732A90-Lab5-Report

Zhixuan Duan(zhidu838) Fengjuan Chen(fench417)

2/28/2020

Question 1

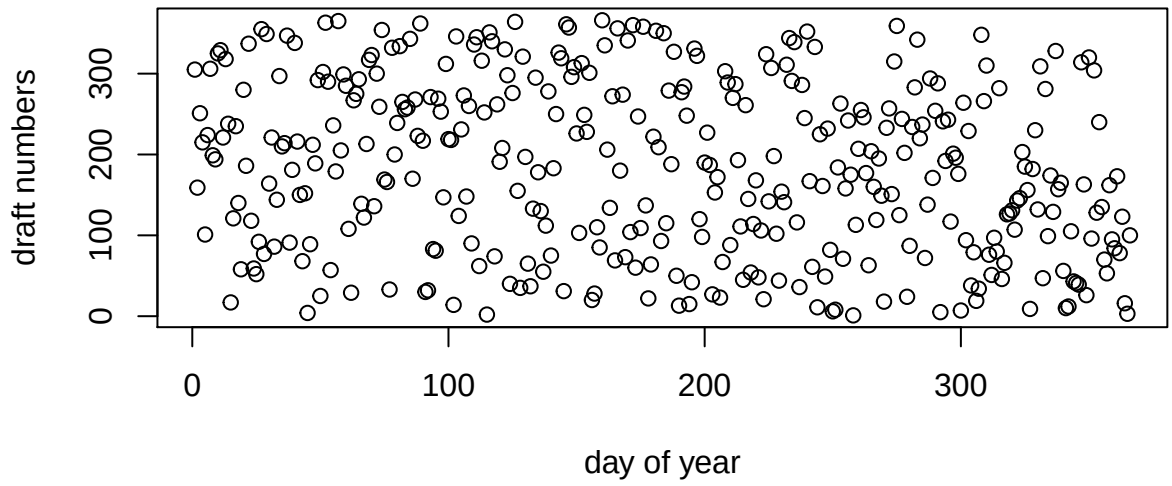
1. Scatterplot of Y versus X

We first import the data from file lottery.xls and make a sctterplot of Y=Draft_no versus X=Day_of_year.

```
library("readxl")

data1=read_excel("lottery.xls")
X=data1$Day_of_year
Y=data1$Draft_No
plot(X,Y,xlab = "day of year",ylab = "draft numbers")
```

From the scatterplot, we can roughly say that the lottery looks random.

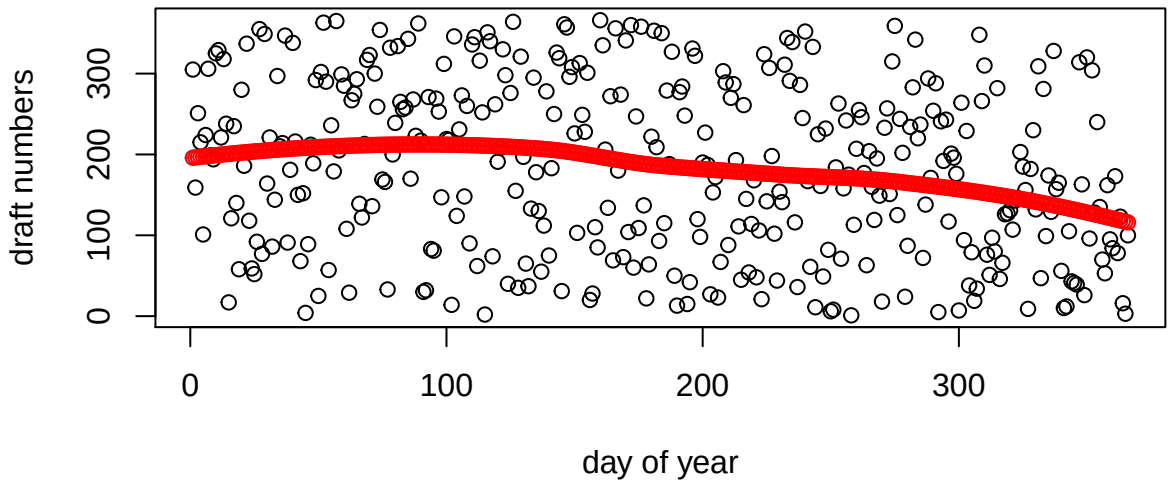


2. Use a loess smoother to compute an estimate

We first fit a loess model on the data file with Y as the response and X as the predictor. Then we use this loess model to predict the value of response Y. Finally, plot a curve line of predicted Y and predictor X in the previous graph.

```r
fit_loess=loess(Y~X,data = data1)
y_hat=predict(fit_loess,data1)
plot(X,Y,xlab = "day of year",ylab = "draft numbers")
points(X,y_hat,col="red")
```

From the plot, we can roughly say that the lottery looks random.



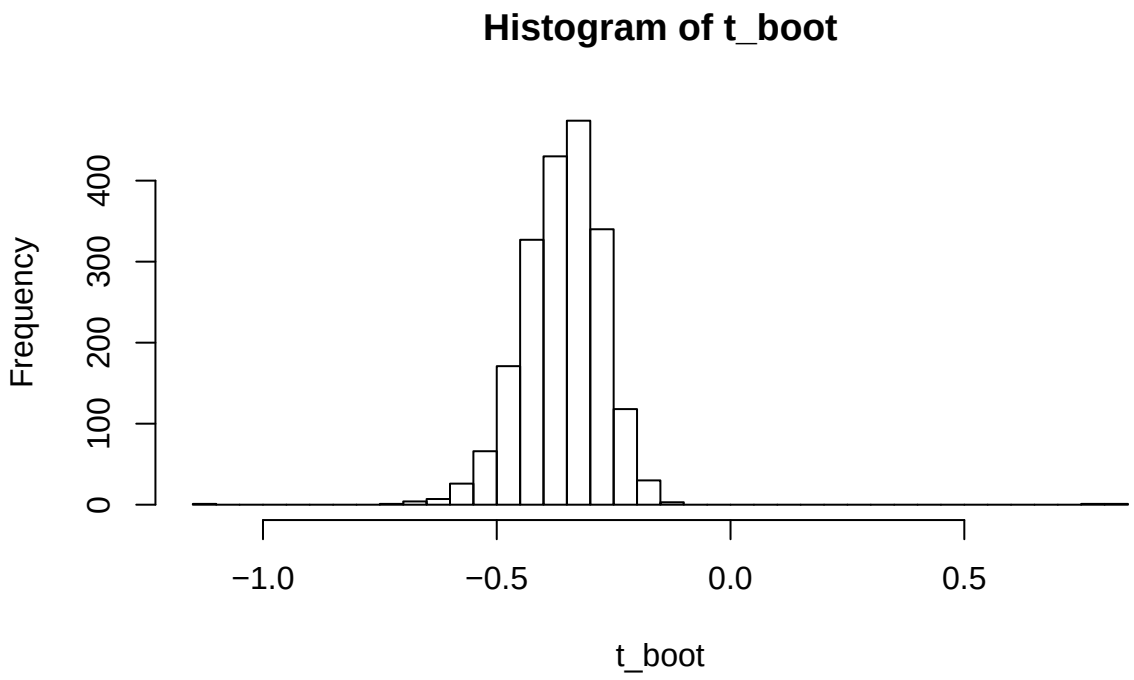
3. Estimate distribution of statistics T with bootstrap

We use a non-parametric bootstrap with B=2000 to estimate the statistics $T = \frac{\hat{Y}(X_b)-\hat{Y}(X_a)}{X_b-X_a}$, where $X_b = argmax_X \hat{Y}(X)$, $X_a = argmin_X \hat{Y}(X)$.

```r
pos_a=which(y_hat==min(y_hat))
pos_b=which(y_hat==max(y_hat))
X_a=X[pos_a]
X_b=X[pos_b]
t_original=(y_hat[pos_b]-y_hat[pos_a])/(X_b-X_a)
B=2000
t_boot=vector(length = B)
for (i in 1:B) {
  index=sample(1:366,length(X),replace = TRUE)
  X1=X[index]
  Y1=Y[index]
  data_new=data.frame(X1,Y1)
  model=loess(Y1~X1,data = data_new)
  y_hat1=predict(model,data_new)
  pos_a=which(y_hat1==min(y_hat1))
  pos_b=which(y_hat1==max(y_hat1))
  X1_a=X1[pos_a[1]]
  X1_b=X1[pos_b[1]]
```

```
  t_boot[i]=(max(y_hat1)-min(y_hat1))/(X1_b-X1_a)
}
```

The histogram of estimations of statistics T is shown below.



From the given information, we know that if the statistics T is significantly greater than zero, then there should be a trend in the data and the lottery is not random.

After using a non-parametric bootstrap with B=2000, we obtain 2000 estimations of this statistics T. These values are range from -1.1351715 to 0.8147785,not significantly greater than 0. Therefore, we can conclude that the lottery is random.

From these estimations of statistics T, we calculate the p-value by code

```
p_value=mean(t_boot>=0)
```

and obtain the p-value of the test :

```
## [1] 0.001
```

4. A permutation test with statistics T

The function that tests the hypothesis by using a permutation test with statistics T is shown below.

```
permutation <- function(B,X,Y){
  t_permute=vector(length = B)
  for (i in 1:B) {
    X2=sample(X,length(X))
```

```
    data_new=data.frame(X2,Y)
    model=loess(Y~X2,data = data_new)
    y_hat=predict(model,data_new)
    pos_a=which(y_hat==min(y_hat))
    pos_b=which(y_hat==max(y_hat))
    X2_a=X2[pos_a[1]]
    X2_b=X2[pos_b[1]]
    t_permute[i]=(max(y_hat)-min(y_hat))/(X2_b-X2_a)
  }
  p_value=mean(abs(t_permute)>=abs(t_original))
  return(p_value)
}
```

And the $p-value$ is

```
## [1] 0.1415
```

So you should not reject the null hypothesis. You should really give some interpretation...

5. Estimate the power of the test constructed in Step 4

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

```
gene_new_data <- function(alpha,X=X){
  n=length(X)
  beta=rnorm(n,mean = 183,sd=10)
  Y=vector(length = n)
  for (i in 1:n) {
    Y[i]=max(0,min(alpha*X[i]+beta[i],366))
  }
  return(Y)
}
Y3=gene_new_data(alpha = 0.1,X=X)
```

(b) Then plug these data into the permuation test with B=200 and call the function we defined in Step 4. We obtain the p-value with $\alpha = 0.1$ below.

```
p_value3=permutation(B=200,X,Y3)
```

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

Since p-value is equal to 0, we reject the null hypothesis that lottery is random.

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

```
alpha=seq(0.2,10,0.1)
p_value_alpha=vector(length = length(alpha))

for (i in alpha) {
  Y4=gene_new_data(alpha = i,X=X)
  p_value_alpha[i]=permutation(B=200,X=X,Y=Y4)
}
p_value_alpha
```

Then we get these p-values corresponding to $\alpha = 0.2, 0.3, ..., 10$:

```
##  [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
## [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
## [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 0 0 0
```

All the p-values for the generated dataset with $\alpha = 0.1, 0.2, ..., 10$ are zero, so we reject all the null hypothesis for these datasets.

And we know the generated dataset is an obviously non-random and we correctly reject them all. Then the power is equal to 1. Therefore, we can say that the quality of this test statistics T is good.

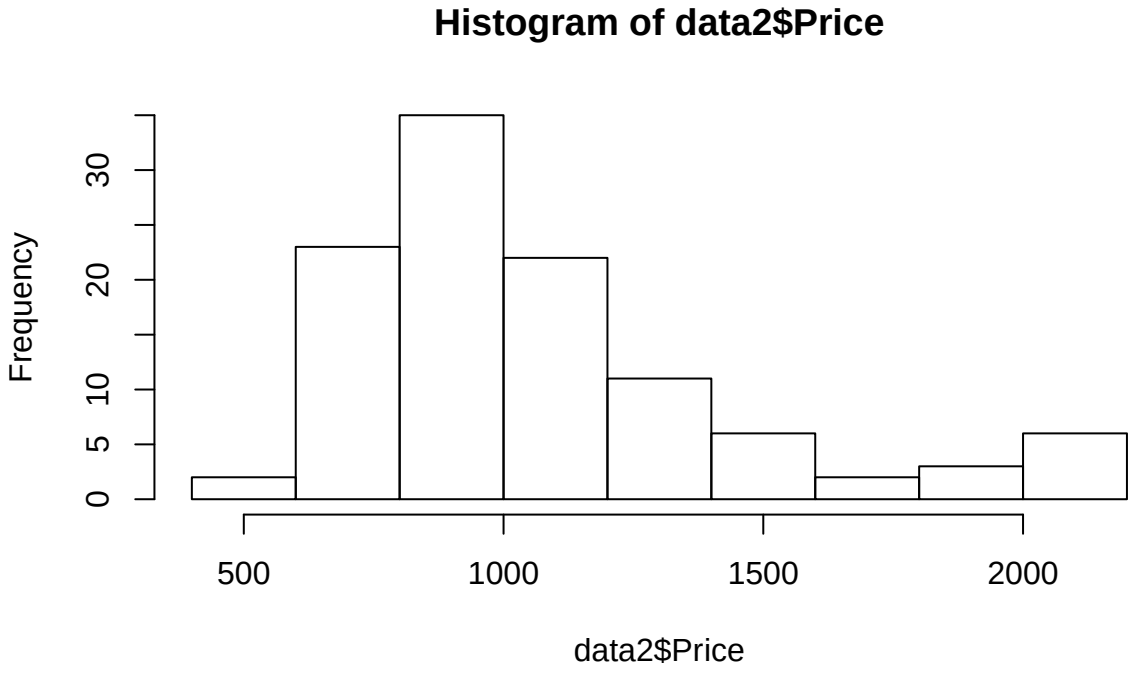
You can only conclude that it's good when the data is linear/close to linear. Feed it a sin(x) curve and this test will fail on that.

Question 2 Bootstrap, jackknife and confidence intervals

1. Plot the histogram of Price

We import the data from file prices1.xls and plot the histogram of Price.

```
data2=read_excel("prices1.xls")
hist(data2$Price)
```



This histogram looks like the gamma distribution.

The mean price is

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

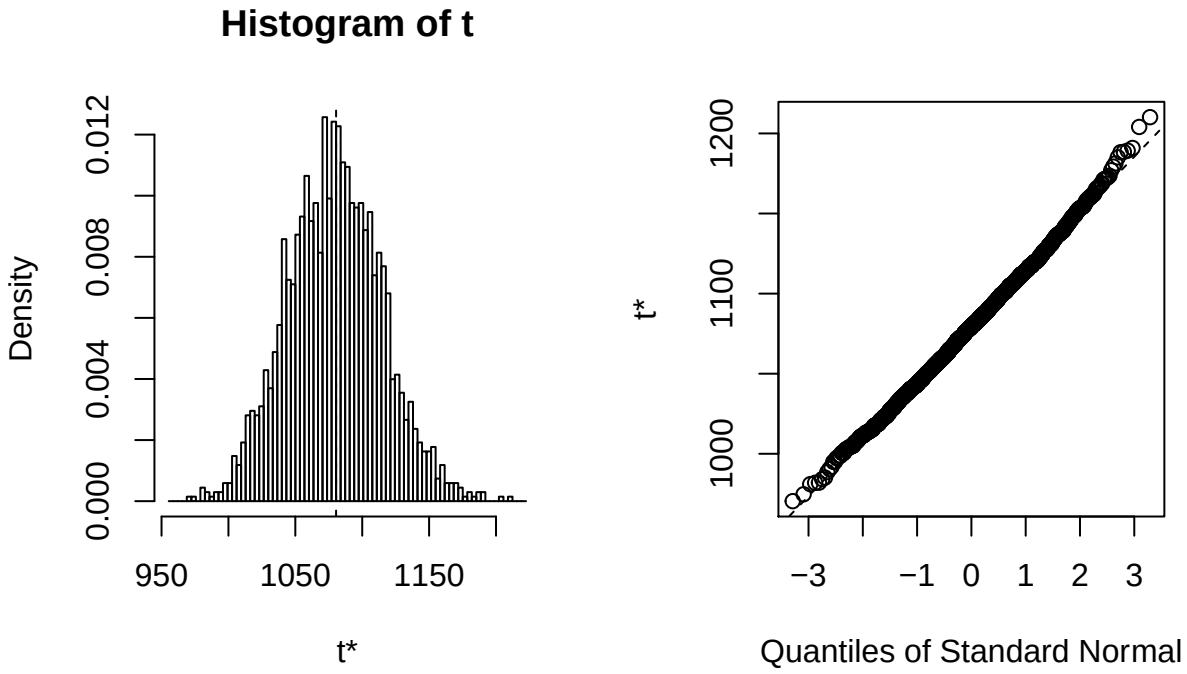
2. bootstrap and 95% confidence interval

We estimate the distribution of the mean price of the house using bootstrap with 2000 times sampling.We call the boot() function in library boot.

```
library(boot)
f <- function(data,index){
  data3 <- data[index,]
  return(mean(data3$Price))
}
res <- boot(data2,f,R=2000)
```

After bootstrap, we obtain two plots from the boot() function.

```
plot(res)
```



From the results of bootstrap, we calculate the bias corrected estimator by formula

$T_{bias-corrected} = 2 * T(D) - \frac{1}{B}\sum_{i=1}^{B} T_i^*$ where B is the sample number, $T(D)$ is the statistics T from the original data and T_i^* is the statistics from the sampled data D_i^*, $i \in [1, B]$. Then we calculate the variance of the mean price from the bootstrap results using the formula

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

The values of the bias-correction, variance and standard deviation of the mean price are:

```
## [1] 1081.444
```

```
## [1] 1246.958
```

```
## [1] 35.31229
```

These three values are consistent with the result returned by function c(res).

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data2, statistic = f, R = 2000)
##
##
## Bootstrap Statistics :
##     original     bias    std. error
## t1* 1080.473 -0.9710045    35.31229
```

The 95% confidence interval for the mean price using bootstrap percentile, bootstrap BCa, and the first-order normal approximation are calculated by function boot.ci(). The results are shown below.

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = res)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1012, 1151 )   (1009, 1148 )
##
## Level     Percentile            BCa
## 95%   (1013, 1152 )   (1015, 1157 )
## Calculations and Intervals on Original Scale
```

3. Jackknife method

We sample from the original data B times (B<=n, n is the number of observations in the original data). To sample the B datasets, we delete an observation i from original dataset each time where $i \in [1 : B]$. After using this jackknife method, we can estimate the uncertainty of estimation by variance of the mean price. We use the formalua

$\hat{Var}[T(\cdot)] = \frac{1}{n(n-1)} \sum_{i=1}^{n} (T_i^* - J(T))^2$ where $T_i^* = nT(D) - (n-1)T(D_i^*)$ and $J(T) = \frac{1}{n} \sum_{i=1}^{n} T_i^*$

to estimate the variance of the mean price.The code for jackknife method is shown below.

```
n=nrow(data2)
t_star=vector(length = n)
for (i in 1:n) {
  price_new=data2$Price[-i]
  t_star[i]=n*mean(data2$Price)-(n-1)*mean(price_new)
}
var_jack=sum((t_star-mean(t_star))^2)/(n*(n-1))
sd_jack=sqrt(var_jack)
```

Then we compare the value with the value from bootstrap method in table 1.

Table 1: Variance and sd in two methods

	Variance	Standard_deviationc
bootstrap	1246.958	35.31229
jackknife	1320.911	36.34434

4. Compare the confidence intervals

By using boot() function and boot.ci() function, we obtain the estimated mean price and several confidence intervals of it.

These confidence intervals are the first order normal approximation, the basic bootstrap interval, the bootstrap percentile interval, and the adjusted bootstrap percentile (BCa) interval. We compare these intervals in table 2.

Table 2: comparison with different CIs

	lower_bound	upper_bound	length	middle	estimated
Normal	1012.233	1150.655	138.4216	1081.444	1081.444
Basic	1009.080	1148.378	139.2985	1078.729	1081.444
Percentile	1012.567	1151.866	139.2985	1082.216	1081.444
BCa	1015.368	1157.221	141.8532	1086.295	1081.444

From the results of these confidence intervals and the estimated mean, we find that BCa has the shortest length of confidence interval and all these confidence intervals contain the estimated mean price. Furthermore, the Normal confidence interval has the best location of the estimated mean price because the estimated value is nearest to the center of the Normal confidence interval.

Appendix

```
set.seed(12345)

## Question 1


## 1. scatterplot of Y versus X

library("readxl")
data1=read_excel("lottery.xls")
X=data1$Day_of_year
Y=data1$Draft_No

plot(X,Y,xlab = "day of year",ylab = "draft numbers")


## 2. Use a loess smoother to compute an estimate

fit_loess=loess(Y~X,data = data1)
y_hat=predict(fit_loess,data1)
plot(X,Y,xlab = "day of year",ylab = "draft numbers")
points(X,y_hat,col="red")


## 3. Estimate distribution of statistics T with bootstrap

pos_a=which(y_hat==min(y_hat))
pos_b=which(y_hat==max(y_hat))
X_a=X[pos_a]
X_b=X[pos_b]
t_original=(y_hat[pos_b]-y_hat[pos_a])/(X_b-X_a)
B=2000
t_boot=vector(length = B)

for (i in 1:B) {
  index=sample(1:366,length(X),replace = TRUE)
  X1=X[index]
  Y1=Y[index]
  data_new=data.frame(X1,Y1)
  model=loess(Y1~X1,data = data_new)
  y_hat1=predict(model,data_new)
  pos_a=which(y_hat1==min(y_hat1))
  pos_b=which(y_hat1==max(y_hat1))
  # if there are more than one values that have the
  #   maxmimum values, then randomly select the first position
  X1_a=X1[pos_a[1]]
  X1_b=X1[pos_b[1]]
  t_boot[i]=(max(y_hat1)-min(y_hat1))/(X1_b-X1_a)
}

p_value=mean(t_boot>=0)
hist(t_boot,breaks = 50)
```

```r
## 4. A permutation test with statistics T

permutation <- function(B,X=X,Y=Y){
  t_permute=vector(length = B)
  for (i in 1:B) {
    X2=sample(X,length(X))
    data_new=data.frame(X2,Y)
    model=loess(Y~X2,data = data_new)
    y_hat=predict(model,data_new)
    pos_a=which(y_hat==min(y_hat))
    pos_b=which(y_hat==max(y_hat))
    X2_a=X2[pos_a[1]]
    X2_b=X2[pos_b[1]]
    t_permute[i]=(max(y_hat)-min(y_hat))/(X2_b-X2_a)
  }
  p_value=mean(abs(t_permute)>=abs(t_original))
  return(p_value)
}

p_value2=permutation(B=2000,X=X,Y=Y)

## 5. Estimate the power of the test constructed in Step 4

## generate new dataset
gene_new_data <- function(alpha,X=X){
  n=length(X)
  beta=rnorm(n,mean = 183,sd=10)
  Y=vector(length = n)
  for (i in 1:n) {
    Y[i]=max(0,min(alpha*X[i]+beta[i],366))
  }
  return(Y)
}

Y3=gene_new_data(alpha = 0.1,X=X)

p_value3=permutation(B=200,X,Y3)
alpha=seq(0.2,10,0.1)
p_value_alpha=vector(length = length(alpha))
for (i in alpha) {
  Y4=gene_new_data(alpha = i,X=X)
  p_value_alpha[i]=permutation(B=200,X=X,Y=Y4)
}
p_value_alpha
```

```r
# Question 2 Bootstrap, jackknife and confidence intervals

## 1. Plot the histogram of Price
data2=read_excel("prices1.xls")

hist(data2$Price)
mean(data2$Price)

## 2. Use bootstrap to estimate the distribution of the mean price

library(boot)
f <- function(data,index){
  data3 <- data[index,]
  return(mean(data3$Price))
}
res <- boot(data2,f,R=2000)
ci=boot.ci(res,type = "all")
plot(res)
print(ci)

bias_corr=2*mean(data2$Price)-sum(res$t)/1000
c(res)
## the variance of the mean price
var_meanprice=(sum((res$t-mean(res$t))^2))/(1000-1)
sd_meanprice=sqrt(var_meanprice)


## 3. jackknife method
n=nrow(data2)
t_star=vector(length = n)
for (i in 1:n) {
  price_new=data2$Price[-i]
  t_star[i]=n*mean(data2$Price)-(n-1)*mean(price_new)
}

var_jack=sum((t_star-mean(t_star))^2)/(n*(n-1))
sd_jack=sqrt(var_jack)


## 4. compare the confidence intervals

con_int=data.frame(c(ci$normal[2],ci$basic[4],
                     ci$percent[4],ci$bca[4]),
                   c(ci$normal[3],ci$basic[5],
                     ci$percent[5],ci$bca[5]))
colnames(con_int)=c("lower_bound","upper_bound")
rownames(con_int)=c("Normal","Basic","Percentile","BCa")
con_int$length=con_int$upper_bound-con_int$lower_bound
con_int$middle=con_int$lower_bound+con_int$length/2
con_int$estimated=rep(bias_corr,4)
```