Lab 5

Group 9 - [Jiawei Wu (jiawu449), Roshni Sundaramurthy (rossu809)]

12 February 2019

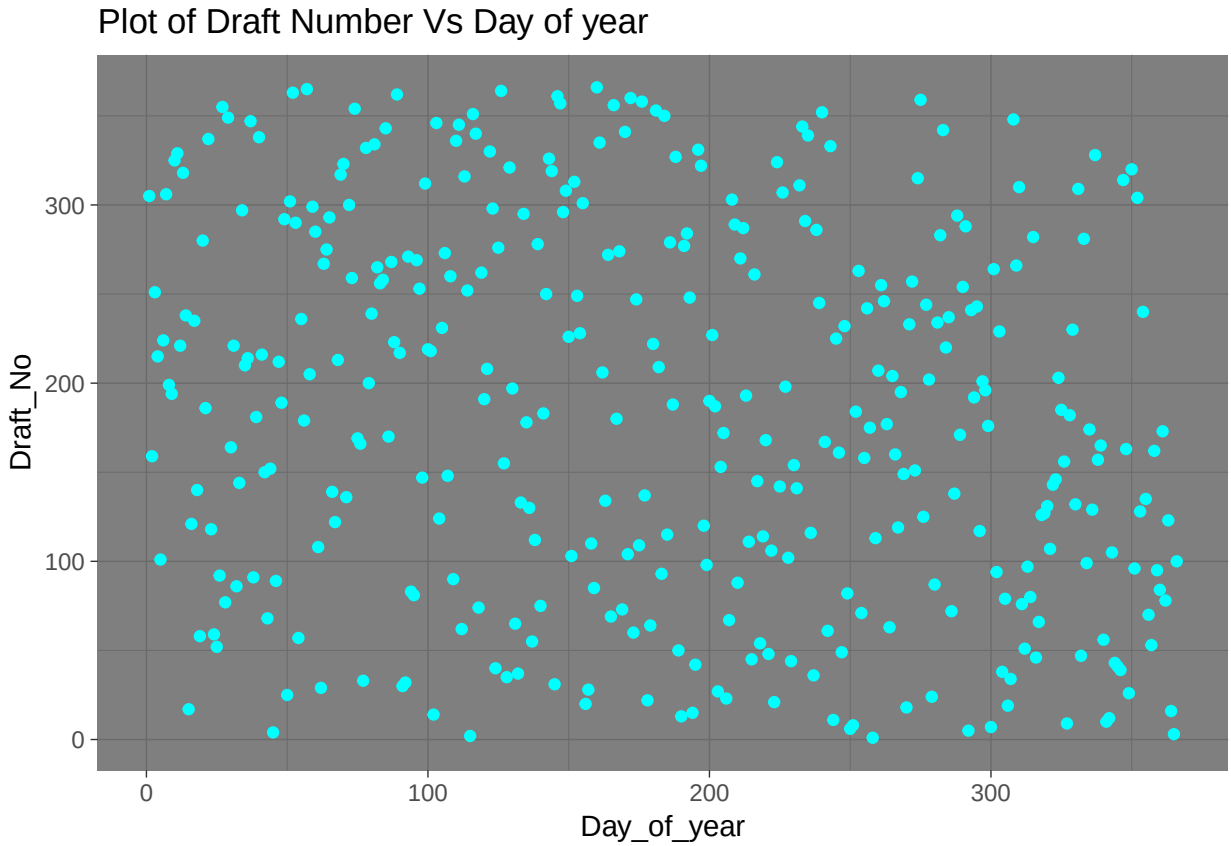
Contents

1 Hypothesis testing

In 1970, the US Congress instituted a random selection process for the military draft. All 366 possible birth dates were placed in plastic capsules in a rotating drum and were selected one by one. The first date drawn from the drum received draft number one, the second date drawn received draft number two, etc. Then, eligible men were drafted in the order given by the draft number of their birth date. In a truly random lottery there should be no relationship between the date and the draft number. Your task is to investigate whether or not the draft numbers were randomly selected. The draft numbers (Y=Draft No) sorted by day of year (X=Day of year) are given in the file lottery.xls.

1.0.1 Make a scatterplot of Y versus X and conclude whether the lottery looks random.

```
lottery_df <- read.csv("lottery.csv", sep = ";")
library(ggplot2)
plot1 <- ggplot(lottery_df)+geom_point(aes(Day_of_year,Draft_No),colour=117733)+
         ggtitle("Plot of Draft Number Vs Day of year")+theme_dark()
plot1
```



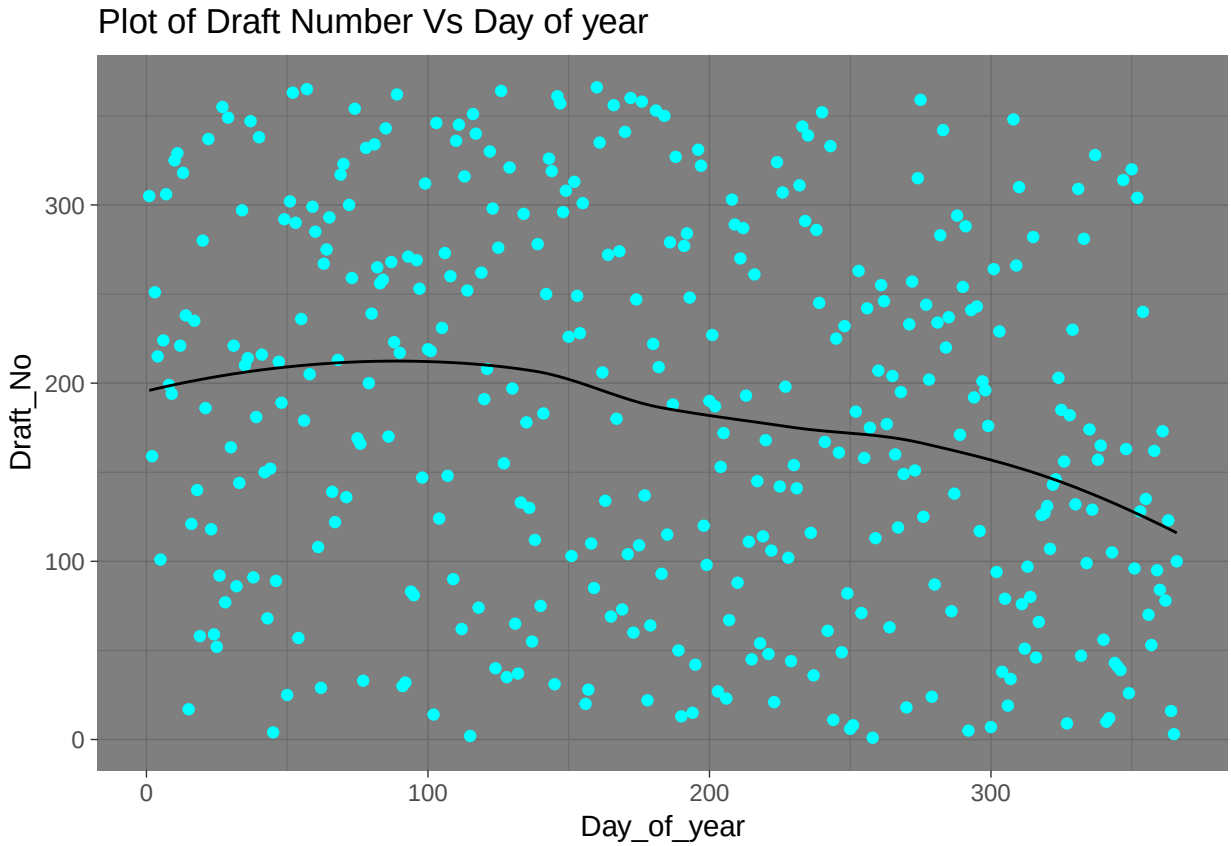

```
#plot(lottery_df$Day_of_year,lottery_df$Draft_No)
```

From the plot, it is observed that the lottery looks random visually.

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

```
model <- loess(Draft_No ~ Day_of_year, lottery_df)
Yhat <- fitted.values(model)
#plot(fit, ylim = c(0,400))
#points(lottery_df$Day_of_year,lottery_df$Draft_No)

ggplot(lottery_df, aes(x=Day_of_year, y = Draft_No)) + geom_point(colour=117733) +
  geom_line(aes(y = Yhat)) + theme_dark()+
  ggtitle("Plot of Draft Number Vs Day of year")
```



Yhat is estimated as a function of X by using loess smoother and plotted over the previous graph. The curve seems to follow a distribution such that any person can be choosen based on number of days in a year. So now, the lottery does not seems to be random.

1.0.3 To check whether the lottery is random, it is reasonable to use test statistics. If this value is significantly greater than zero, then there should be a trend in the data and the lottery 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?

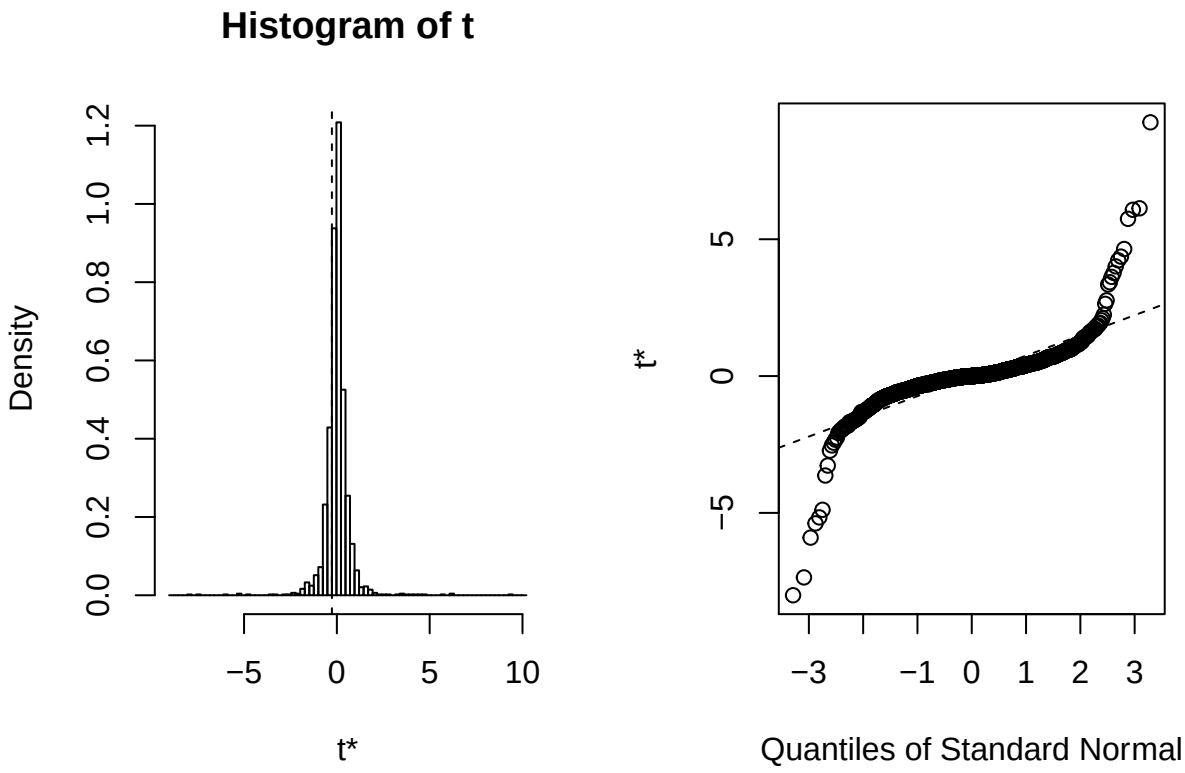
```
library(boot)
bs <- function(formula, data, indices){
  d <- data[indices,] # allows boot to select sample
  fit <- loess(formula, data=d)
  #return(fit$coefficients[2])
  Xa <- d$Day_of_year[which.min(d$Draft_No)]
  Xb <- d$Day_of_year[which.max(d$Draft_No)]
  #pred <- predict(fit,d)
  pred <- fitted.values(fit)
  Yhat_Xa <- pred[Xa]
  Yhat_Xb <- pred[Xb]
  test_stat <- (Yhat_Xb - Yhat_Xa) / (Xb - Xa)
  test_stat
}
# bootstrapping with 2000 replications
results <- boot(data=lottery_df, statistic=bs,
```

```
   R=2000, formula=Draft_No ~ Day_of_year)

# view results
results
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = lottery_df, statistic = bs, R = 2000, formula = Draft_No ~
##     Day_of_year)
##
##
## Bootstrap Statistics :
##       original    bias    std. error
## t1* -0.2671794 0.2747321   0.7391435
```

```
plot(results, index=1) # intercept
```




```
# get 95% confidence intervals
boot.ci(results, type="bca", index=1)
```

```
## Warning in norm.inter(t, adj.alpha): extreme order statistics used as
## endpoints
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca", index = 1)
```

```
##
## Intervals :
## Level       BCa
## 95%   (-8.0085,  0.0985 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```
cat(paste("p_value:",mean(results$t >0)))
```

```
## p_value: 0.4935
```

From the results, the original t value is *-0.2671794* and this value is significantly lesser than zero. So, the lottery is random. It is evident from the density plot, t value remains below *0*.

1.0.4 Implement a function depending on data and B that tests the hypothesis H0: Lottery is random versus H1: Lottery is non-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.

```
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
```

```
new_df <- lottery_df %>% select(c(Day, Mo.Number, Day_of_year, Draft_No))
fit <- loess(Draft_No ~., new_df)
permutation <- function(data, B, ...){
  Xa <- which.min(data$Day_of_year)
  Xb <- which.max(data$Day_of_year)
  #pred <- fitted.values(fit)
  pred<-predict(fit,data,type="response")
  Yhat_Xa <- pred[Xa]
  Yhat_Xb <- pred[Xb]
  test_stat <- (Yhat_Xb - Yhat_Xa) / (Xb - Xa)
  test_stat
  B=2000  # slide 16
  stat=numeric(B)
  n=dim(data)[1]
  for (b in 1:B){
    Gb=sample(data$Day_of_year , n, replace = F) # without replacement since number of permutations is
    stat[b]=(pred[which.max(Gb)] - pred[which.min(Gb)]) / (which.max(Gb) - which.min(Gb))
  }
  p_value <- mean(stat>=test_stat)
  #print(c(test_stat,mean(stat>test_stat)))
  cat(paste("p-value:", p_value, "\n"))
  if(p_value < 0.05){paste("\n The null hypothesis is rejected")}
  else{paste("The null hypothesis is not rejected")}
```

```
}

permutation(new_df, 2000, set.seed(123))
```

```
## p-value: 0.5695
```

```
## [1] "The null hypothesis is not rejected"
```

As specified, the permutation test function is implemented to return the p-value. We tested this function on our data with B = 2000 and the p-value is showed above. It seems that the computed p-value is greater than the significance criterion (alpha value = 0.05). So, the null hypothesis can't be rejected.

1.0.5 Make a crude estimate of the power of the test constructed in Step 4: 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 $\beta \tilde{N}(183; sd = 10)$. b) Plug these data into the permutation test with B = 200 and note whether it was rejected. c) Repeat Steps 5a-5b for $\alpha = 0.2, 0.3, ..., 10$. What can you say about the quality of your test statistics considering the value of the power?

```
X <- lottery_df$Day_of_year
alpa <- 0.1
beta <- rnorm(length(X),mean=183,sd=10)
Y=max(0, min(alpa*X + beta, 366))
new_df1 <- as.data.frame(cbind(X,Y))
new_df1$Day_of_year <- new_df1$X
fit <- loess(Y ~ Day_of_year, data=new_df1)
pred <- predict(fit, new_df1, type="response")
permutation(data=new_df1, B=200, alpa=0.1, pred)
```

```
## p-value: 0.6075
```

```
## [1] "The null hypothesis is not rejected"
```

```
many_alpa<- seq(0.2,10,0.1)
a<-data.frame()
b <- data.frame()
for (i in many_alpa){
  b <- permutation(data=new_df1, B=200, alpha=i, pred)
  a <- rbind(a,b)
}
```

```
## p-value: 0.6025
## p-value: 0.607
## p-value: 0.6055
## p-value: 0.602
## p-value: 0.6175
## p-value: 0.601
## p-value: 0.5815
## p-value: 0.596
## p-value: 0.59
## p-value: 0.6065
## p-value: 0.599
## p-value: 0.6075
## p-value: 0.6095
## p-value: 0.598
```

```
## p-value: 0.6175
## p-value: 0.587
## p-value: 0.59
## p-value: 0.605
## p-value: 0.6015
## p-value: 0.614
## p-value: 0.591
## p-value: 0.615
## p-value: 0.594
## p-value: 0.6085
## p-value: 0.6
## p-value: 0.603
## p-value: 0.596
## p-value: 0.592
## p-value: 0.5905
## p-value: 0.5805
## p-value: 0.5865
## p-value: 0.5915
## p-value: 0.6015
## p-value: 0.6025
## p-value: 0.603
## p-value: 0.593
## p-value: 0.6
## p-value: 0.5935
## p-value: 0.6005
## p-value: 0.6025
## p-value: 0.591
## p-value: 0.5945
## p-value: 0.6045
## p-value: 0.61
## p-value: 0.611
## p-value: 0.605
## p-value: 0.5675
## p-value: 0.5905
## p-value: 0.597
## p-value: 0.6045
## p-value: 0.617
## p-value: 0.6165
## p-value: 0.5915
## p-value: 0.6095
## p-value: 0.621
## p-value: 0.6025
## p-value: 0.605
## p-value: 0.6025
## p-value: 0.601
## p-value: 0.5875
## p-value: 0.6045
## p-value: 0.5965
## p-value: 0.597
## p-value: 0.6005
## p-value: 0.618
## p-value: 0.6075
## p-value: 0.6035
## p-value: 0.6035
```

```
## p-value: 0.6095
## p-value: 0.596
## p-value: 0.6085
## p-value: 0.5965
## p-value: 0.593
## p-value: 0.6035
## p-value: 0.59
## p-value: 0.5995
## p-value: 0.598
## p-value: 0.596
## p-value: 0.6025
## p-value: 0.6005
## p-value: 0.5955
## p-value: 0.582
## p-value: 0.6075
## p-value: 0.6085
## p-value: 0.6105
## p-value: 0.606
## p-value: 0.5985
## p-value: 0.6
## p-value: 0.615
## p-value: 0.605
## p-value: 0.5895
## p-value: 0.604
## p-value: 0.5835
## p-value: 0.5995
## p-value: 0.613
## p-value: 0.602
## p-value: 0.623
## p-value: 0.584
## p-value: 0.5925
```

```
#a # prints that the null hypothesis is not rejected for any values.
```

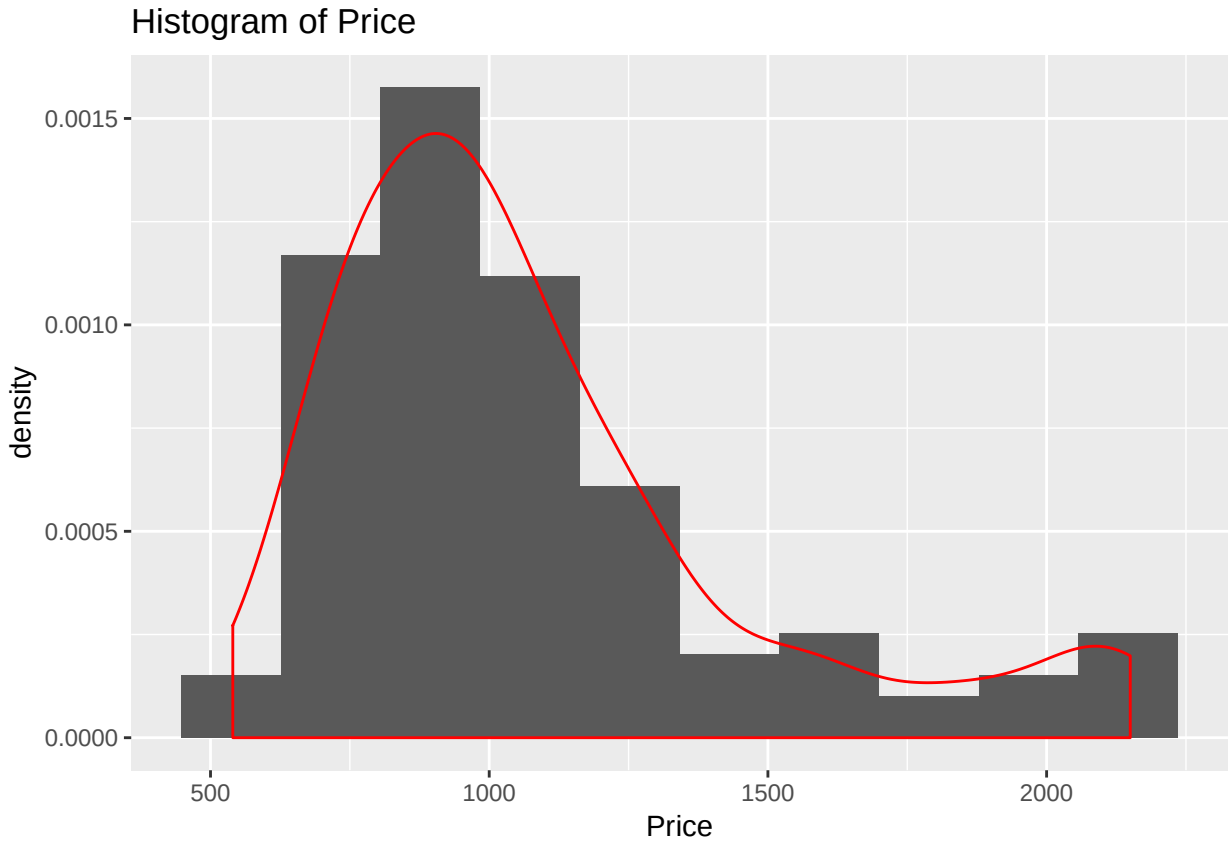
The new dataset with X as Day of year and Y as $max(0; min(\alpha * x + \beta; 366))$ has been created. It is used with our permutation test with B=200. The computed p-value still seems to be greater than alpha value. So, the null hypothesis can't be rejected. Again it's been tested with many alpha values. The power value seems to be still high. The quality of our test statistics considering the value of the power is quite not good.

2 Bootstrap, jackknife and confidence intervals

The data you are going to continue analyzing is the database of home prices in Albuquerque, 1993. The variables present are Price; SqFt: the area of a house; FEATS: number of features such as dishwasher, refrigerator and so on; Taxes: annual taxes paid for the house. Explore the file prices1.xls.

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

```
Pricedata <- read.csv2("prices1.csv")
#hist(Pricedata$Price, main = "Histogram of Price")
ggplot(Pricedata,aes(x=Price))+geom_histogram(bins = 10,aes(y=..density..))+geom_density(color="red")+la
```




```
cat(paste("Mean price:", round(mean(Pricedata$Price),4)))
```

```
## Mean price: 1080.4727
```

The histogram is shown above, and from the density line, it seems to remain a Gamma distribution. And the mean of the price is 1080.4727.

2.0.2 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 (Hint: use boot(),boot.ci(),plot.boot(),print.bootci())

```
library("boot")
set.seed(1234567)
my.mean <- function(x, indices) {
  return(mean(x[indices]))
  }
res = boot(Pricedata$Price, my.mean, R=10000)
res
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Pricedata$Price, statistic = my.mean, R = 10000)
##
```

```
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 1080.473 0.6439055    35.84212
```

```
print(boot.ci(res,type=c('perc','bca','norm')))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res, type = c("perc", "bca", "norm"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (1010, 1150 )   (1014, 1154 )   (1017, 1159 )
## Calculations and Intervals on Original Scale
```

As the results shows above, the bootstrap bias-correction is 0.6439055, variance of the mean price is 1080.473.

The confidence interval for the mean price using bootstrap percentile is $(1014, 1154)$;

The confidence interval for the mean price using bootstrap BCa is $(1017, 1159)$;

The confidence interval for the mean price using first-order normal approximation is $(1010, 1150)$;

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

```
require("bootstrap")
```

```
## Loading required package: bootstrap
```

```
res1 = jackknife(Pricedata$Price, my.mean)
mean(res1$jack.values)
```

```
## [1] 1080.473
```

```
res1$jack.bias
```

```
## [1] 0
```

```
res1$jack.se
```

```
## [1] 36.34434
```

Table 1: Compare

	Jackknife	Bootstrap
mean price	1080.4727	1080.4730
bias	0.0000	0.6439
standard error	36.3443	35.8421

As the result shows, the estimate variance of the mean price is 1080.473.

Compared to the bootstrap, we can see the Jackknife result is more close to the real value, and the bias value is 0, which is much better than the bootstrap results. But the standard error of the Jackknife is larger than

the bootstrap. We can say the Jackknife method will overestimate the variance, and it will lead to a larger error compared to the Bootstrap method.

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

```
ci<-boot.ci(res)
```

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

```
bca<-ci$bca
perc<-ci$percent
norm<-ci$normal
```

Table 2: Compare between intervals

	BCa	Percentil	normal approximation
length of interval	141.35610	140.22545	140.49854
difference between estimate mean and lower bound	63.16947	66.55318	70.89318
difference between estimate mean and upper bound	78.18666	73.67227	69.60537

As the comparision table shows, the bootstrap BCa has the largest interval, bootstrap percentil has the smallest interval. And in BAc and Percentil's situation, the estimated mean is much closer to the intervals' lower bound, and located at the left part of the intervals. And in normal approximation's interval, estimated mean is almost at the center of the interval.

3 Appendix

```
knitr::opts_chunk$set(echo = TRUE)
lottery_df <- read.csv("lottery.csv", sep = ";")
library(ggplot2)
plot1 <- ggplot(lottery_df)+geom_point(aes(Day_of_year,Draft_No),colour=117733)+
         ggtitle("Plot of Draft Number Vs Day of year")+theme_dark()
plot1
#plot(lottery_df$Day_of_year,lottery_df$Draft_No)
model <- loess(Draft_No ~ Day_of_year, lottery_df)
Yhat <- fitted.values(model)
#plot(fit, ylim = c(0,400))
#points(lottery_df$Day_of_year,lottery_df$Draft_No)

ggplot(lottery_df, aes(x=Day_of_year, y = Draft_No)) + geom_point(colour=117733) +
  geom_line(aes(y = Yhat)) + theme_dark()+
  ggtitle("Plot of Draft Number Vs Day of year")

library(boot)
bs <- function(formula, data, indices){
  d <- data[indices,] # allows boot to select sample
  fit <- loess(formula, data=d)
  #return(fit$coefficients[2])
```

```r
  Xa <- d$Day_of_year[which.min(d$Draft_No)]
  Xb <- d$Day_of_year[which.max(d$Draft_No)]
  #pred <- predict(fit,d)
  pred <- fitted.values(fit)
  Yhat_Xa <- pred[Xa]
  Yhat_Xb <- pred[Xb]
  test_stat <- (Yhat_Xb - Yhat_Xa) / (Xb - Xa)
  test_stat
}
# bootstrapping with 2000 replications
results <- boot(data=lottery_df, statistic=bs,
   R=2000, formula=Draft_No ~ Day_of_year)

# view results
results
plot(results, index=1) # intercept

# get 95% confidence intervals
boot.ci(results, type="bca", index=1)

cat(paste("p_value:",mean(results$t >0)))

library(dplyr)
new_df <- lottery_df %>% select(c(Day, Mo.Number, Day_of_year, Draft_No))
fit <- loess(Draft_No ~., new_df)
permutation <- function(data, B, ...){
  Xa <- which.min(data$Day_of_year)
  Xb <- which.max(data$Day_of_year)
  #pred <- fitted.values(fit)
  pred<-predict(fit,data,type="response")
  Yhat_Xa <- pred[Xa]
  Yhat_Xb <- pred[Xb]
  test_stat <- (Yhat_Xb - Yhat_Xa) / (Xb - Xa)
  test_stat
  B=2000  # slide 16
  stat=numeric(B)
  n=dim(data)[1]
  for (b in 1:B){
    Gb=sample(data$Day_of_year , n, replace = F) # without replacement since number of permutations is
    stat[b]=(pred[which.max(Gb)] - pred[which.min(Gb)]) / (which.max(Gb) - which.min(Gb))
  }
  p_value <- mean(stat>=test_stat)
  #print(c(test_stat,mean(stat>test_stat)))
  cat(paste("p-value:", p_value, "\n"))
  if(p_value < 0.05){paste("\n The null hypothesis is rejected")}
  else{paste("The null hypothesis is not rejected")}
}

permutation(new_df, 2000, set.seed(123))
X <- lottery_df$Day_of_year
alpa <- 0.1
beta <- rnorm(length(X),mean=183,sd=10)
Y=max(0, min(alpa*X + beta, 366))
```

```r
new_df1 <- as.data.frame(cbind(X,Y))
new_df1$Day_of_year <- new_df1$X
fit <- loess(Y ~ Day_of_year, data=new_df1)
pred <- predict(fit, new_df1, type="response")
permutation(data=new_df1, B=200, alpa=0.1, pred)
many_alpa<- seq(0.2,10,0.1)
a<-data.frame()
b <- data.frame()
for (i in many_alpa){
  b <- permutation(data=new_df1, B=200, alpha=i, pred)
  a <- rbind(a,b)
}
#a # prints that the null hypothesis is not rejected for any values.
Pricedata <- read.csv2("prices1.csv")
#hist(Pricedata$Price, main = "Histogram of Price")
ggplot(Pricedata,aes(x=Price))+geom_histogram(bins = 10,aes(y=..density..))+geom_density(color="red")+la
cat(paste("Mean price:", round(mean(Pricedata$Price),4)))

library("boot")
set.seed(1234567)
my.mean <- function(x, indices) {
  return(mean(x[indices]))
  }
res = boot(Pricedata$Price, my.mean, R=10000)
res
print(boot.ci(res,type=c('perc','bca','norm')))
require("bootstrap")
res1 = jackknife(Pricedata$Price, my.mean)
mean(res1$jack.values)
res1$jack.bias
res1$jack.se
knitr::kable(
cbind('Jackknife'=c('mean price'=round(mean(res1$jack.values),4),'bias'=res1$jack.bias,'standard error'=

ci<-boot.ci(res)
bca<-ci$bca
perc<-ci$percent
norm<-ci$normal
knitr::kable(
cbind('BCa'=c('length of interval'=round(bca[5]-bca[4],4),'difference between estimate mean and lower b
```