Computational Statistics Lab

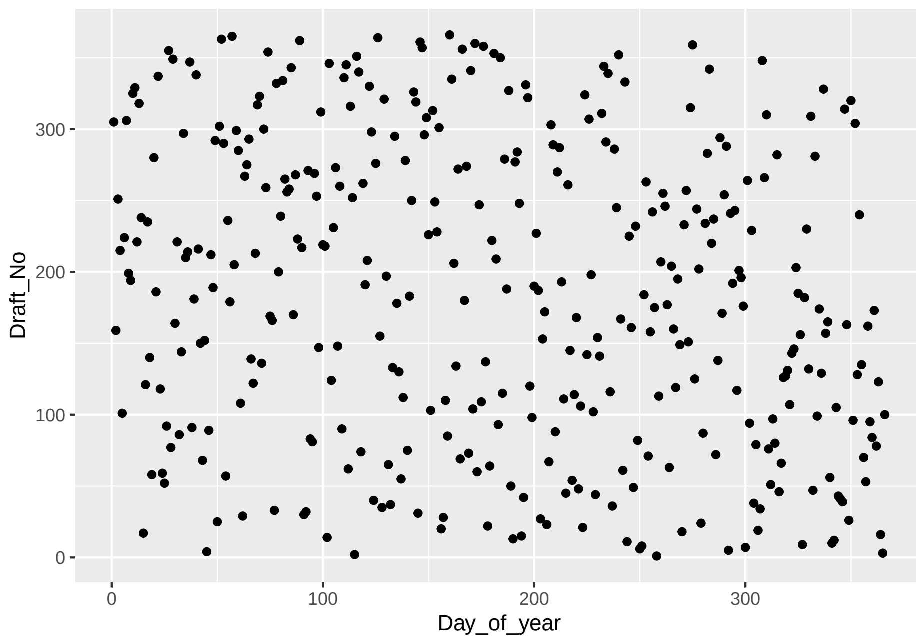
Lab 5 report

Jonathan Dorairaj, Yi-Hung Chen

2022-12-13

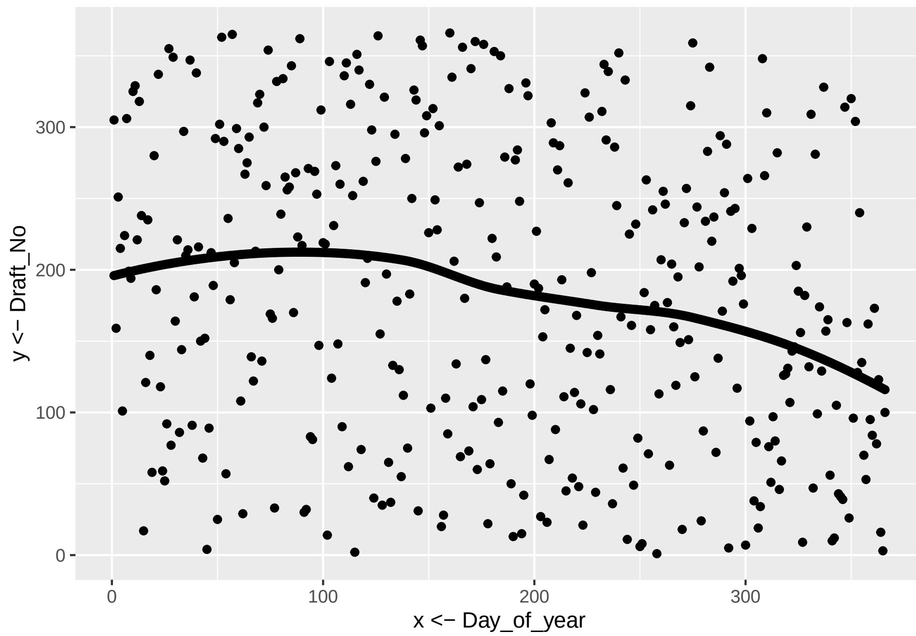
Question 1: Hypothesis testing

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



By observing the above graph, it can be conclude that the lottery looks random, since there is no obvious relation between Date(Day__of__year) and Draft Number(Draft__No)

2.Compute an estimate $\hat{Y}$ of the expected response as a function of X by using a loess smoother



According to the graph, the curve has a slight downward trend instead of a flat line. This indicate the lottery might not be truly random.

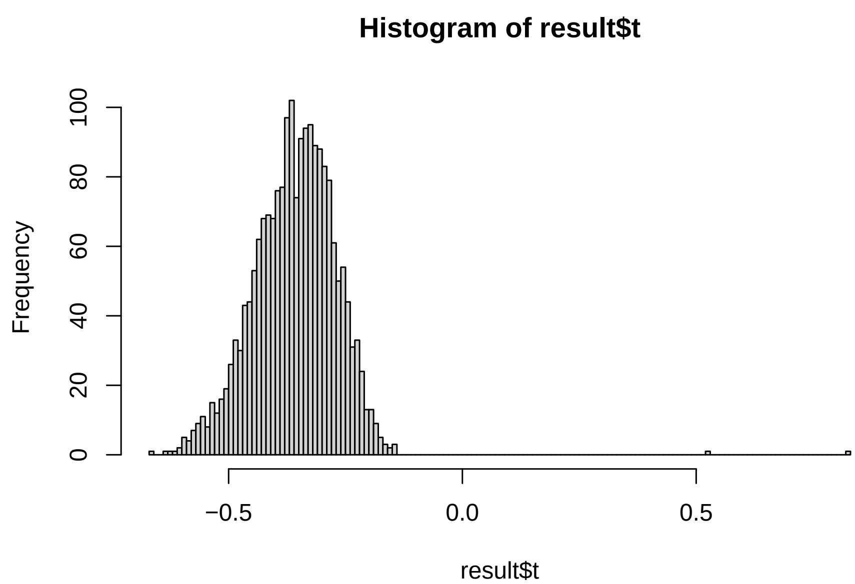
3.Using Test statistic to check whether the lottery is random (using a non–parametricbootstrap)

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

If this value is different from 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–quantile of T=0?

```
my_test_statistic <- function(data,i){
  bootdata <- data[i,]

  # We first take the loess smoother since we need to get X is from Yhat
  loessMod <- loess(Draft_No ~ Day_of_year, data<-bootdata )
  loess_prediction <-loessMod$fitted
  index_of_xb <-  which.max(loess_prediction)
  index_of_xa <-  which.min(loess_prediction)
  yb <- loess_prediction[index_of_xb]
  ya <- loess_prediction[index_of_xa]
  xb <-  bootdata$Day_of_year[index_of_xb]
  xa <-  bootdata$Day_of_year[index_of_xa]

  test_statistic <-(yb-ya)/(xb -xa)
  return(test_statistic)
```

```
}

#Using bootstrap with B = 2000
set.seed(12345)
result <- boot(data=lottery, statistic=my_test_statistic,2000)

#As we are given if T is significant different from zero, the data is not random
quantile <- length(which(result$t!=0))/length(result$t)
hist(result$t,breaks = 200)
```



Since the value of test statistic -0.3479163 ,which is not equal to zero, this means the lottery is likely to be random. The p–quantile of T=0 is calculate as $\frac{number\ of\ T \neq 0}{total\ number\ of\ T}$, which is 1

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.

```
#For permutation test, we use the example code from the course website

 hypo_test <- function(data,B){

     stat <- numeric(B)
     n <- dim(data)[1]
     for(b in 1:B){
     Y_sample<-sample(lottery$Draft_No, n)
```

```
      loessMod <- loess(Y_sample ~ data$Day_of_year )
      loess_prediction <-loessMod$fitted
      index_of_xb <-  which.max(loess_prediction)
      index_of_xa <-  which.min(loess_prediction)
      yb <- loess_prediction[index_of_xb]
      ya <- loess_prediction[index_of_xa]
      xb <-  data$Day_of_year[index_of_xb]
      xa <-  data$Day_of_year[index_of_xa]

      test_statistic <-(yb-ya)/(xb -xa)
      stat[b] <- test_statistic
      }

      # We use the calculation for two-sided test according to the lecture slide p.11,
      t0 <- my_test_statistic(data)
      pvalue <- sum(abs(stat) >=abs(t0))/B

      return(pvalue)

    }
set.seed(12345)

pvalue2 <- hypo_test(lottery,2000)
```

The P-value is calculate as 0.1595 which is larger than 0.05, so we cannot reject H0. Thus we conclude that lottery is random.

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 $\beta\sim$ N (183,sd = 10).

```
new_Y <- function(alpha){
  new_lottery <- data.frame(lottery$Day_of_year)
  new_yx <- numeric()

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

  new_lottery <- data.frame(new_lottery,new_yx)
  colnames(new_lottery) <- c("Day_of_year","Draft_No")

  return(new_lottery)
}
set.seed(12345)
new_y_01 <- new_Y(0.1)
head(new_y_01)
```

```
##   Day_of_year Draft_No
```

```
## 1           1 188.9553
## 2           2 190.2947
## 3           3 182.2070
## 4           4 178.8650
## 5           5 189.5589
## 6           6 165.4204
```

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

```
pvalue_01 <- hypo_test(new_y_01,200)
```

P-value is 0.895 which is greater than 0.05 so we cannot reject H0, which means the lottery is random

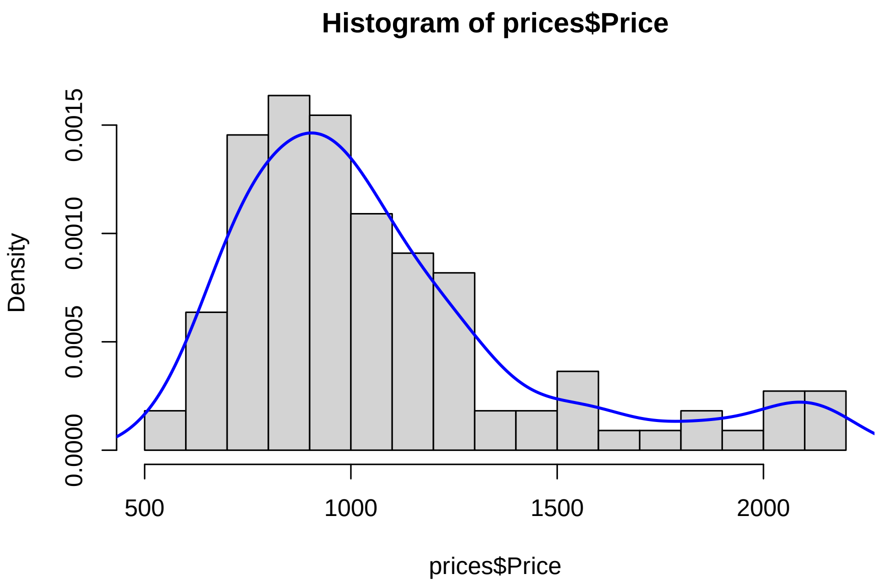
(c) Repeat Steps 5a–5b for $\alpha = 0.01, 0.02, \ldots, 1$.

```
alphas <- seq(from=0.01, to=1,by=0.01)
set.seed(12345)
pvalue_of_diff_alpha <- numeric(length(alphas))

for (i in 1:length(alphas)){
  new_alpha_lottery <- new_Y(alphas[i])
  pvalue_of_diff_alpha[i]<- hypo_test(new_alpha_lottery,200)

}
power <- length(which(pvalue_of_diff_alpha<0.05))/length(pvalue_of_diff_alpha)
```

The power(the probability to reject the null hypothesis) is 0.53. Which means the test statistic is not strong enough to differentiate random or non-random data sets.

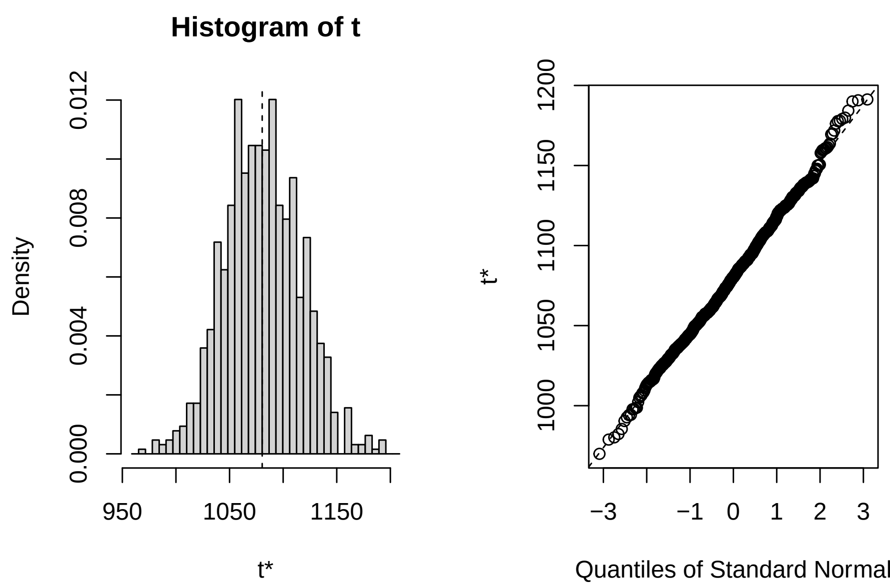
Question 2 : Bootstrap, jackknife and confidence intervals

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



The histogram of the mean prices resembles a Gamma Distribution approximately. The computed mean of the Price is 1080.4727273.

Q: 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.



The distribution of the mean price computed using bootstrap resembles a normal distribution according to the histogram above.

Bias-Correction Formula :

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

The bias-correction for the bootstrap is 1080.0920091. Variance :

$$Var[\hat{T(.)}] = \frac{1}{B-1}\sum_{i-1}^{B}(T(D_i^*) - T(\bar{D_i}))^2$$

The variance of the mean price is 1272.8363163.

Calculating 95% CI for the mean with different values for `type` in `boot.ci()` below:

```
result1 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'norm')
print(result1)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootobj, conf = 0.95, type = "norm")
##
## Intervals :
## Level      Normal
## 95%   (1010, 1150 )
## Calculations and Intervals on Original Scale
```

```
result2 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'perc')
print(result2)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootobj, conf = 0.95, type = "perc")
##
## Intervals :
## Level     Percentile
## 95%   (1014, 1150 )
## Calculations and Intervals on Original Scale
```

```
result3 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'bca')
print(result3)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bootobj, conf = 0.95, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   (1016, 1160 )
## Calculations and Intervals on Original Scale
```

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

Table 1: Comparing Variances

	JackKnife	Bootstrap
Variance	1320.911	1272.836

Comparing the variances of the mean price for the jackknife and bootstrap estimates, we see that the jackknife has a higher variance because, by leaving out some data, it gives more weight to the remaining data and thus increases the variance of the mean.

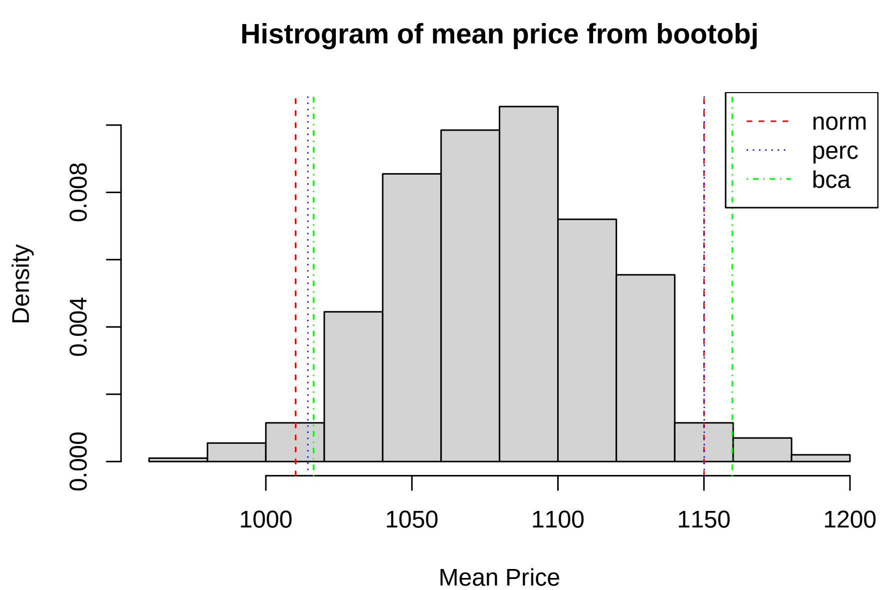
Q: Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.

Table 2: Comparing Confidence Intervals and Estimated Mean

	lower_CI	upper_CI
normal	1010.167	1150.017
bootstrap percentile	1014.379	1150.080
bootstrap bca	1016.314	1159.682

Table 3: Comparing length of Confidence Intervals relative to mean

	interval_length	lower_CI_to_mean	upper_CI_to_mean
normal	139.8506	70.30602	69.16387
bootstrap percentile	135.7015	66.47448	69.22699
bootstrap bca	143.3678	64.53926	78.82858



From Table 2 and Table 3 and the histogram above, we can compare the confidence intervals calculated using the `normal,percent and bca` methods. We can see that the intervals calculated from the `bootstrap bca` methods have the longest interval length for the same confidence level. The mean is located approximately equidistant from the lower and upper CI's calculated using the `norm` method. The location of the mean is closer to the lower CI with respect to the CI's calculated according to type = `bca` and `percent`.

Appendix

```
library(ggplot2)
library(boot)

#Q1.1 load data and polt it to check if the data is random

 lottery <- read.csv(file = 'lottery.csv',sep = ";") # use sep to divide the data in to different colum

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

#Q1.2 Using loess smoother to check if the data is random

loessMod <- loess(Draft_No ~ Day_of_year, data<-lottery )
loess_prediction<- predict(loessMod)
lottery <- cbind(lottery,as.vector(loess_prediction))

ggplot(lottery,aes(x<-Day_of_year,y<-Draft_No))+
      geom_point()+
      geom_point(aes(x<-Day_of_year,y<-loess_prediction))

#Q1.3 Using teststatistic and bootstrap

my_test_statistic <- function(data,i){
  bootdata <- data[i,]

  # We first take the loess smoother since we need to get X is from Yhat
  loessMod <- loess(Draft_No ~ Day_of_year, data<-bootdata )
  loess_prediction <-loessMod$fitted
  index_of_xb <-  which.max(loess_prediction)
  index_of_xa <-  which.min(loess_prediction)
  yb <- loess_prediction[index_of_xb]
  ya <- loess_prediction[index_of_xa]
  xb <-  bootdata$Day_of_year[index_of_xb]
  xa <-  bootdata$Day_of_year[index_of_xa]

  test_statistic <-(yb-ya)/(xb -xa)
  return(test_statistic)


}

#Using bootstrap with B = 2000
  set.seed(12345)
  result <- boot(data=lottery, statistic=my_test_statistic,2000)
  quantile <- length(which(result$t!=0))/length(result$t) #As we are given if T is significant differen
  hist(result$t,breaks = 200)

#Q1.4 Hypothesis Testing and Permutation test

  #For permutation test, we use the example code from the course website
```

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

      stat <- numeric(B)
      n <- dim(data)[1]
      for(b in 1:B){
      Y_sample<-sample(lottery$Draft_No, n)

      loessMod <- loess(Y_sample ~ data$Day_of_year )
      loess_prediction <-loessMod$fitted
      index_of_xb <-  which.max(loess_prediction)
      index_of_xa <-  which.min(loess_prediction)
      yb <- loess_prediction[index_of_xb]
      ya <- loess_prediction[index_of_xa]
      xb <-  data$Day_of_year[index_of_xb]
      xa <-  data$Day_of_year[index_of_xa]

      test_statistic <-(yb-ya)/(xb -xa)
      stat[b] <- test_statistic
      }
      t0 <- my_test_statistic(data)
      pvalue <- sum(abs(stat) >=abs(t0))/B  # We use the calculation for two-sided test according to th

      return(pvalue)

    }
  set.seed(12345)

  pvalue2 <- hypo_test(lottery,2000)

#Q1.5
#a
  new_Y <- function(alpha){
    new_lottery <- data.frame(lottery$Day_of_year)
    new_yx <- numeric()

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

    new_lottery <- data.frame(new_lottery,new_yx)
    colnames(new_lottery) <- c("Day_of_year","Draft_No")

    return(new_lottery)
  }
  set.seed(12345)
  new_y_01 <- new_Y(0.1)
  head(new_y_01)

#b
pvalue_01 <- hypo_test(new_y_01,200)

#c Check the power of test
```

```r
alphas <- seq(from=0.01, to=1,by=0.01)
set.seed(12345)
pvalue_of_diff_alpha <- numeric(length(alphas))

for (i in 1:length(alphas)){
  new_alpha_lottery <- new_Y(alphas[i])
  pvalue_of_diff_alpha[i]<- hypo_test(new_alpha_lottery,200)

}
power <- length(which(pvalue_of_diff_alpha<0.05))/length(pvalue_of_diff_alpha)
```

Question 2

```r
prices <- read.csv("prices1.csv",sep = ";")
prices <- as.data.frame(prices)

hist(x = prices$Price,breaks = 15)
mean(prices$Price)

#looks like a gamma distribution.
## 2.2
computemean <- function(newdata,i){
  d2 <- newdata[i,]
  return(mean(d2$Price))
}

set.seed(12345)

bootobj <- boot(data = prices,statistic = computemean,R = 1000)
bootobj

hist(x = bootobj$t,main = "Histrogram of mean price from bootobj",xlab = "Mean Price")
summary(bootobj)

plot(bootobj)

# Bias
mean(bootobj$t) - bootobj$t0

# Bias - correction
# R must be same as supplied in boot function
R <- 1000
bc_factor <- 2*mean(prices$Price) - 1/R * sum(bootobj$t)
bc_factor


# Standard Deviation
sd(bootobj$t)^2
#variance
var(bootobj$t)

# CI

result1 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'norm')
print(result1)

result2 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'perc')
print(result2)

result3 <- boot.ci(boot.out = bootobj,conf = 0.95,type = 'bca')
print(result3)

## 1.3
```

```r
n <- nrow(prices)
n
B <- 1000
newt <- c()
for (i in 1:B) {

  #generate new data set
  newdf <- prices[-i,]
  # newt[i] <- mean(newdf$Price)
  newt[i] <- ((n*mean(prices$Price)) - ((n-1)*mean(newdf$Price)))

}
#B in the second term because summation of newt is divided by n where n is nrow of newt
term1 <- sum((newt - (sum(newt)/B))^2)
term1
term2 <- term1/n
term2
jknife_var <- term2/(n-1)
jknife_var
#final jackknife variance
ssummary_df <- data.frame(jackknifevar = jknife_var,
                          biasotherwise = var(bootobj$t))
colnames(ssummary_df) <- c("JackKnife","Bootstrap")
rownames(ssummary_df) <- c("Variance")
ssummary_df

summary_df <- data.frame(lower_CI = numeric(),
                         upper_CI = numeric())
summary_df[1,] <- c(result1$normal[2:3])
summary_df[2,] <- c(result2$percent[4:5])
summary_df[3,] <- c(result3$bca[4:5])

row.names(summary_df) <- c("normal","bootstrap percentile",'bootstrap bca')

summary_df1 <- data.frame(interval_length= numeric(),
                         lower_CI_to_mean = numeric(),
                         upper_CI_to_mean = numeric())

summary_df1[1,] <- c(result1$normal[3]-result1$normal[2],
                     mean(bootobj$t0) - result1$normal[2],
                     result1$normal[3] - mean(bootobj$t))

summary_df1[2,] <- c(result2$percent[5]-result2$percent[4],
                     mean(bootobj$t) - result2$percent[4],
                     result2$percent[5] - mean(bootobj$t))

summary_df1[3,] <- c(result3$bca[5]- result3$bca[4],
                     mean(bootobj$t)-result3$bca[4],
                     result3$bca[5] - mean(bootobj$t) )

row.names(summary_df1) <- c("normal","bootstrap percentile",'bootstrap bca')

hist(x = bootobj$t,main = "Histrogram of mean price from bootobj",
```

```r
     xlab = "Mean Price",freq = F)
# hist(x = prices$Price,breaks = 15,freq = F)
abline(v =result1$normal[2:3],lty = 2, col = 'red')
abline(v =result2$percent[4:5],lty =3, col = 'blue')
abline(v =result3$bca[4:5], lty = 4 , col = 'green')
legend("topright", legend = c("norm", "perc",'bca'),
       col = c('red',"blue", "green"),lty = c(2,3,4),horiz = F)
```