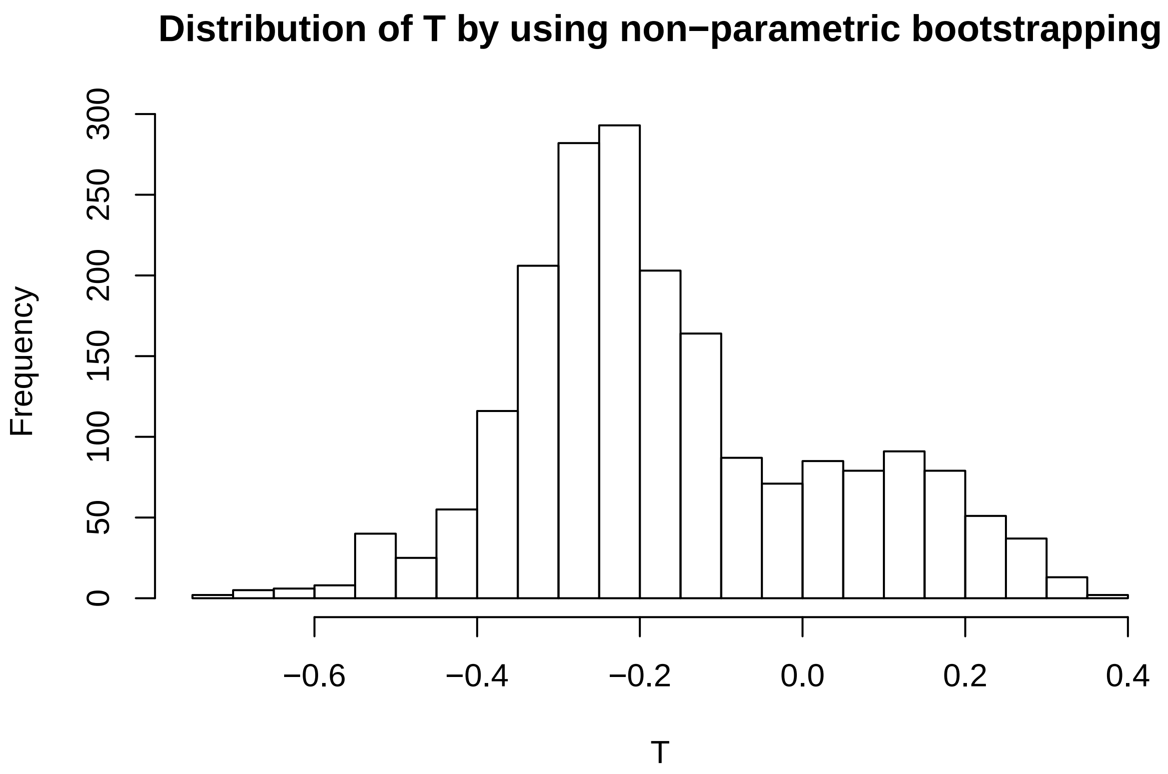
3. To check whether the lottery is random, it is reasonable to use test statistics :

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

If this value is significantly greater than zero, then there should be a trend in the data and lottry is not random. Estimate the distribution of T by using a **non-parametric bootstrap** with B=2000 and comment whether the lottery is random or not. What is the p-value of the test?

Since we are doing non-parametric bootstraping, we assume the distribution follows True population. So we simply need to choose the maximum Y and following index X and minimum Y and following index X.

```r
set.seed(12345)

X <- data$Day_of_year
Y <- data$Draft_No

T <- numeric()

for (i in 1:2000) {
        boot <-sample(length(X), replace=TRUE)
        data1 <- cbind(X,boot)
        X_a <- data1[which.min(data1[,2])]
        X_b <- data1[which.max(data1[,2])]

        model2 <- loess(Y~X, data=as.data.frame(data1), method="loess")

        fitted_X_a <- model2$fitted[X_a]
        fitted_X_b <- model2$fitted[X_b]

        test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
        T[i] <- test
        }


pval <- length(which(T>=0))


hist(T, breaks=30,
     main="Distribution of T by using non-parametric bootstrapping")
```



```
print("Estimated p-value:")
```

```
## [1] "Estimated p-value:"
```

```
print(pval/2000)
```

```
## [1] 0.2185
```

Obtained plot above is the estimated distribution of T. The distribution seems to be centered around -0.2 in normal distribution shape but with heavier right tail. The value is bigger than 0 at most of the T as well, so we can say that the lottery is not random. The obtained p-value is 0.2185.

4. Implement a *function* depending on data and B that tests the hypothesis

$$H_0 : Lottery\ is\ random \quad VS \quad H_1 : Lottery\ is\ not\ random$$

by using a permutation test with statistics T. The function is to return the p-value of this test. Test this function on our data with B=2000.

First, we need to compute the test statistics from observed(original) data

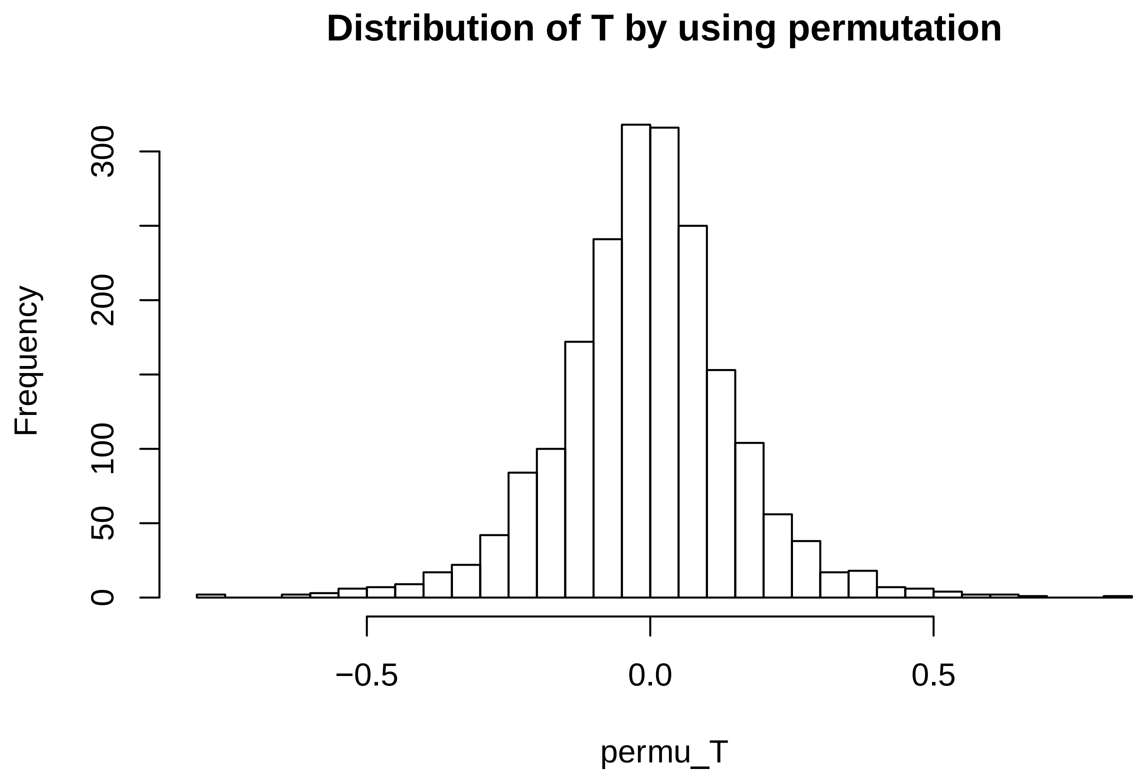
```r
data <- cbind(X,Y)
X_a <- data[which.min(data[,2])]
X_b <- data[which.max(data[,2])]

model <- loess(Y~X, data=as.data.frame(data), method="loess")

fitted_X_a <- model$fitted[X_a]
fitted_X_b <- model$fitted[X_b]

test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
print(test)
```

```
## [1] -0.2671794
```

Then, we compute the test statistics from permuted data(B=2000)

```r
#To do permutation - we have to compare two groups (observed data group vs permutated group).
#for permutated group - We may permute labels (from the lecture note
#So basically, we are comparing between two groups - Before permutation vs After permutation.
set.seed(12345)
permu_T <- numeric()

for (i in 1:2000){
        permu_Y <- sample(1:length(Y), length(Y), replace=FALSE)
        permu_data <- cbind(X,permu_Y)

        #and do the same thing
        X_a <- permu_data[which.min(permu_data[,2])]
        X_b <- permu_data[which.max(permu_data[,2])]

        permu_model <- loess(permu_Y~X, data=as.data.frame(permu_data), method="loess")

        fitted_X_a <- permu_model$fitted[X_a]
        fitted_X_b <- permu_model$fitted[X_b]

        permu_test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
        permu_T[i] <- permu_test

}
```

```
hist(permu_T, breaks=30,
     main="Distribution of T by using permutation")
```



```
#compute p value(two-sided test)
pval2 <- length(which(abs(permu_T)>=abs(test)))

print("Estimated p-value:")
```

```
## [1] "Estimated p-value:"
```

```
print(pval2/2000)
```

```
## [1] 0.086
```

Analysis: The plot above indicates that the lottery is random since observed frequencies are significatly bigger than 0. Tehe obtained p-value is 0.058, so we can't reject the null hypothesis, so we can say that the data is lottery is random. However, the p-value has decreased compared to previous step, which could be more likely to reject the null hypothesis.

The r code following is to put previous steps into function called *permu_test*:

```
#always go with set.seed(12345) for the same result

permu_test <- function(B,X,Y){

        #compute test statistics from original data
        data <- cbind(X,Y)
        X_a <- data[which.min(data[,2])]
        X_b <- data[which.max(data[,2])]

        model <- loess(Y~X, data=as.data.frame(data), method="loess")

        fitted_X_a <- model$fitted[X_a]
        fitted_X_b <- model$fitted[X_b]

        test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)

        #then compute the permuted data
        permu_T <- numeric()

        for (i in 1:B){
                permu_Y <- sample(1:length(Y), length(Y), replace=FALSE)
                permu_data <- cbind(X,permu_Y)

                #and do the same thing
                X_a <- permu_data[which.min(permu_data[,2])]
                X_b <- permu_data[which.max(permu_data[,2])]

                permu_model <- loess(permu_Y~X, data=as.data.frame(permu_data), method="loess")

                fitted_X_a <- permu_model$fitted[X_a]
                fitted_X_b <- permu_model$fitted[X_b]

                permu_test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
                permu_T[i] <- permu_test
        }


        #then compute the estimated p value
        estimated_pval <- length(which(abs(permu_T)>=abs(test)))/B
        return(estimated_pval)

}
```

5. Make a crude estimate of the power of the test constructed in Step4 :

(a) Generate(an obviously non-random) dataset with n=366 observations by using same X as in the original data set and $Y(x) = max(0, min(\alpha x+\beta, 366))$, where $\alpha = 0.1$ and $\beta \sim N(183, sd = 10)$.

```
set.seed(12345)

data <- read.csv("lottery.csv", sep=";")

new_Y <- function(alpha){
        X <- data$Day_of_year
        new_y <- numeric()

        for (i in 1:length(X)){
                beta <- rnorm(1,183,10)
                new_y[i] <- max(0,min(alpha*X[i]+beta,366))
        }

        return(new_y)
}

non_ran_Y <- new_Y(alpha=0.1)
head(non_ran_Y)
```

```
## [1] 188.9553 190.2947 182.2070 178.8650 189.5589 165.4204
```

(b) Plug this data into the permutation test with B=200 and note whether it was rejected.

```
permu_test(B=200, X=data$Day_of_year,Y=non_ran_Y)
```

```
## [1] 0.48
```

The obtained p-value is quite high, so we can't reject Obtained p-value is 0, so the null hypothesis is rejected.

(c) Repeat Steps 5a-5b for $\alpha = 0.2, 0.3, ..., 10$.

```
seq <- seq(0.2,10,by=0.1)
pvals <- numeric()

for (i in 1:length(seq)){
        alpha <- seq[i]
        newY <- new_Y(alpha)

        pvals[i] <- permu_test(B=200, X=data$Day_of_year, Y=newY)
}

signif_p <- which(pvals<0.05)
power <- 1-sum(pvals>0.05)/length(pvals)

list("total_number_of_hypothesis_test"=length(seq),
     "number_of_test_which_reject_H0"=length(signif_p),
     "power"=power)
```

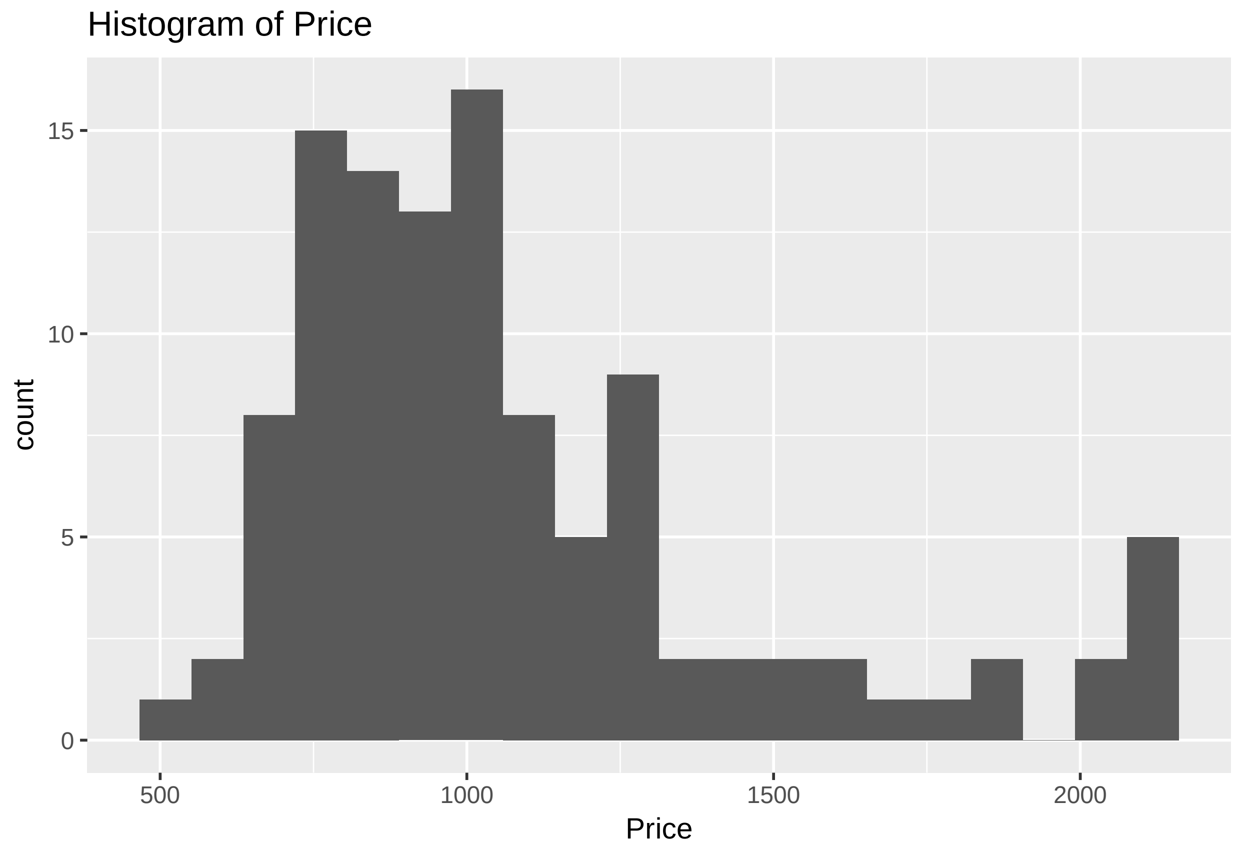
```
## $total_number_of_hypothesis_test
## [1] 99
##
## $number_of_test_which_reject_H0
## [1] 97
##
## $power
## [1] 0.979798
```

Analysis: From the result above, we can see that 98% of the cases reject the null hypothesis, which indicates that the lottery is not random. Since we have generated obviously non-random dataset, we can say that the test statistics performs well enough. The obtaiend power also supports that the quality of statistics is good enough.

Question 2: Bootstrap, jackknife and confidence intervals

1. Plot the histogram of Price. Does it remind any conventional distribution? Compute the mean price.

```r
price_data <- read.csv("prices1.csv", sep=";")

ggplot(data=price_data,aes(Price)) +
  geom_histogram(bins=20) +
  ggtitle("Histogram of Price")
```



```r
cat("The mean price is",mean(price_data$Price))
```

```
## The mean price is 1080.473
```

Analysis: The distribution reminds us of the 'Beta distribution'

2. Estimate the distribution of the mean price of the house using bootstrap. Determine the bootstrap bias-correction and the variance of the mean price. Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first-order normal approximation

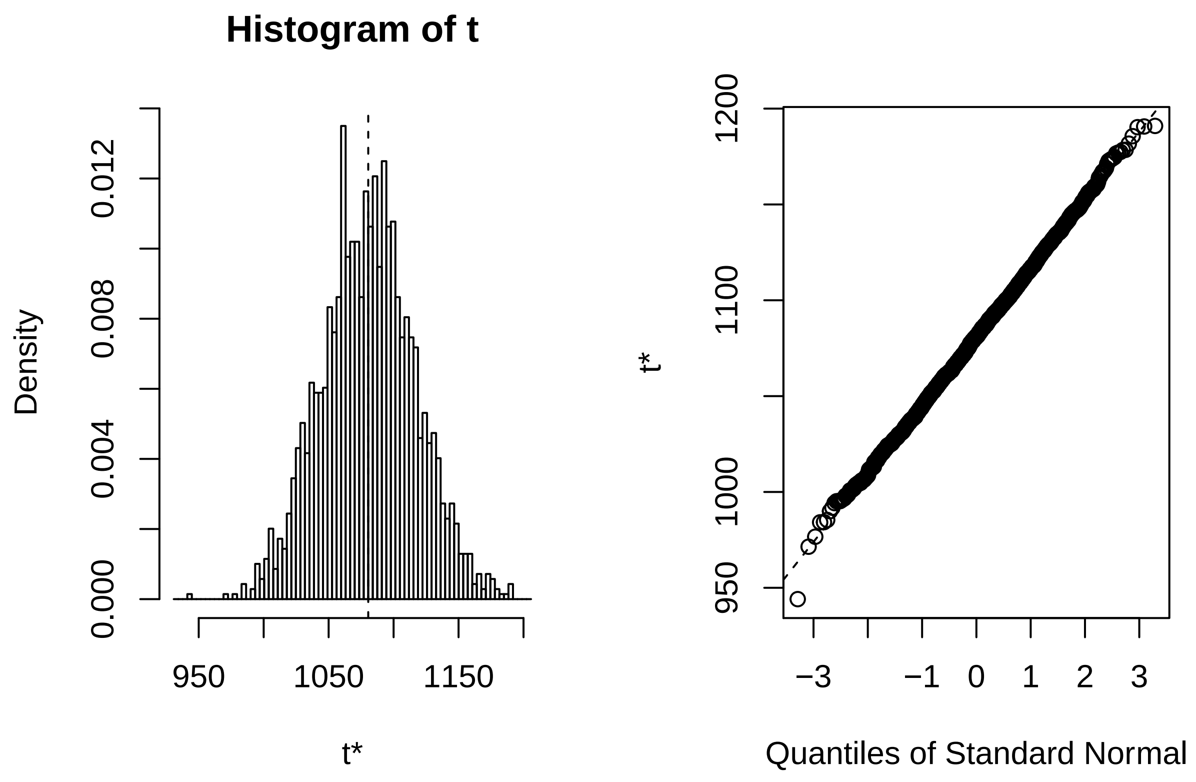
Bias correction

$$T1 = 2.T(D) - \frac{1}{D}\sum_{i=1}^{B} T_i^*$$

```r
# Estimation of mean of Price
stat_mean <- function(data, index){
    data <- data[index,]
    answer <- mean(data$Price)
    return(answer)
}

res <- boot::boot(data=price_data, statistic = stat_mean, R=2000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot::boot(data = price_data, statistic = stat_mean, R = 2000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 1080.473 0.4657227    35.67115
```

```r
plot(res,index = 1)
```



```r
#95% CI for mean using percentile
boot.ci(res, index=1, type=c('perc'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("perc"), index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (1012, 1151 )
## Calculations and Intervals on Original Scale
```

```r
#95% CI for mean using bca
boot.ci(res, index=1, type=c('bca'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("bca"), index = 1)
##
## Intervals :
## Level       BCa
## 95%   (1013, 1154 )
## Calculations and Intervals on Original Scale
```

```r
#95% CI for mean using first order normal
boot.ci(res, index=1, type=c('norm'))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("norm"), index = 1)
##
## Intervals :
## Level      Normal
## 95%   (1010, 1150 )
## Calculations and Intervals on Original Scale
```

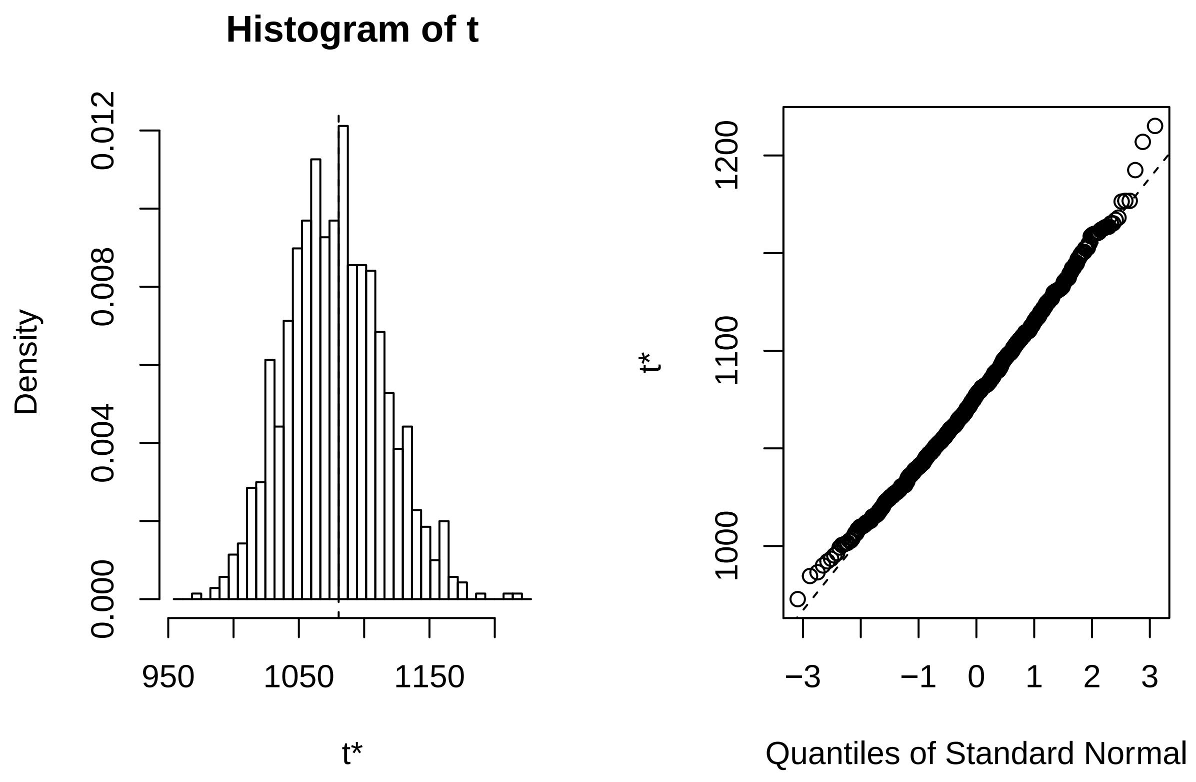
```r
# Bias-correction and Varience of Price
boot.fn <- function(data,index){
        d <- data[index]
        res <- mean(d)
}

boot.result <- boot(data=price_data$Price, statistic=boot.fn, R=1000)
bias_cor <- 2*mean(price_data$Price)-mean(boot.result$t)


list("bias_correction"=bias_cor, "variance_of_the_mean_price"=35.93^2)
```

```
## $bias_correction
## [1] 1083.135
##
## $variance_of_the_mean_price
## [1] 1290.965
```

```r
boot.result
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = price_data$Price, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original     bias    std. error
## t1* 1080.473 -2.661991    36.89815
```

```r
plot(boot.result)
```



```r
boot.ci(boot.result)
```

```
## Warning in boot.ci(boot.result): bootstrap variances needed for studentized
## intervals

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.result)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1011, 1155 )   (1006, 1151 )
##
## Level     Percentile            BCa
## 95%   (1010, 1155 )   (1016, 1163 )
## Calculations and Intervals on Original Scale
```

3. Estimate the variance of the mean price using the jackknife and compare it with the bootstrap estimate

Jacknife(n=B):

$$\widehat{Var[T(\cdot)]} = \frac{1}{n(n-1)} \sum_{i=1}^{n} (T_i^* - J(T))^2$$

where,

$$T_i^* = nT(D) - (n-1)T(D_i^*) \ , \qquad J(T) = \frac{1}{n}\sum_{i=1}^{n} T_i^*$$

When you compute the equation given aboce, you got

$$\frac{n-1}{n}\sum_{i=1}^{n}(T_i^* - J(T))^2$$

Reference:The Jackknife Estimation Method, Avery I. McIntosh (http://people.bu.edu/aimcinto/jackknife.pdf)

```r
result <- numeric()
n <- NROW(price_data)

for (i in 1:n){
        updated_price <- price_data$Price[-i]
        result[i] <- mean(updated_price)
}

var_T <- (n-1)/n*sum((result-mean(result))^2)
mean_T <- mean(result)

cat("The variance from jacknife method is:", var_T)
```

```
## The variance from jacknife method is: 1320.911
```

Analysis: The obtained variance using Jackknife method is 1320.911 while using bootstrapping the obtained value was 1290.965. Considering the fact that Jackknife overestimate variance, the answer seems reasonable.

4. Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.

```r
confidence_interval_jackknife <- c((mean_T - 1.96*var_T), (mean_T + 1.96*var_T))

confidence_interval_jackknife
```

```
## [1] -1508.513  3669.458
```

```r
intervals <- c("(1150-1011)","(1148-1011)","(1149-1013)","(1146-1007)")
length <- c(1150-1011, 1148-1011,1149-1013,1146-1007)
Center_of_interval <- c((1150+1011)/2,(1148+1011)/2,(1149+1013)/2,(1146+1007)/2)

dt <- data.frame(intervals, length, Center_of_interval,row.names = c("Normal", "Basic", "Percentile", "

dt %>% kable(col.names = c("Confidence interval", "Length of interval","Center of interval"))
```

	Confidence interval	Length of interval	Center of interval
Normal	(1150-1011)	139	1080.5
Basic	(1148-1011)	137	1079.5
Percentile	(1149-1013)	136	1081.0
BCa	(1146-1007)	139	1076.5

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
options(scipen=999)
library(dplyr)
library(ggplot2)
library(knitr)
library("boot")
set.seed(12345)
data <- read.csv("lottery.csv", sep=";")

ggplot(data, aes(x=Day_of_year, y = Draft_No)) + geom_point() +
  ggtitle("Plot of Draft Number vs. day of birth")
ggplot(data, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_smooth(method = loess) +
  ggtitle("Plot of Draft Number vs. Day of birth")


model <- loess(Draft_No ~ Day_of_year, data)
data$Y_hat <- predict(model, data)

ggplot(data, aes(x=Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_line(aes(y = Y_hat)) +
  ggtitle("Plot of Draft Number vs. Day of birth without using ggplot loess")

set.seed(12345)

X <- data$Day_of_year
Y <- data$Draft_No

T <- numeric()

for (i in 1:2000) {
        boot <-sample(length(X), replace=TRUE)
        data1 <- cbind(X,boot)

        X_a <- data1[which.min(data1[,2])]
        X_b <- data1[which.max(data1[,2])]

        model2 <- loess(Y~X, data=as.data.frame(data1), method="loess")

        fitted_X_a <- model2$fitted[X_a]
        fitted_X_b <- model2$fitted[X_b]

        test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
        T[i] <- test
        }


pval <- length(which(T>=0))
```

```r
hist(T, breaks=30,
     main="Distribution of T by using non-parametric bootstrapping")

print("Estimated p-value:")
print(pval/2000)



data <- cbind(X,Y)
X_a <- data[which.min(data[,2])]
X_b <- data[which.max(data[,2])]

model <- loess(Y~X, data=as.data.frame(data), method="loess")

fitted_X_a <- model$fitted[X_a]
fitted_X_b <- model$fitted[X_b]

test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
print(test)
#To do permutation - we have to compare two groups (observed data group vs permutated group).
#for permutated group - We may permute labels (from the lecture note
#So basically, we are comparing between two groups - Before permutation vs After permutation.
set.seed(12345)
permu_T <- numeric()

for (i in 1:2000){
        permu_Y <- sample(1:length(Y), length(Y), replace=FALSE)
        permu_data <- cbind(X,permu_Y)

        #and do the same thing
        X_a <- permu_data[which.min(permu_data[,2])]
        X_b <- permu_data[which.max(permu_data[,2])]

        permu_model <- loess(permu_Y~X, data=as.data.frame(permu_data), method="loess")

        fitted_X_a <- permu_model$fitted[X_a]
        fitted_X_b <- permu_model$fitted[X_b]

        permu_test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
        permu_T[i] <- permu_test

}


hist(permu_T, breaks=30,
     main="Distribution of T by using permutation")

#compute p value(two-sided test)
pval2 <- length(which(abs(permu_T)>=abs(test)))

print("Estimated p-value:")
print(pval2/2000)
```

```r
#always go with set.seed(12345) for the same result

permu_test <- function(B,X,Y){

        #compute test statistics from original data
        data <- cbind(X,Y)
        X_a <- data[which.min(data[,2])]
        X_b <- data[which.max(data[,2])]

        model <- loess(Y~X, data=as.data.frame(data), method="loess")

        fitted_X_a <- model$fitted[X_a]
        fitted_X_b <- model$fitted[X_b]

        test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)

        #then compute the permuted data
        permu_T <- numeric()

        for (i in 1:B){
                permu_Y <- sample(1:length(Y), length(Y), replace=FALSE)
                permu_data <- cbind(X,permu_Y)

                #and do the same thing
                X_a <- permu_data[which.min(permu_data[,2])]
                X_b <- permu_data[which.max(permu_data[,2])]

                permu_model <- loess(permu_Y~X, data=as.data.frame(permu_data), method="loess")

                fitted_X_a <- permu_model$fitted[X_a]
                fitted_X_b <- permu_model$fitted[X_b]

                permu_test <- (fitted_X_b - fitted_X_a)/(X_b - X_a)
                permu_T[i] <- permu_test
        }


        #then compute the estimated p value
        estimated_pval <- length(which(abs(permu_T)>=abs(test)))/B
        return(estimated_pval)

}
set.seed(12345)

data <- read.csv("lottery.csv", sep=";")

new_Y <- function(alpha){
        X <- data$Day_of_year
        new_y <- numeric()

        for (i in 1:length(X)){
                beta <- rnorm(1,183,10)
                new_y[i] <- max(0,min(alpha*X[i]+beta,366))
        }

        return(new_y)
}

non_ran_Y <- new_Y(alpha=0.1)
head(non_ran_Y)
permu_test(B=200, X=data$Day_of_year,Y=non_ran_Y)

seq <- seq(0.2,10,by=0.1)
pvals <- numeric()

for (i in 1:length(seq)){
        alpha <- seq[i]
        newY <- new_Y(alpha)

        pvals[i] <- permu_test(B=200, X=data$Day_of_year, Y=newY)
}

signif_p <- which(pvals<0.05)
power <- 1-sum(pvals>0.05)/length(pvals)

list("total_number_of_hypothesis_test"=length(seq),
     "number_of_test_which_reject_H0"=length(signif_p),
     "power"=power)

price_data <- read.csv("prices1.csv", sep=";")

ggplot(data=price_data,aes(Price)) +
  geom_histogram(bins=20) +
  ggtitle("Histogram of Price")

cat("The mean price is",mean(price_data$Price))

# Estimation of mean of Price
stat_mean <- function(data, index){
    data <- data[index,]
    answer <- mean(data$Price)
    return(answer)
}

res <- boot::boot(data=price_data, statistic = stat_mean, R=2000)
res
plot(res,index = 1)

#95% CI for mean using percentile
boot.ci(res, index=1, type=c('perc'))

#95% CI for mean using bca
boot.ci(res, index=1, type=c('bca'))

#95% CI for mean using first order normal
boot.ci(res, index=1, type=c('norm'))
```

```r
# Bias-correction and Varience of Price
boot.fn <- function(data,index){
        d <- data[index]
        res <- mean(d)
}

boot.result <- boot(data=price_data$Price, statistic=boot.fn, R=1000)
bias_cor <- 2*mean(price_data$Price)-mean(boot.result$t)


list("bias_correction"=bias_cor, "variance_of_the_mean_price"=35.93^2)
boot.result
plot(boot.result)
boot.ci(boot.result)


result <- numeric()
n <- NROW(price_data)

for (i in 1:n){
        updated_price <- price_data$Price[-i]
        result[i] <- mean(updated_price)
}

var_T <- (n-1)/n*sum((result-mean(result))^2)
mean_T <- mean(result)

cat("The variance from jacknife method is:", var_T)

confidence_interval_jackknife <- c((mean_T - 1.96*var_T), (mean_T + 1.96*var_T))

confidence_interval_jackknife

intervals <- c("(1150-1011)","(1148-1011)","(1149-1013)","(1146-1007)")
length <- c(1150-1011, 1148-1011,1149-1013,1146-1007)
Center_of_interval <- c((1150+1011)/2,(1148+1011)/2,(1149+1013)/2,(1146+1007)/2)

dt <- data.frame(intervals, length, Center_of_interval,row.names = c("Normal", "Basic", "Percentile", "I

dt %>% kable(col.names = c("Confidence interval", "Length of interval","Center of interval"))
```