Computational Statistics Lab05

Min-chun Shih(shimi077), Saewon Jun(saeju204)

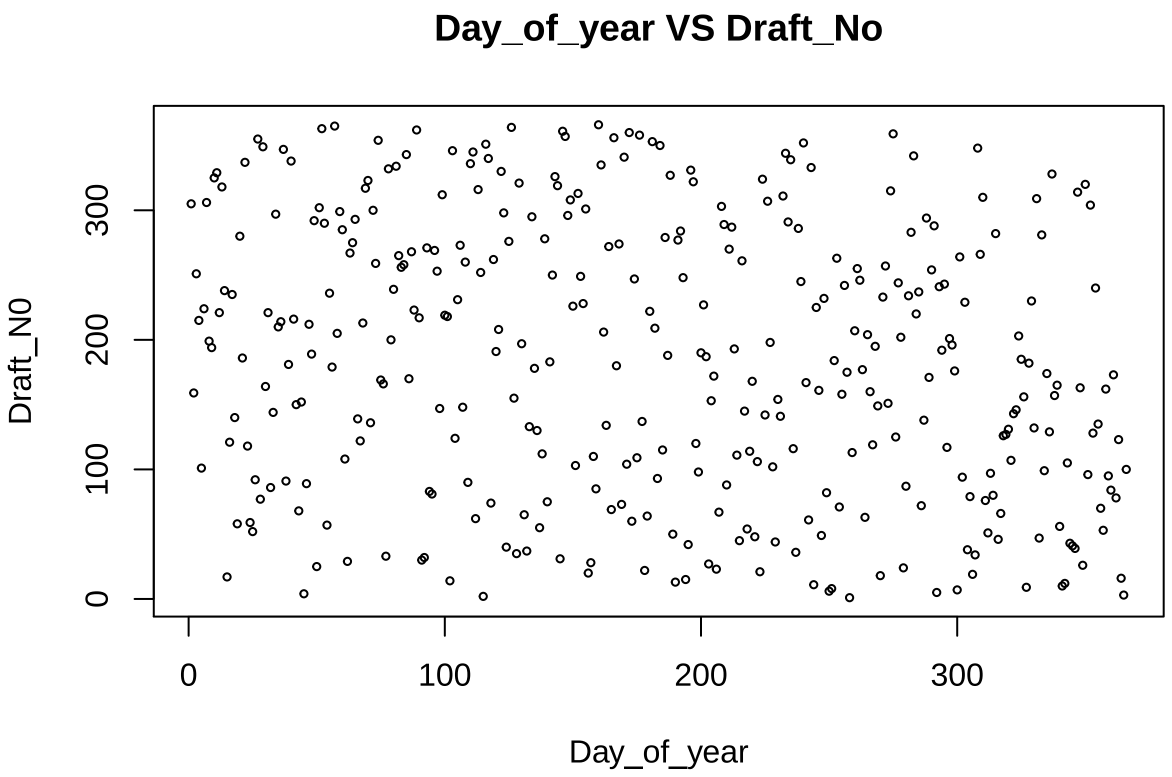
2019 2 19

Question 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.

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

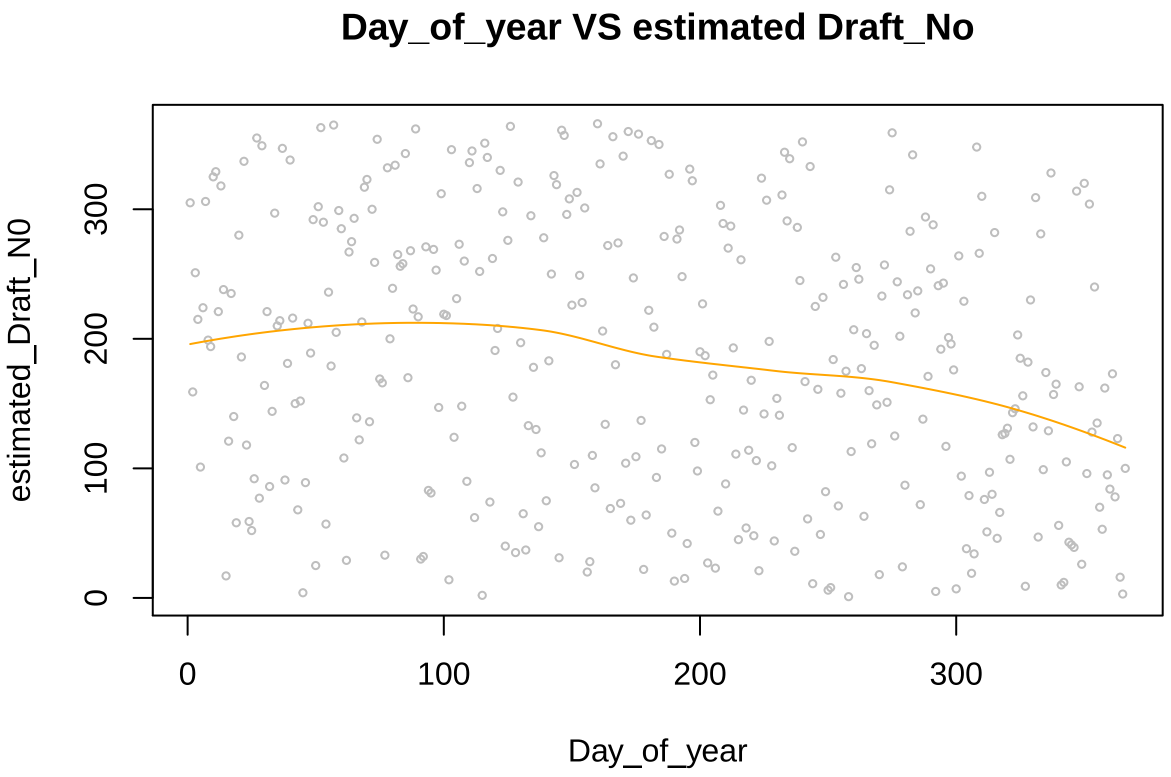
```
lottery <- read.csv2("lottery.csv")
X <- lottery$Day_of_year
Y <- lottery$Draft_No

plot(X,Y,main="Day_of_year VS Draft_No", xlab="Day_of_year", ylab="Draft_NO",
     cex=0.5)
```



The plot seems randomly distributed as you can't find any pattern.

1-2 Compute an estimate $\hat{Y}$ of the expected reponse as a function of X by using a loess smoother, put the curve $\hat{Y}$ versus X in the previous graph and state again whether the lottery looks random.

```r
model <- loess(Y ~ X, method="loess")
plot(X,Y, main="Day_of_year VS estimated Draft_No",
     xlab="Day_of_year", ylab="estimated_Draft_NO", cex=0.5, col="grey")
lines(X, model$fitted, col="orange")
```



As you can see from the plot above, we were able to find the certain pattern among the observations, so it is hard to say the lottery is random.

1-3 To check whether the lottery is random, it is reasonable to use test statistics :

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

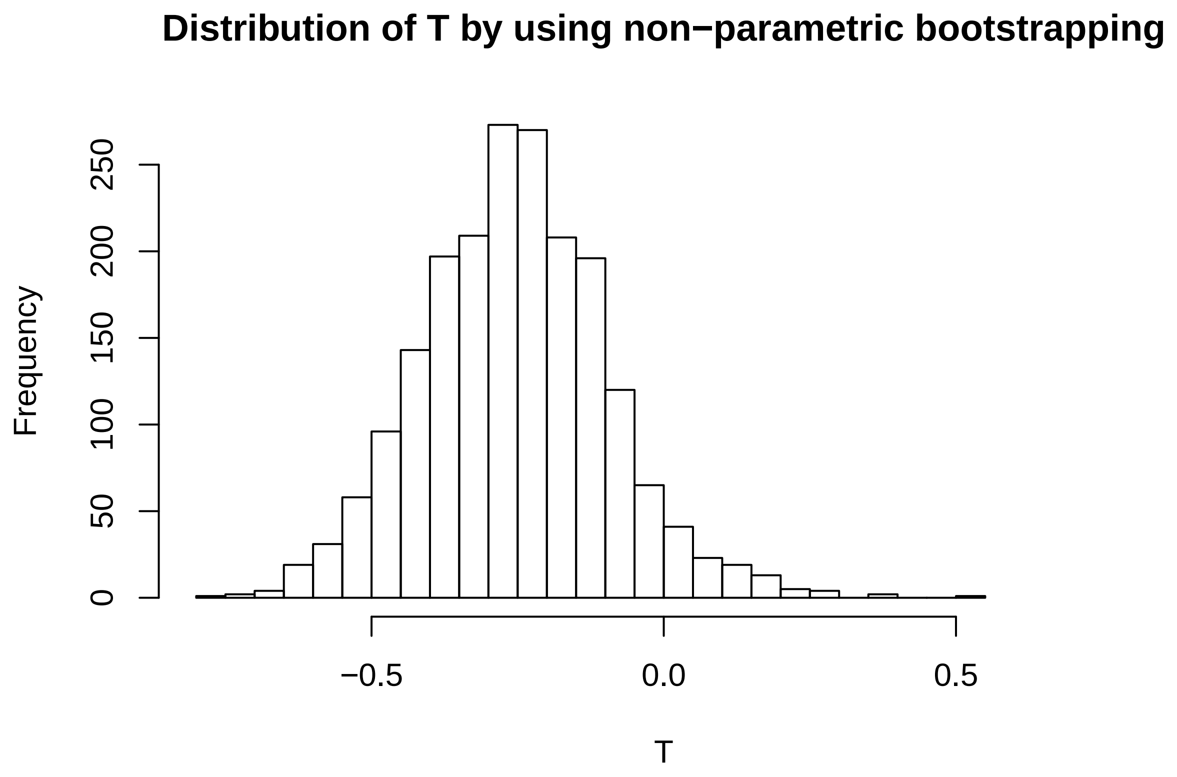
If this value is significantly greater than zero, then there should be a trend in the data and lottry is not random. Estimate the distribution of T by using a **non-parametric bootstrap** with B=2000 and comment whether the lottery is random or not. What is the p-value of the test?

By using a non-parametric bootstrap with B = 2000, the distribution of T-statistic could be estimated. For the every iteration, the index of 366 rows is sampled with replacement and test statistic t is calculated based on the above formula.

Additional explanation about bootstrapping: There are three way of doing bootstrapping - 1)Nonparametric(resampling) 2)Semiparametric(Adding noise) 3)Parametric(simulation) 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. - Nonparametric (resampling) bootstrap : **In the nonparametric bootstrap a sample of the same size as the data is take from the data**

with replacement. What does this mean? It means that if you measure 10 samples, you create a new sample of size 10 by replicating some of the samples that you've already seen and omitting others. At first this might not seem to make sense, compared to cross validation which may seem to be more principled. However, it turns out that this process actually has good statistical properties. - Parametric bootstrap : Parametric bootstrapping assumes that the data comes from a known distribution with unknown parameters. (For example the data may come from a Poisson, negative binomial for counts, or normal for continuous distribution.) You estimate the parameters from the data that you have and then you use the estimated distributions to simulate the samples.

```
set.seed(0621)

T <- numeric()
dat <- cbind(X,Y)

for (i in 1:2000) {
        boot <-sample(length(X), replace=TRUE)
        data1 <- dat[boot,]
        #by doing dat[boot,] we are doing resampling on our given data set.

        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$fitted[which.min(data1[,2])]
        fitted_X_b <- model2$fitted[which.max(data1[,2])]

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


pval <- mean(T>=0)


hist(T, breaks=30, xlim=c(-0.8,0.8),
     main="Distribution of T by using non-parametric bootstrapping")
```



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

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

```r
print(pval)
```

```
## [1] 0.054
```

Obtained plot above is the estimated distribution of T. The distribution seems to follow the normal distribution. 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 0.054, concluding that there is no trend at all in the data.

(Your test statistic is positive if there is a positive correlation on the data (curve going up) and negative otherwise. By reporting a p-value of around 0.5, you are concluding that there is no trend at all in the data, which goes against what your plot above was showing.)

1-4 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

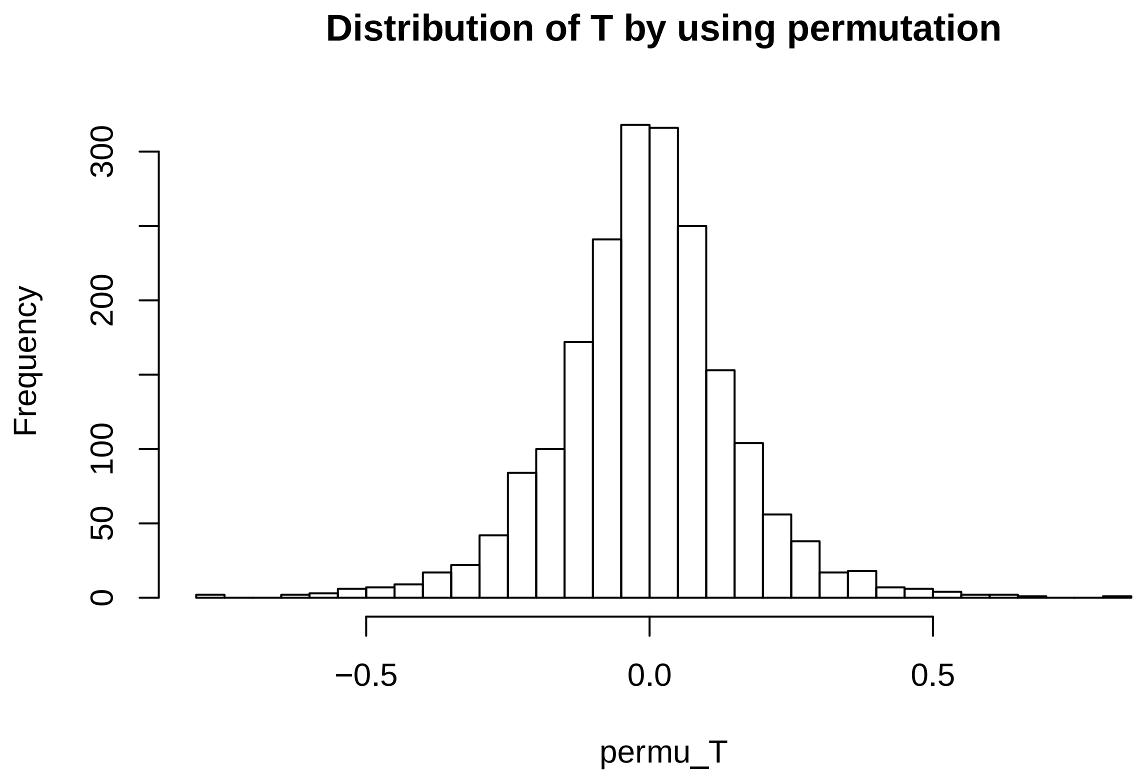
```r
data <- cbind(X,Y)
X_a <- data[which.min(data[,2])]
X_b <- data[which.max(data[,2])]

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

```r
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")
```



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

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

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

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

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

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*:

```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)

}
```

1-5 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)

new_Y <- function(alpha){

        X <- lottery$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=lottery$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=lottery$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
```

From the result above, we can see that 98% of the cases rejected 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.

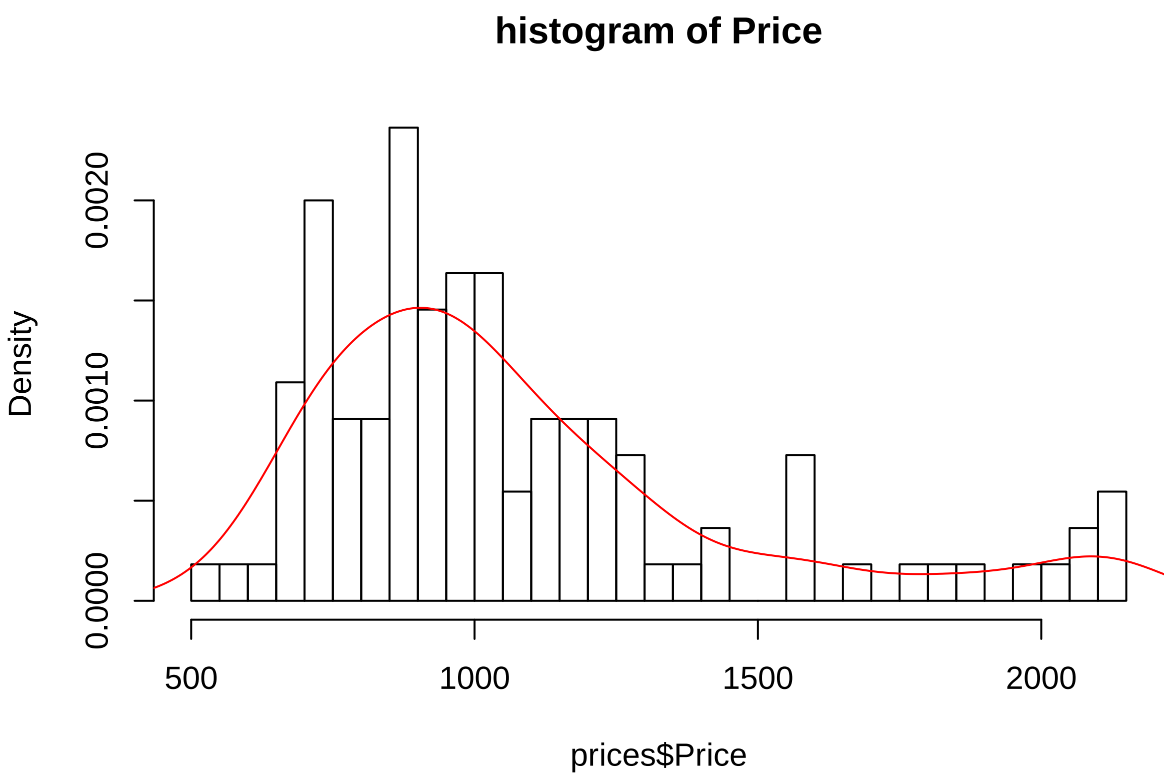
It is important to note that this power is for a simple linear test statistic, and things might change on other cases.

Question 2:Bootstrap, jackknife and confidence intervals

The data you are going to continue analyzing is the database of home priced in Albuquerque. 1993. The variables present are *Price*, *SqFt*(the area of a house), *FEATS*(the number of features such as dishwasher, refrigerator, and so on), *Taxes*(annual taxes paid for the house).

2-1.Plot the histogram of *Price*. Does it remind any conventional distribution? Compute the mean price.

```
prices <- read.csv2("prices1.csv")
hist(prices$Price, main="histogram of Price", breaks=30, freq=F)
lines(density(prices$Price), col="red")
```



```
mean(prices$Price)
```

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

The plot above seems like gamma distriution(with left skewed long right tail), or could be the mixture model.

2-2 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 corrected estimator is :

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

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

boot.fn <- function(data,index){
        d <- data[index]
        res <- mean(d)
}
```

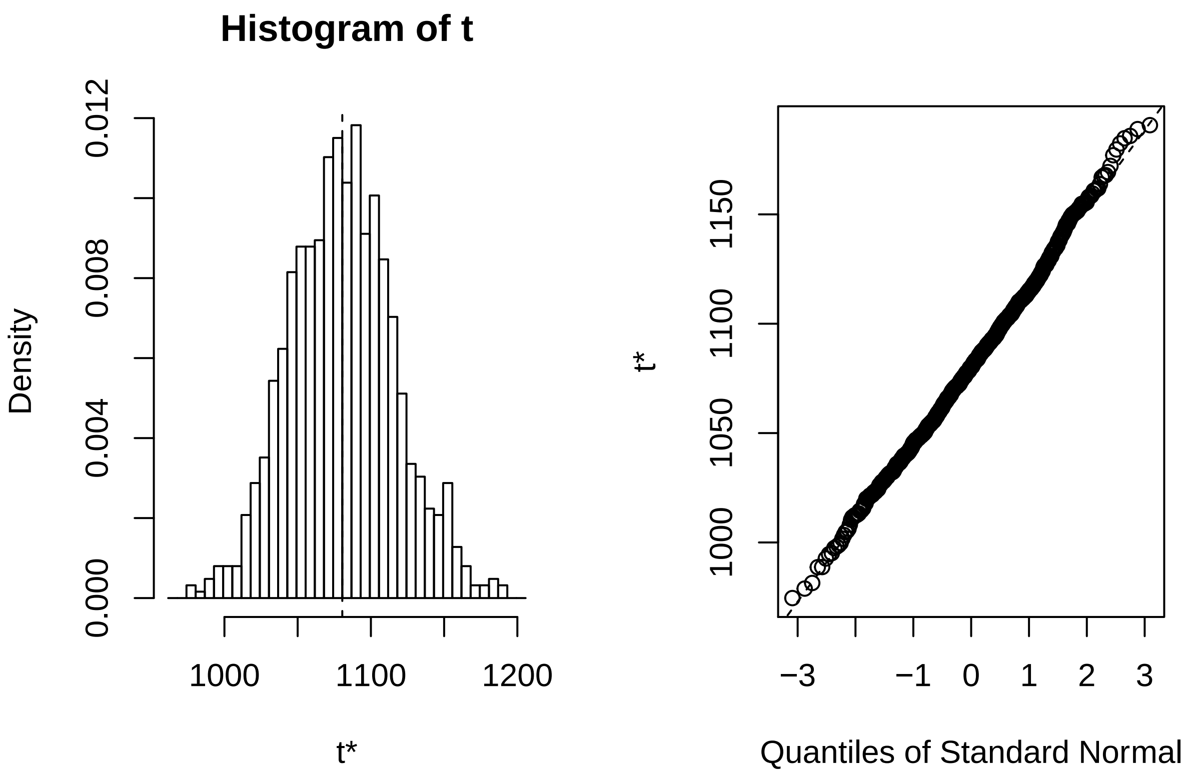
```r
boot.result <- boot(data=prices$Price, statistic=boot.fn, R=1000)
bias_cor <- 2*mean(prices$Price)-mean(boot.result$t)


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

```
## $bias_correction
## [1] 1080.22
##
## $variance_of_the_mean_price
## [1] 1290.965
```

```r
boot.result
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = prices$Price, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original     bias    std. error
## t1* 1080.473 0.2524091    35.92783
```

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



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

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.result)
##
## Intervals :
## Level      Normal              Basic
## 95%   (1010, 1151 )   (1006, 1148 )
##
## Level     Percentile            BCa
## 95%   (1013, 1155 )   (1017, 1158 )
## Calculations and Intervals on Original Scale
```

The distribution of the mean price of the house using bootstrap seems follow normal distribution.

2-3.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 <- length(prices$Price)

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

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

print("Jacknife Variance:")
```

```
## [1] "Jacknife Variance:"
```

```r
print(var_T)
```

```
## [1] 1320.911
```

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.

2-4. Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.

```r
library(knitr)
library(kableExtra)

intervals <- c("(1010,1151)","(1006,1148)","(1013,1155","(1017,1158)")
length <- c(1151-1010,1148-1006,1155-1013,1158-1017)
Center_of_interval <- c((1151+1010)/2,(1148+1006)/2,(1013+1155)/2,(1017+1158)/2)

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

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

	Confidence interval	Length of interval	Center of interval
Normal	(1010,1151)	141	1080.5
Basic	(1006,1148)	142	1077.0
Percentile	(1013,1155	142	1084.0
BCa	(1017,1158)	141	1087.5

The length of intervel doesn't vary much among four different methods. Considering the obtained mean from original data is 1080.473, Normal method seems to be reasonable in this case which has closest center of interval.