Lab5 Computational Statistics

Andreas C Charitos(andch552) Omkar Bhutra (omkbh878)

27 Feb 2019

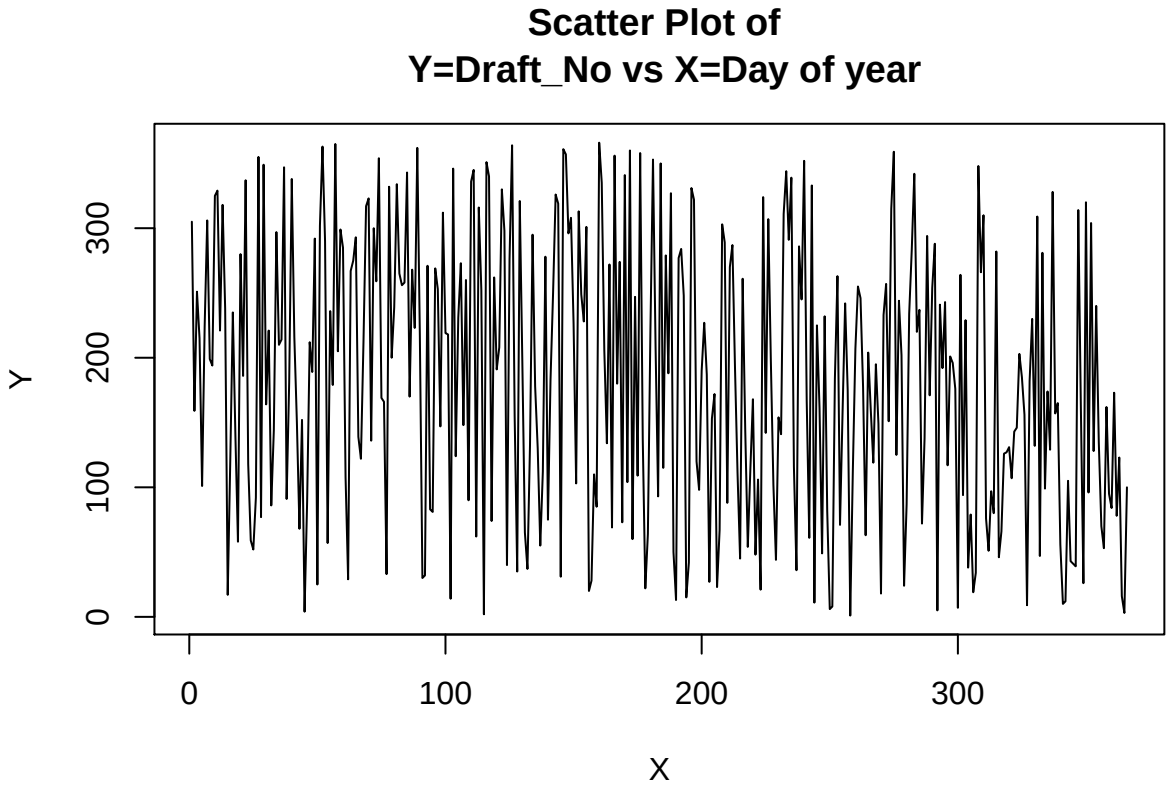
Question 1-Hypothesis testing

Subquestion 1-Scatterplot

```
#import the data
lottery<-readxl::read_excel("lottery.xls")
lottery<-as.data.frame(lottery)
```

The above plot shows the connection between Y=Draft_No (sorted by day of year) and X=Day of_year.As we conclude there doesn't seems to exists a specific patter so the relationship might considered as random.

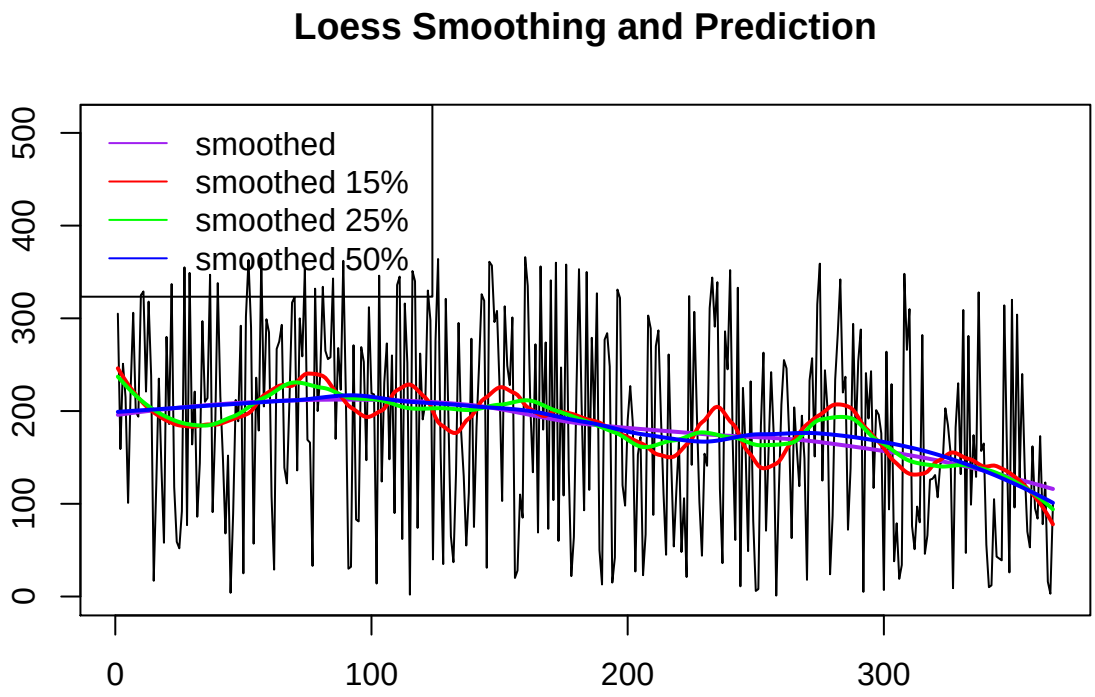
```
#assign the corresponding varaibles
Y=lottery$Draft_No
X=lottery$Day_of_year
#plot the data
plot(X,Y,type="l",main="Scatter Plot of \n Y=Draft_No vs X=Day of year")
```



Subquestion 2-Scatterplot with loess

Next we are compute $\hat{Y}$ using loess smoother with with 4 diffrent spans (default,15%,25%,50%).Again we can see that the curves are indicating that there seems to be a pattern in the original line so we can say that the relationship between Y and X is following a trend and doesn't seems to be random.

```
#calculate loess with diffrent spans
loessMod<-loess(Y~X)
loessMod15 <- loess(Y ~ X, span=0.15) # 15% smoothing span
loessMod25 <- loess(Y ~ X, span=0.25) # 25% smoothing span
loessMod50 <- loess(Y ~ X, span=0.50) #50% smoothing span

#make predictions with the loess models
smoothed<-predict(loessMod)
smoothed15 <- predict(loessMod15)
smoothed25 <- predict(loessMod25)
smoothed50 <- predict(loessMod50)

#scatterplot with loess models lines
plot(x=X,y=Y, type="l", main="Loess Smoothing and Prediction",
     xlab="", ylab="",ylim = c(0,510))
lines(smoothed,  x=X,col="purple",lwd=2)
lines(smoothed15, x=X, col="red",lwd=2)
lines(smoothed25, x=X, col="green",lwd=2)
lines(smoothed50, x=X, col="blue",lwd=2)
legend("topleft",legend = c("smoothed","smoothed 15% ",
                            "smoothed 25%","smoothed 50%"),col=c("purple","red","green","blue"),lty=1)
```



Subquestion 3-Randomness statistic evaluation

We are now going to check if the lottery is random using the following statistic:

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

by using by using a non parametric bootstrap for the previous statistic T with B = 2000

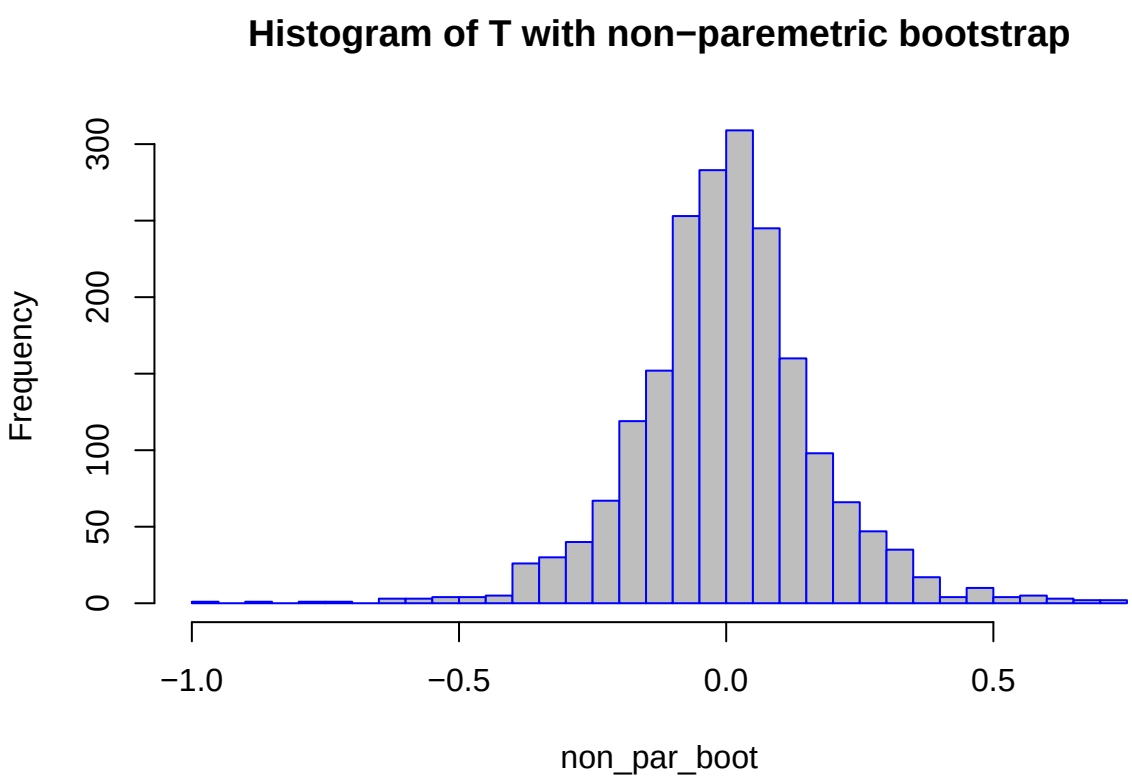
```
#set seed
set.seed(12345)

B=2000
#initialize vector for store the statistics
non_par_boot<-rep(0,B)

for(i in 1:B){
  #sample from the Y valus with replacement
  Y_samp<-sample(1:length(Y),length(Y),replace=T)
  #make a matrix
  dat<-cbind(X,Y_samp)
  #calculate the Xa,Xb
  Xb<-dat[which.max(dat[,2])] # the X that has max Y
  Xa<-dat[which.min(dat[,2])] # the Y that has min Y
  #predict model
  model<-loess(Y_samp ~ X,data=as.data.frame(dat),method="loess")
  #calculate Ya,Yb
  Y_xb<-model$fitted[Xb]
  Y_xa<-model$fitted[Xa]

  Tau<-(Y_xb-Y_xa)/(Xb-Xa)
  non_par_boot[i]<-Tau


}

#calculate the p value
p_val<-sum(non_par_boot>=0)/B

cat("The pvalue for the bootstrap sample is :",p_val)
```

```
## The pvalue for the bootstrap sample is : 0.5035
```

The p-value obtained is much larger that zero thus we can conclude that there seems to be a trend that the lottery is following and we conclude that is non random

Plot of the distribution of T with non parametric bootstrap

```
#histogram of the statistic distribution
hist(non_par_boot,breaks = 30,col="gray",include.lowest = TRUE,
     border="blue",main="Histogram of T with non-paremetric bootstrap")
```



The above plot shows the distribution of the statistic T obtained with non parametric bootstrap we can say that is following a normal curve but with long tails.

Subquestion 4-Hypothesis testing using permutation

In this part we are going to implement hypothesis tasting for our T statistic by using a permutation sampling.The null and alternative hypotheseis are :

$$H_o : Lottery \;\; is \;\; random$$

$$H_1 : Lottery \;\; is \;\; not \;\; random$$

```
set.seed(12345)

#sample size
B=2000
#permutation function
perm_func<-function(Y_value,X_value,B){

  call = match.call()
  ####################################################################
  #calculate the T statistic from the original data
  dat_origin<-cbind(X_value,Y_value)
  Xb_origin<-X_value[which.max(Y_value)]
  Xa_origin<-X_value[which.min(Y_value)]
```

```
  model_origin<-loess(Y_value ~ X_value,
                      data=as.data.frame(dat_origin),method="loess")

  Y_xb_origin<-model_origin$fitted[Xb_origin]
  Y_xa_origin<-model_origin$fitted[Xa_origin]

  Tau_origin<-(Y_xb_origin-Y_xa_origin)/(Xb_origin-Xa_origin)
  ################################################################

  #initialize vector to store perm statistics
  perm_Tau<-rep(0,B)

  for(i in 1:B){
    #take a sample from Y value without replacement
    Y_perm<-sample(1:length(Y_value),length(Y_value),replace=F)
    #make matrix
    dat_perm<-cbind(X_value,Y_perm)

    Xb<-dat_perm[which.max(dat_perm[,2])] #take X with max Y_perm
    Xa<-dat_perm[which.min(dat_perm[,2])]  #take X with min Y_prm
    #fit loess model
    model_perm<-loess(Y_perm ~ X_value,data=as.data.frame(dat_perm),method="loess")
    #calculate Ya,Yb for the predicted permutated data
    Y_xb<-model_perm$fitted[Xb]
    Y_xa<-model_perm$fitted[Xa]

    Tau_perm<-(Y_xb-Y_xa)/(Xb-Xa)  #calculate the T statistic
    perm_Tau[i]<-Tau_perm  #
  }

  perm_Tau<-perm_Tau
  p_value_perm<-sum(abs(perm_Tau)>=abs(Tau_origin))/B

  return(list(perm_Tau=perm_Tau,
              p_value_perm=p_value_perm,
         call=call))

}

#calculate B-permutated statistics
res=perm_func(lottery$Draft_No,lottery$Day_of_year,B)

cat("The pvalue from 2000 permutated sample is :",res$p_value_perm)
```

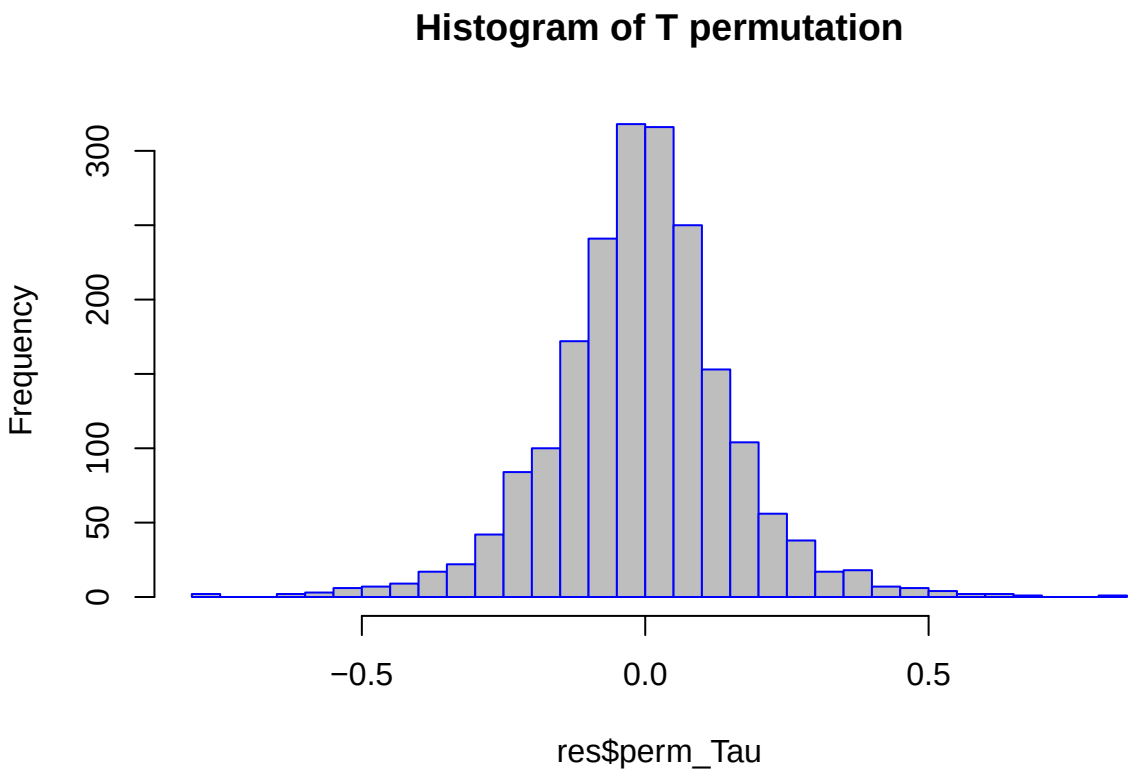
```
## The pvalue from 2000 permutated sample is : 0.086
```

From thhe obtained p-value which is 0.086 and using siginificant level $\alpha = 0.05$ since the p-value is greater we can't reject the null hypothesis, so according to our statistic usinf permutation the lottery is random.

Plot of the distribution of T with permuation

The above plot shows the distribution of the T with permutation sampling again the data seem to follow a normal curve but with long tails.

```r
#histogram of the permutated data
hist(res$perm_Tau,breaks = 30,col="gray",include.lowest = TRUE,
     border="blue",main = "Histogram of T permutation")
```



Subquestion 5-Hypothesis testing with permutation and generated data

Subquestion 5.a-Generate non-random data

In this part we are going to generate data according to the following pattern

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

using X(Day of year) and we are going to implement the previous hypothesis testing with permutation to see if the generated data are random.

```r
set.seed(12345)
#function to generate data
gen_data<-function(n,data,alpha){
  #initialize vector
  Y_gen<-c()
  #loop to create sample of size n
  for (i in 1:n){
    beta<-rnorm(1,183,10)
    x<-data[i]
    Y_gen[i]<-max(0,min(alpha*x+beta,366))
  }
```

```
  return(Y_gen)
}
```

Subquestion 5.b-Hypothesis testing with permutation on generated data

```
#generate data
Y_gen_data<-gen_data(366,lottery$Day_of_year,0.1)

#calculate permutation statistics for generated data
res1<-perm_func(Y_gen_data,lottery$Day_of_year,200)

cat("The pvalue for generated data is :",res1$p_value_perm)
```

```
## The pvalue for generated data is : 0.48
```

We can see that the pvalue that is obtained from our artificially generated data is bigger than an $\alpha = 0.05$,$p - value(> 0.05)$,so we fail to reject null hypothesis and our generated data with 95% confidence are random.

Subquestion 5.c-Hypothesis testing with permutation on generated data with diffrent a

Finally,we are going to use diffrent values for α to generate the data and calculate the pvalue for each sample with diffrent α in the threshold

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

```
set.seed(12345)
#initialize the step
step<-seq(0.2,10,by=0.1)
#initialize vector to store permutated p values
tafs<-rep(0,length(step))
#for loop
for (j in 1:length(step)){
  new_y_data<-gen_data(366,lottery$Day_of_year,step[j])
  p<-perm_func(new_y_data,lottery$Day_of_year,200)
  tafs[j]<-p$p_value_perm
}

#tafs

signif_p <- sum(tafs<0.05)
power <- 1-sum(tafs>0.05)/length(tafs)
cat("The number of significant permutated p values are :",signif_p,"\n",
    "The value of the power is :",power)
```

```
## The number of significant permutated p values are : 97
##  The value of the power is : 0.979798
```

From the results above, we can see that 97 of 99 the cases reject the null hypothesis, which indicates that the lottery is not random.

Question 2-Bootstrap,jacknife and ci

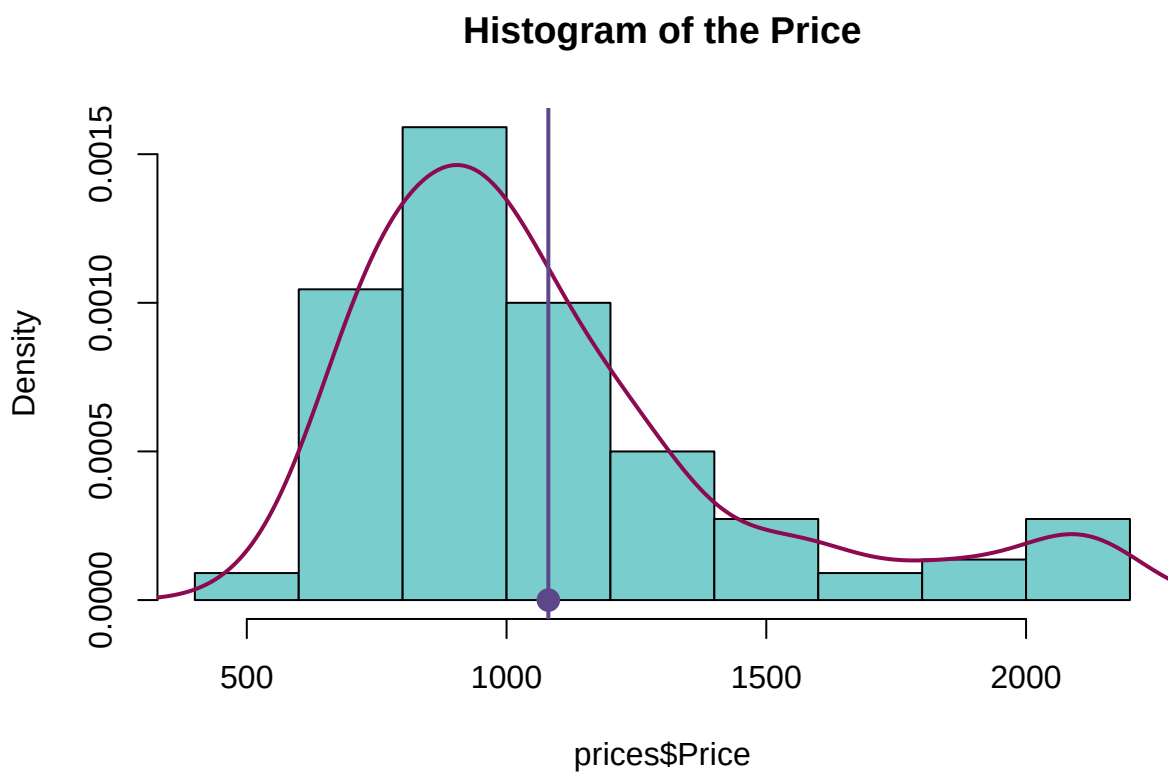
Subquestion 1-Histogram of Price

```
#import prices data
prices<-readxl::read_excel("prices1.xls")
prices<-as.data.frame(prices)

meanP<-mean(prices$Price)
cat("The mean of the Price is :",meanP)
```

```
## The mean of the Price is : 1080.473
```

Plot of the histogram of Price

```
hist(prices$Price,main="Histogram of the Price",
     col="darkslategray3",prob=T)
lines(density(prices$Price),col="deeppink4",lwd=2)
points(meanP,0,col="mediumpurple4",pch=19,cex=1.5)
abline(v=meanP,col="mediumpurple4",lwd=2)
```



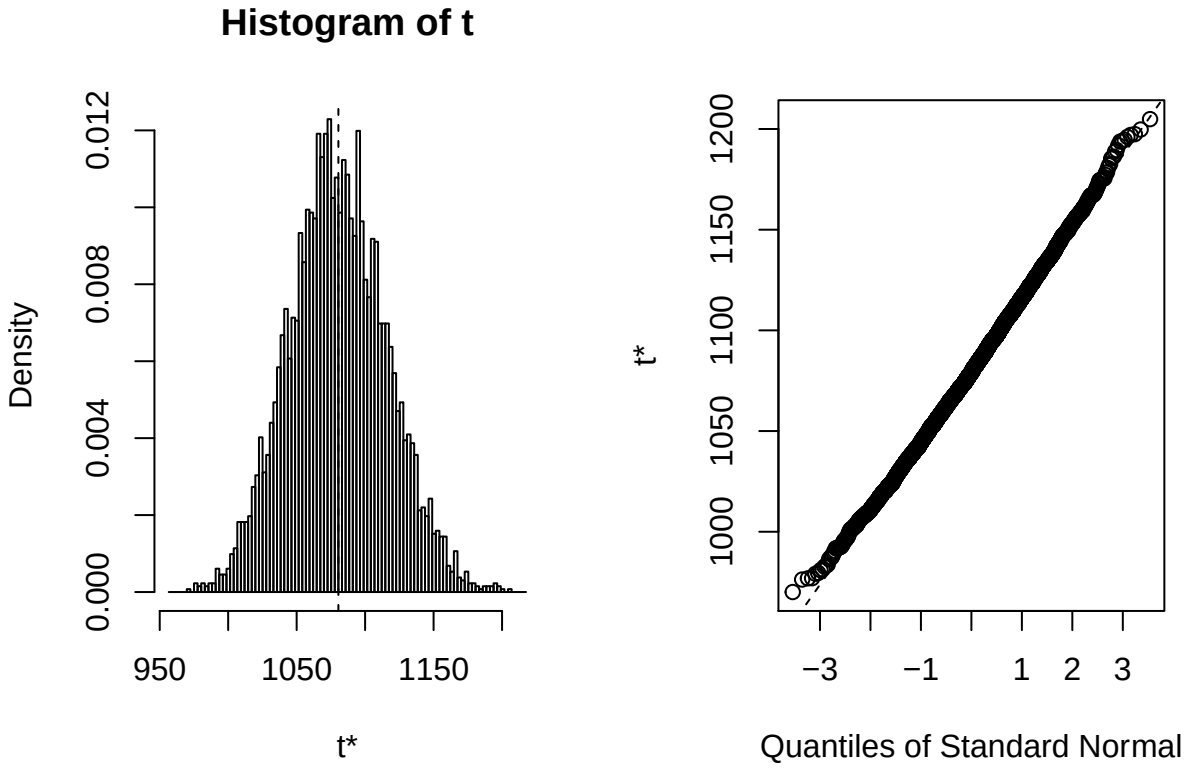
The above plot shows the distribution of the Price with the dot-vertical line to indicate the mean of price. The distribution seems to be gamma.

Subquestion 2-Bootstrap distribution estimation and CI

We are now going to estimate the distribution of the mean price of the house using bootstrap and Compute a 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and first order normal approximation.

The results from the bootstrap sampling are given below with the histogram of t and the quantiles showing a normal distribution

```
set.seed(12345)
library(boot)

#statistic function for boot function
meanfun <- function(dat, idx) mean(dat[idx], na.rm = TRUE)
#calculate bootstrap means
bot <- boot(prices$Price, statistic=meanfun, R=5000)
#plot boot object
plot(bot)
```




```
#
```

Next we calculate variance of the mean and the bias correction which is given by the formula :

$$T_1 := 2T(D) - \frac{1}{B}\sum_{i=1}^{B} T_i*$$

```
##Bias correction
bias_corr<-2*mean(prices$Price)-sum(bot$t)/5000
```

```
##variance of mean price
var_mean<-sum((bot$t-mean(bot$t))^2)/(5000-1)


cat("The bias correction is :",bias_corr)
```

```
## The bias correction is : 1080.654
```

```
cat("\n")
```

```
cat("The variance of the mean is :",var_mean)
```

```
## The variance of the mean is : 1261.819
```

Now we move to calculate the CI's with the BCa,percentile and first-order approximation

CI for mean Price with BCa

The output for the BCa CI is given below

```
#calculate bootstrap CI with BCa
bca_ci=boot.ci(bot, conf=0.95, type="bca",index=1)

print("The output of the Bca CI is :")
```

```
## [1] "The output of the Bca CI is :"
```

```
cat("\n")
```

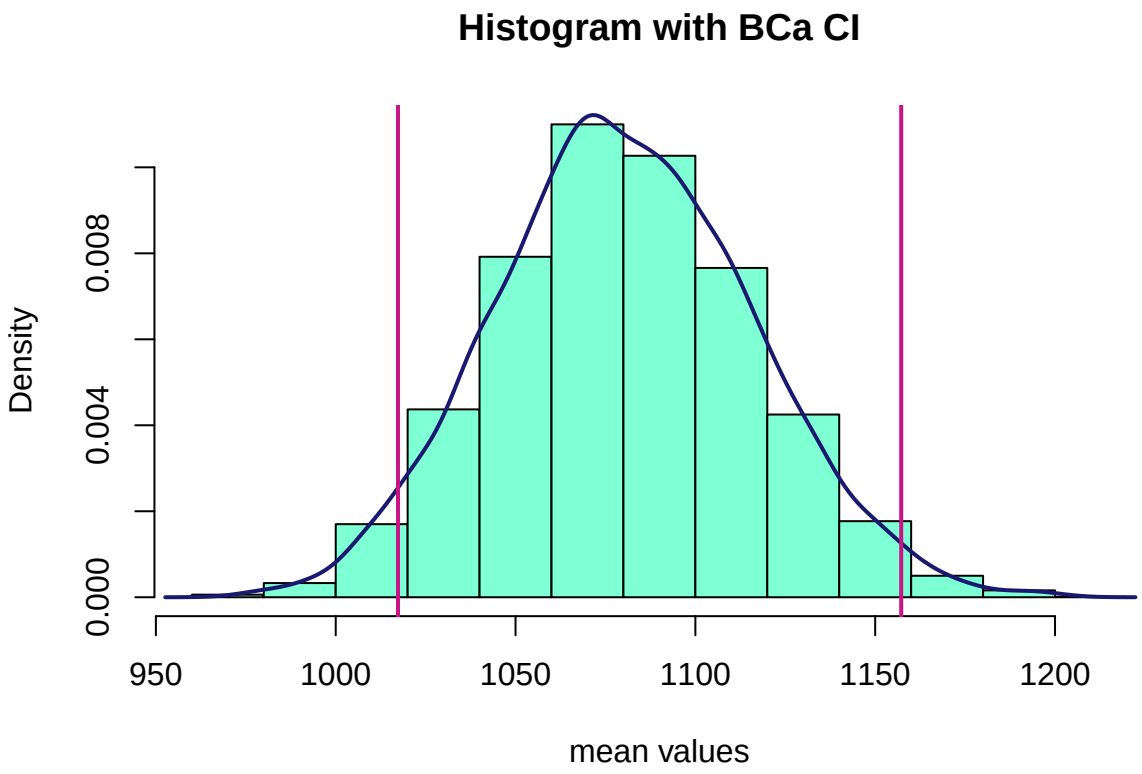
```
(bca_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bot, conf = 0.95, type = "bca", index = 1)
##
## Intervals :
## Level       BCa
## 95%   (1017, 1157 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with BCa

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with BCa method.

```
#take the 4th and 5th objects from bca that correspond to CI values
CI_bca=bca_ci$bca[ , c(4, 5)]
#plot histogram with bca intervals
hist(bot$t[,1],main = 'Histogram with BCa CI',xlab = 'mean values', col = 'aquamarine', prob = T)
lines(density(bot$t[,1]), col = 'midnightblue',lwd=2)
abline(v = CI_bca, col = 'mediumvioletred',lwd=2)
```



CI for mean Price with percentile

The output for the percentile CI are given below

```
perc_ci=boot.ci(bot,conf=0.95,type="perc",index = 1)

print("The output of the percentile CI is :")
```

```
## [1] "The output of the percentile CI is :"
```

```
cat("\n")
```

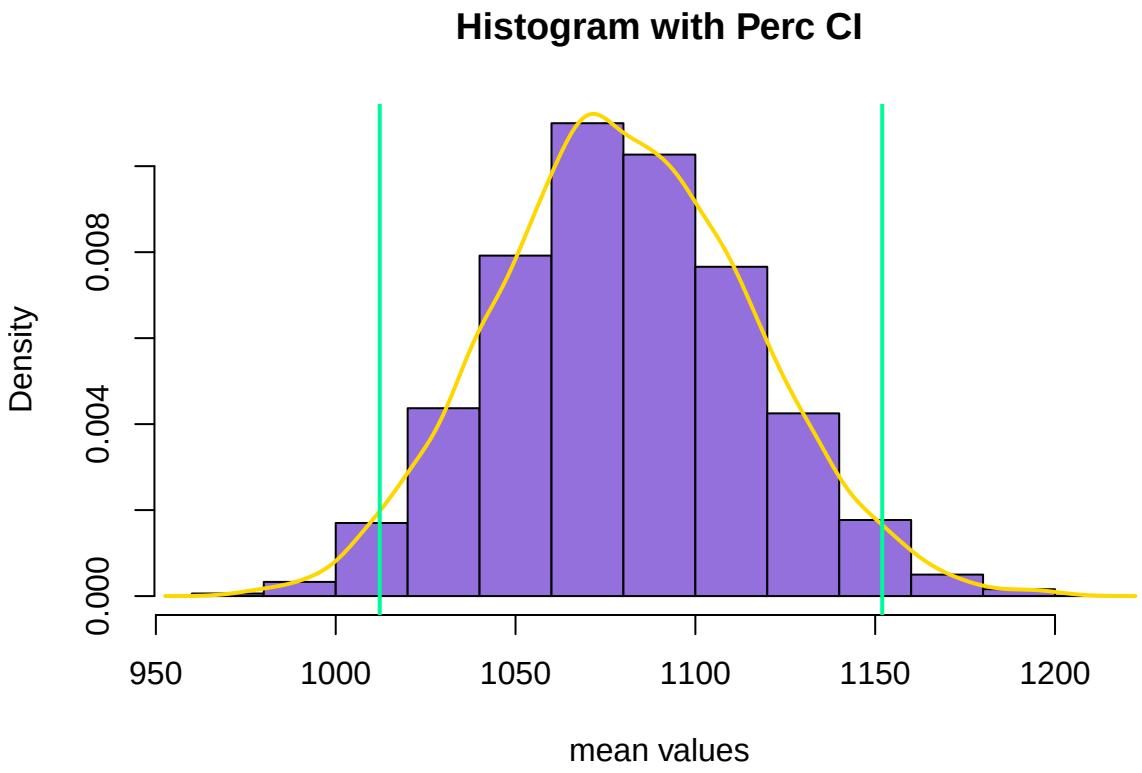
```
(perc_ci)
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bot, conf = 0.95, type = "perc", index = 1)
##
## Intervals :
## Level     Percentile
## 95%   (1012, 1152 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with percentile

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with percentile method.

```
CI_perc=perc_ci$percent[ , c(4, 5)]
#histogram with CI
hist(bot$t[,1],main = 'Histogram with Perc CI',xlab = 'mean values', col = 'mediumpurple', prob = T)
lines(density(bot$t[,1]), col = 'gold',lwd=2)
abline(v = CI_perc, col = 'mediumspringgreen',lwd=2)
```



CI for mean Price with first-order normalization

The output for the first-order normalization CI are given below

```
norm_ci=boot.ci(bot,conf=0.95,type = "norm",index=1)

print("The output of the first order normalization CI is :")
```

```
## [1] "The output of the first order normalization CI is :"
```

```
cat("\n")
```

```
(norm_ci)
```

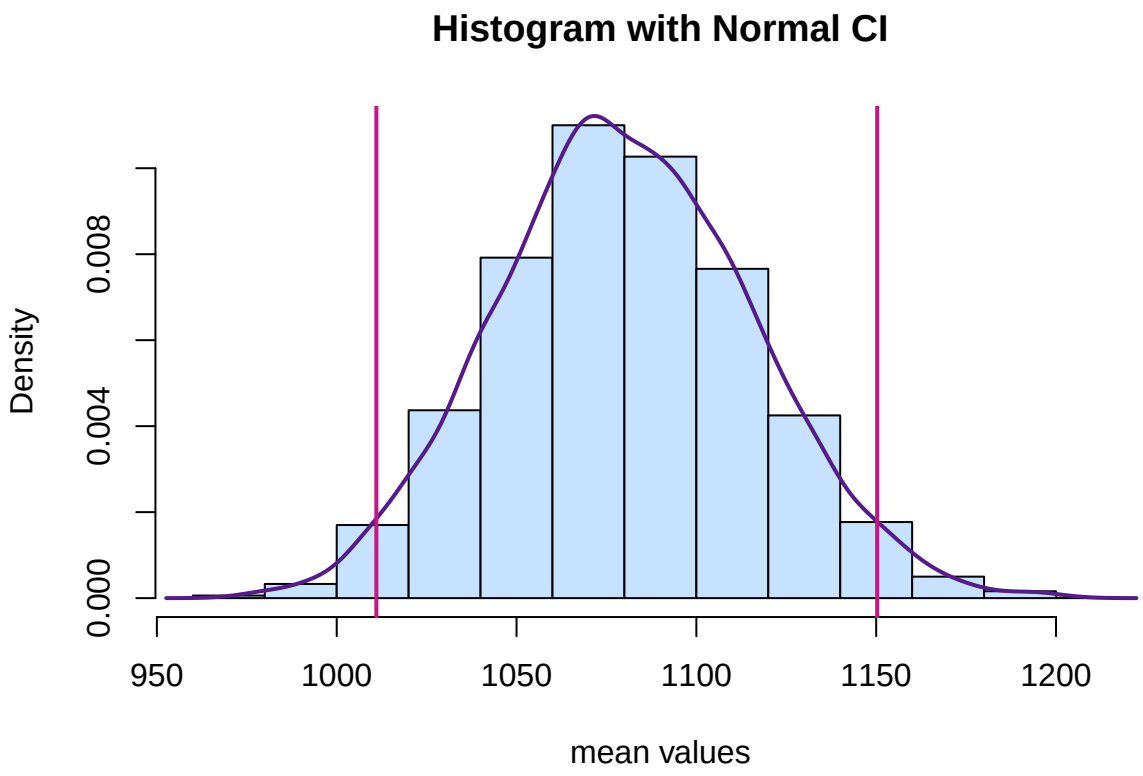
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
```

```
## boot.ci(boot.out = bot, conf = 0.95, type = "norm", index = 1)
##
## Intervals :
## Level      Normal
## 95%   (1011, 1150 )
## Calculations and Intervals on Original Scale
```

Histogram for mean Price with first order normalization

The plot shows the histogram of the t obtained from bootstrap and the vertical lines are the confidence intervals obtained with first order normalization method.

```
CI_norm=norm_ci$normal # we dont need to subscript here normal!!

hist(bot$t[,1], main = 'Histogram with Normal CI',xlab = 'mean values', col = 'slategray1', prob = T)
lines(density(bot$t[,1]), col = 'purple4',lwd=2)
abline(v = CI_norm, col = 'mediumvioletred',lwd=2)
```



Subquestion 3-Jacknife estimation

```
#jacknife the reaper function
#function to calculate the jacknife sample



# jk_func = function (x, theta, ...)
```

```
# {
#   call = match.call()
#   n = length(x)
#   u = rep(0, n)
#   for (i in 1:n) {
#     u[i] = theta(x[-i], ...)
#   }
#   theta.hat = theta(x, ...)
#   pseudo.values = n*theta.hat - (n-1)*u
#   theta.jack = mean(pseudo.values)
#   jack.se = sqrt(sum((pseudo.values - theta.jack)^2)/(n*(n-1)))
#   # jack.bias = theta.jack - theta.hat
#   jack.bias = (n-1)*(theta.hat - mean(u))
#   #
#   return(list(theta.hat = theta.hat,
#               theta.jack = theta.jack,
#               jack.bias = jack.bias,
#               # jack.var = jack.se^2,
#               jack.se = jack.se,
#               leave.one.out.estimates = u,
#               pseudo.values = pseudo.values,
#               call = call))
# }
```

In this part we are going to implement jacknife method and calculate The variance of the mean which is given from the above formula

$$\hat{V}ar[T(.)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} ((T_i*) - J(T))^2, \quad n = B$$

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

The results are given below

```
n <- length(prices$Price)
#the statistic function
theta <- median(prices$Price)
#apply the statistic function to the Price but without the current point-i
jk <- sapply(1 : n,function(i) mean(prices$Price[-i]))
#mean of jacknife
mean_jk <- mean(jk)
#bias jacknife
bias_jk <- (n - 1) * (mean_jk - theta)
#variance jacknife
var_jk <- (n - 1) * mean((jk - mean_jk)^2)


jk_dt<-data.frame(c("jacknife mean"=mean_jk,"jacknife bias"=bias_jk,"variane jacknife"=var_jk))
colnames(jk_dt)<-"value"
library(knitr)
kable(jk_dt)
```

	value
jacknife mean	1080.473
jacknife bias	11496.527
variane jacknife	1320.911

The obtained variance using Jackknife method is 1320.911 while using bootstrapping, the obtained value was 1261.819.

Subquestion 4-Compare CI

Finally, we are going to compare the results for the diffrent CI methods used.The table below summarises the results.

```
intervals<-c("(1017.324 ,1157.234)" ,"(1012.239, 1151.954)" ,"(1011.032, 1150.276)"  )
lengths<-c( (CI_bca[2]-CI_bca[1]),(CI_perc[2]-CI_perc[1]),(CI_norm[3]-CI_norm[2]) )
centers<-c( sum(CI_bca)/2,sum(CI_perc)/2,sum(CI_norm[2:3])/2 )

ci_dataset<-data.frame(intervals,lengths,centers)

kable(ci_dataset)
```

intervals	lengths	centers
(1017.324 ,1157.234)	139.9098	1087.279
(1012.239, 1151.954)	139.7150	1082.097
(1011.032, 1150.276)	139.2441	1080.654

As we can see the lengths of the intervals are quite the same but the center obtained by the first-order normalization is closer to original mean data which is 1080.473 thus this method is preferable.