Lab5_report

Simon Jorstedt & Siddhesh Sreedar

2023-12-05

Question 1

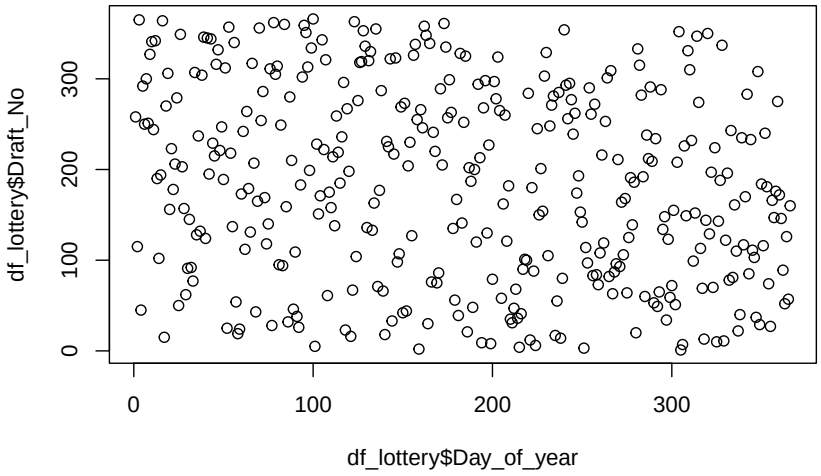
```
# Setup packages and data

library(magrittr)
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
```

```
library(boot)

# Read lottery data
df_lottery <- read.table("lottery.txt", col.names = c("Draft_No", "Day_of_year", "Letter_rank"))
```

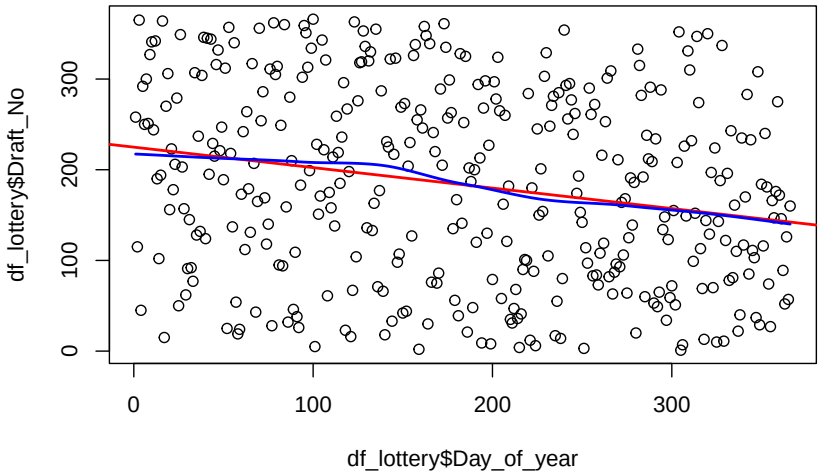
Question 1.1

Fig 1.1: Draft Num. vs Day of year



In figure 1.1, there is no immediate pattern that emerges. The spread of draft numbers appears roughly uniform.

Fig 1.2: Draft Num. vs Day of year with



In Figure 1.2, we fit a classical linear regression line, and a Local Polynomial Regression model (loess) on the data from Figure 1.1. We see that the linear regression line and the loess curve behave very similarly. They are both almost horizontal, which imples a low correlation between the variables. The slight negative slopes implies that higher draft numbers are ever so slightly more likely for the early days of the years. The lottery is surely random, but is it truly uniform?

Question 1.3

Now we wish to investigate whether or not the true values Y truly are uniformly random. We will do so by drawing 2000 bootstrap samples from the fitted values $\hat{Y}$, and calculating the statistic

$$S = \frac{1}{n}\sum_{i=1}^{n} |\hat{Y}_i - \bar{Y}|$$

where $\hat{Y}_i$ is the i:th predicted value of a particular Bootstrap sample, and $\bar{Y}$ is the mean of the true observations that correspond to that particular Bootstrap sample. We have introduced the scale factor $1/n$ in the statistic because it makes it more easily interpretable.

```
# Question 1.3

# Create an S-statistic function
statistic_1 <- function(data, indexes){
  data <- data[indexes]
  S <- (1/length(indexes)) * sum(abs(data-mean(df_lottery$Day_of_year[indexes])))
}

# Perform a test on the fitted data
result <- boot(loess_curve$fitted, statistic = statistic_1, R=2000)

print(boot.ci(result))
```

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

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = result)
##
## Intervals :
## Level      Normal              Basic
## 95%   (21.30, 24.72 )   (20.75, 24.35 )
##
## Level     Percentile            BCa
## 95%   (22.61, 26.21 )   (21.98, 24.52 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

```
cat("\nS estimate:", result$t0)
```

```
##
## S estimate: 23.47958
```

```
cat("\nEstimated p-value:", mean(result$t >= result$t0))
```

```
##
## Estimated p-value: 0.706
```

In the chunk above we carried out the Bootstrap sampling using the package `boot`. The estimate of S lies within all the confidence intervals, and the estimated p-value is not significant under a standard confidence level of 0.05. All this combined implies that the lottery at least is not significantly un-uniform and thus unfair. We can thus not reject the null hypothesis that the lottery is "random" (although the instructions surely must mean *uniform*).

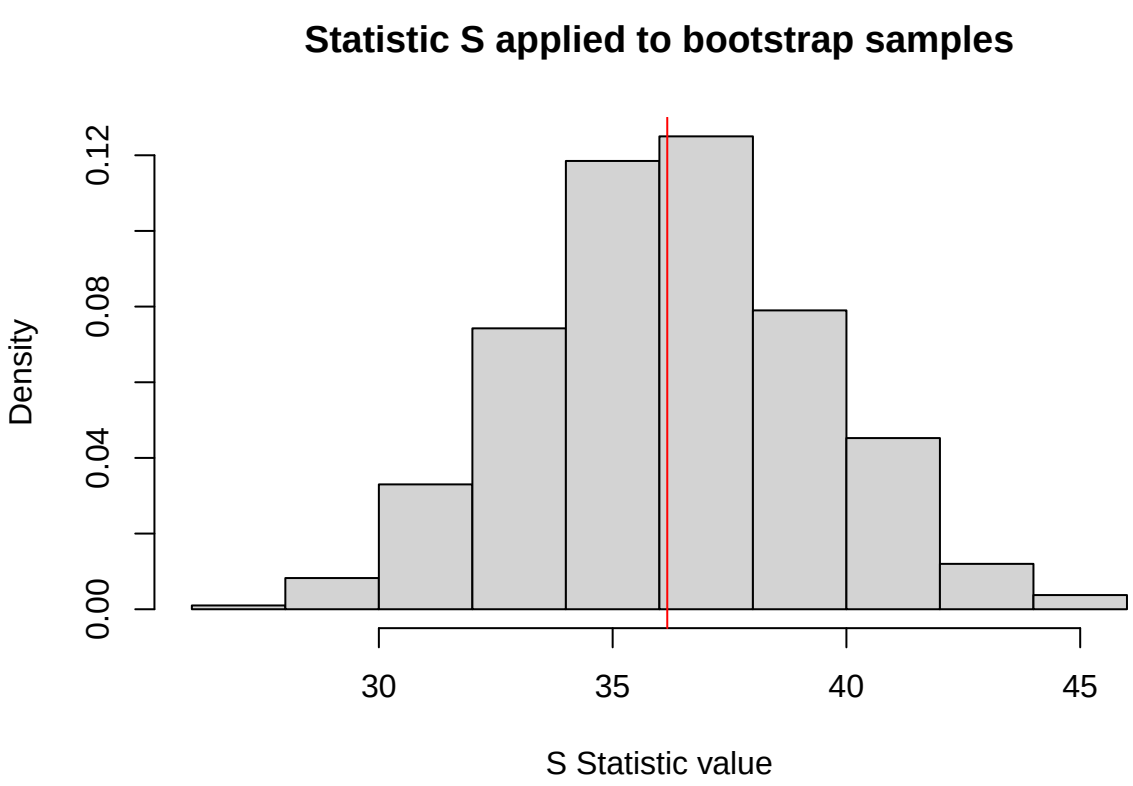
Question 1.4

```
# Question 1.4

# Function to test null hypothesis: that the lottery is
#
# (----> uniformly <----)
#
# random
test_null <- function(data, B){
  # Train a loess model
  loess_hypotest <- loess(data = data,
                          Draft_No ~ Day_of_year)

  # Generate Bootstrap samples
  result <- boot(loess_hypotest$fitted, statistic = statistic_1, R=B)

  # Plot histogram of statistic applied to the Bootstrap samples
  hist(result$t,
       freq = F,
       main = "Statistic S applied to bootstrap samples",
       xlab = "S Statistic value")
  # Add a red line for the observation of S on the original data
  abline(v=result$t0, col="red")

  return(list("Estimate" = result$t0, "p_value" = mean(result$t >= result$t0)))
}
```

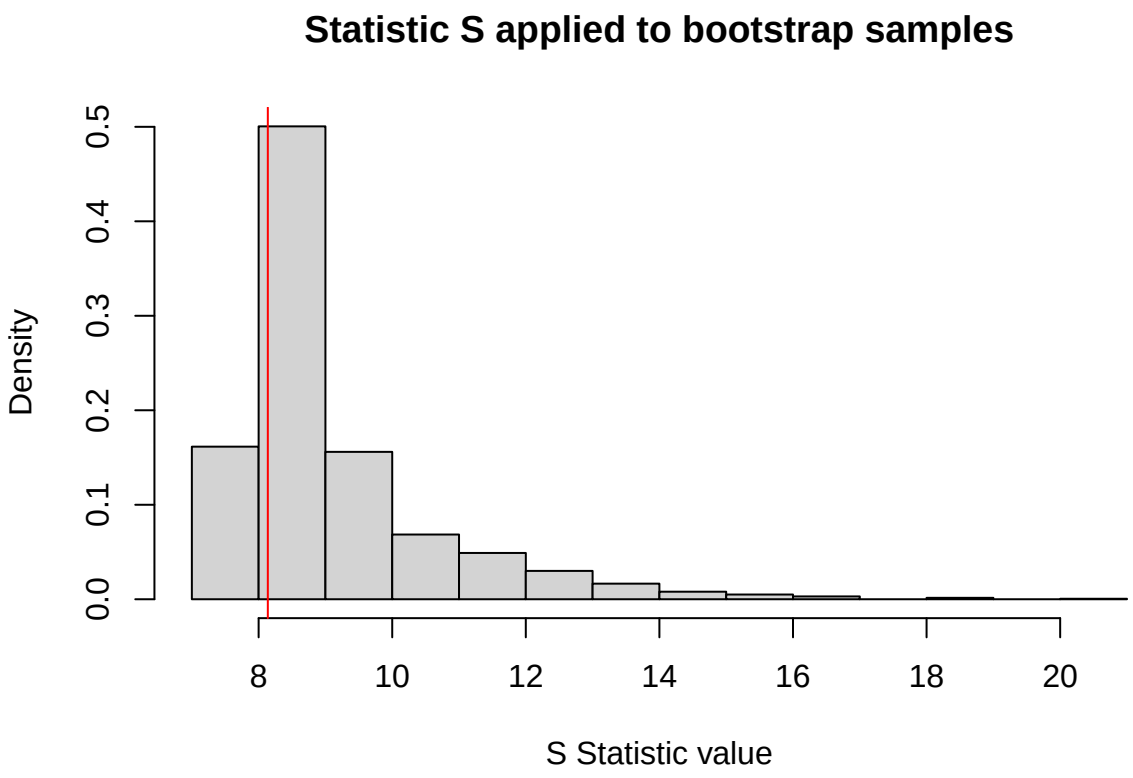
Question 1.5

For some reason our designed test appears to never reject the null hypothesis. The reason for this has likely to do with the statistic function being erroneous in some way. This error might have to do with the indexes that are used to describe the bootstrap samples. It might also have to do with the fact that we chose to use the mean $\bar{Y}$ of each bootstrap sample in the statistic S, rather than the global mean of the observed dates. In this regard the instructions were unclear. In order to further investigate this phenomenon, we have included the creation of histograms of the statistic applied to the Bootstrap samples when the null hypothesis function is run.

```
## Lottery with 40 consecutive dates at the end:
```




```
## $Estimate
## [1] 36.16846
##
## $p_value
## [1] 0.5085

## Lottery with blocks of 13 dates
```

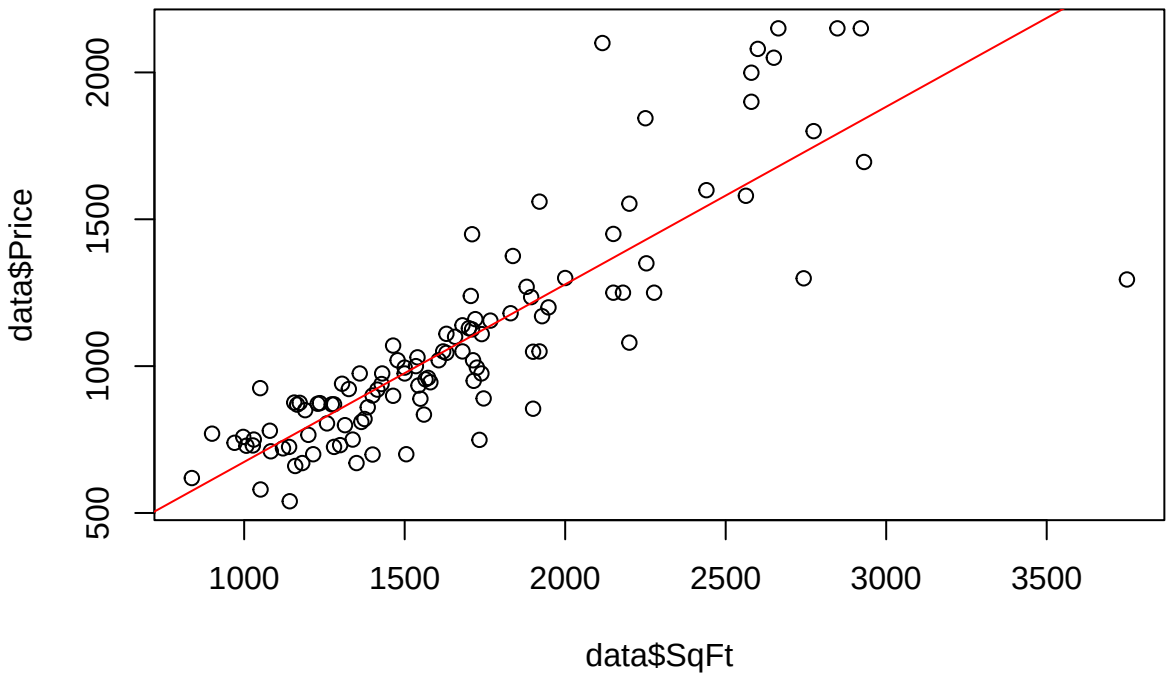



```
## $Estimate
## [1] 8.137427
##
## $p_value
## [1] 0.7475
```

Question 2

1)

```
data<-read.csv2("prices1.csv")
plot(data$SqFt,data$Price)

model<-lm(Price ~ SqFt, data = data)

lines(1:3750,model$coefficients[1]+model$coefficients[2]*(1:3750), col = "red")
```



Yes, a straight line does seem like a good fit for the data.

2)

```
#Price = b + a1*SqFt + a2*(SqFt - c)

sum(model$residuals^2)
```

```
## [1] 4584291
```

```
min<-function(c,data){
  data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
  model2<-lm(Price ~ SqFt + para, data = data)
  sum(model2$residuals^2)
}


opt<-optim(2000,min,data= data,method = "Brent",lower = -500, upper = 4000)       #setting inital value
opt
```

```
## $par
## [1] 3358.36
##
## $value
## [1] 3309413
```

```
##
## $counts
## function gradient
##       NA       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
#Using the optimum value of c found in the model:
data$para<-ifelse(data$SqFt>opt$par,data$SqFt-opt$par,0)
model3<-lm(Price ~ SqFt + para, data = data)
model3
```

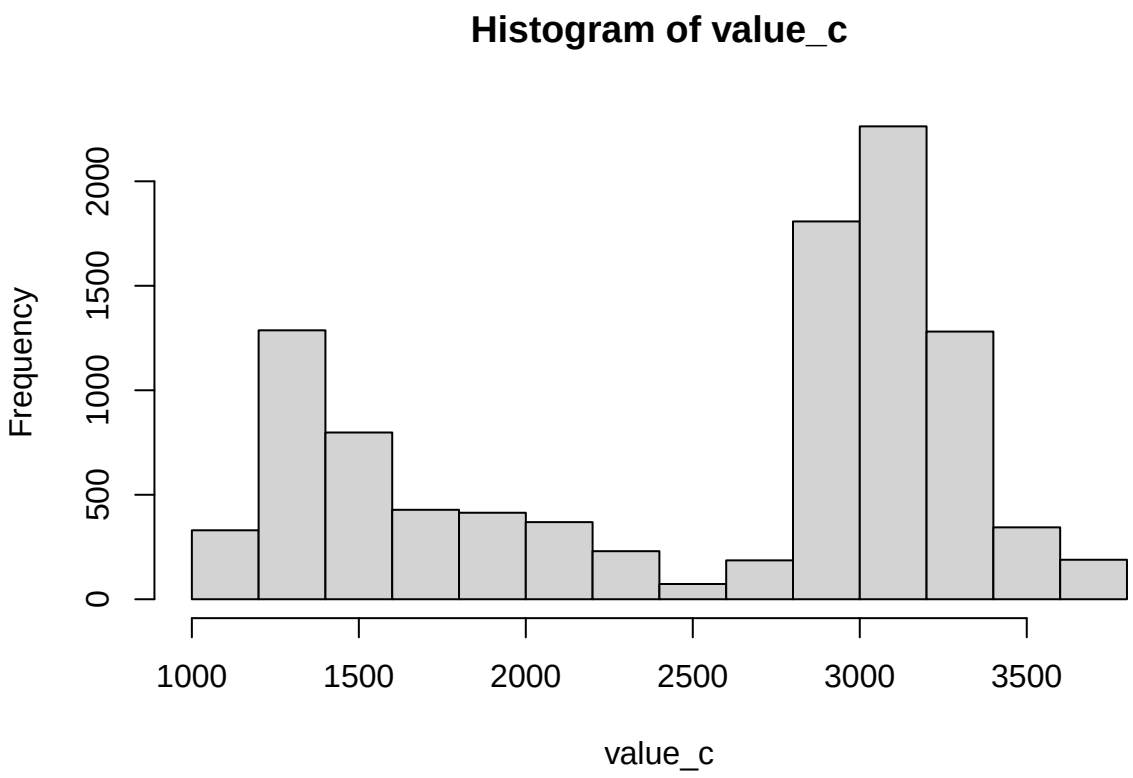
```
##
## Call:
## lm(formula = Price ~ SqFt + para, data = data)
##
## Coefficients:
## (Intercept)         SqFt         para
##    -57.7146       0.6872      -3.1257
```

```
sum(model3$residuals^2) #RSS value
```

```
## [1] 3309413
```

3)

```
#110 rows
data<-read.csv2("prices1.csv")


value_c<-c()
for (i in 1:10000){
  #if (i %% 50 == 0) {print(sprintf("Iteration %d",i))}
  index<-sample(1:110,110,replace = TRUE)
  data2<-data[index,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
  }

  c_opt<-optim(2000,min,data= data2,method = "Brent",lower = -500, upper = 4000)
  value_c<-c(value_c,c_opt$par)
}

#value_c
hist(value_c)
```




```
#based on the plot, we can see 2 peaks, it looks like 2 Gaussian.



#Bootstrap bias correction
2*opt$par-sum(value_c)/10000
```

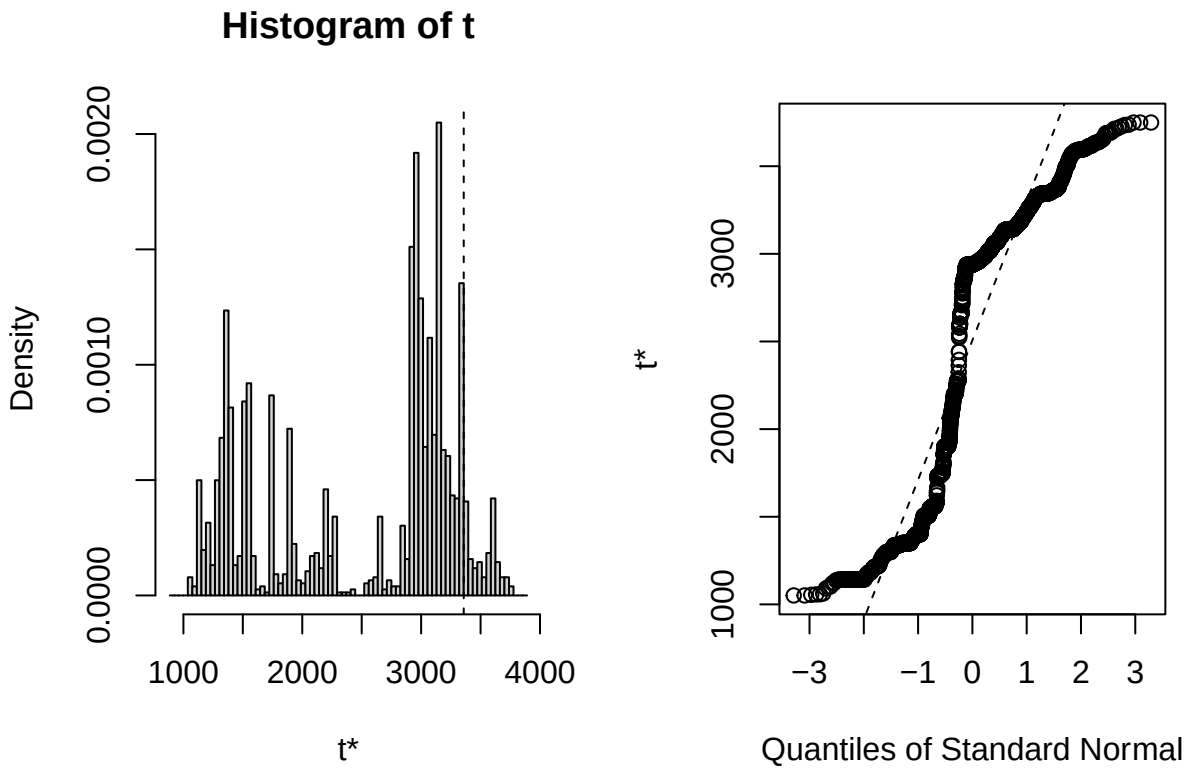
```
## [1] 4186.699
```

```
#Variance
sum((value_c-mean(value_c))^2)/9999
```

```
## [1] 618692.2
```

```
library(boot)


stat<-function(data,index){
  data2<-data[index,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
```

```
  }

  c_opt<-optim(2000,min,data= data2,method = "Brent",lower = -500, upper = 4000)
  c_opt$par
}


var<-boot(data = data,stat , R = 2000)
plot(var)
```




```
#the value of c from the entire data
t0<-var$t0

#the values of c from the bootstrap data
t<-var$t

#Confidence Interval
suppressWarnings(boot.ci(var))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = var)
##
```

```
## Intervals :
## Level      Normal              Basic
## 95%   (2650, 5770 )   (3124, 5558 )
##
## Level     Percentile            BCa
## 95%   (1159, 3593 )   (3243, 3749 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

4)

```
value_c_2<-c()
for (i in 1:110){
  if (i %% 50 == 0) {print(sprintf("Iteration %d",i))}
  data3<-data[-i,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
  }

  c_opt<-optim(2000,min,data= data3,method = "Brent",lower = -500, upper = 4000)
  value_c_2<-c(value_c_2,c_opt$par)
}
```

```
## [1] "Iteration 50"
## [1] "Iteration 100"
```

```
#value_c_2

#variance

term1<-nrow(data)*var$t0 - (nrow(data)-1)*(value_c_2)

term2<-sum(value_c_2)/nrow(data)

sum((term1-term2)^2)/(nrow(data)*(nrow(data)-1))
```

```
## [1] 10155440
```

Upon comparing the variance for jackknife and bootstrap, we can see that jackknife estimate has a relatively higher variance as compared to the bootstrap estimate.

5)

```
boot.ci(var)
```

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

## 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 = var)
##
## Intervals :
## Level      Normal              Basic
## 95%   (2650, 5770 )   (3124, 5558 )
##
## Level     Percentile            BCa
## 95%   (1159, 3593 )   (3243, 3749 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

We see that based on a 95% CI, the BCa gives a tighter CI while the Normal gives the broadest CI. The Basic and Percentile CI, have the same difference value between the max and min range of the CI. While Percentile and BCa gives a range from around 1000s to higher 3000s, Normal and Basic gives a range around 2000s to mid-5000s.

Appendix

```r
# Setup packages and data

library(magrittr)
library(dplyr)
library(boot)

# Read lottery data
df_lottery <- read.table("lottery.txt", col.names = c("Draft_No", "Day_of_year", "Letter_rank"))

# Question 1.1

plot(y=df_lottery$Draft_No,
     x=df_lottery$Day_of_year,
     main = "Fig 1.1: Draft Num. vs Day of year")

# Question 1.2

# Fit a regular linear model
linmod_curve <- lm(data = df_lottery,
                   Draft_No ~ Day_of_year)

# Fit a loess model
loess_curve <- loess(data = df_lottery,
                     Draft_No ~ Day_of_year)

# Add loess fitted values to dataframe
df_lottery <- df_lottery %>% mutate(loess_fitted = loess_curve$fitted)
```

```r
# Plot data along with both fitted curves
p_linmod_curve <- plot(y=df_lottery$Draft_No,
                       x=df_lottery$Day_of_year,
                       main = "Fig 1.2: Draft Num. vs Day of year with")
# Add lm line
abline(b = linmod_curve$coefficients[2], a = linmod_curve$coefficients[1], col="red", lwd=2)

# Add loess curve
loess_data <- df_lottery %>% arrange(Day_of_year)
points(x=loess_data$Day_of_year, loess_data$loess_fitted, type = "l", col="blue", lwd = 2)

# Question 1.3

# Create an S-statistic function
statistic_1 <- function(data, indexes){
  data <- data[indexes]
  S <- (1/length(indexes)) * sum(abs(data-mean(df_lottery$Day_of_year[indexes])))
}

# Perform a test on the fitted data
result <- boot(loess_curve$fitted, statistic = statistic_1, R=2000)

print(boot.ci(result))

cat("\nS estimate:", result$t0)

cat("\nEstimated p-value:", mean(result$t >= result$t0))

# Question 1.4

# Function to test null hypothesis: that the lottery is
#
# (----> uniformly <----)
#
# random
test_null <- function(data, B){
  # Train a loess model
  loess_hypotest <- loess(data = data,
                          Draft_No ~ Day_of_year)

  # Generate Bootstrap samples
  result <- boot(loess_hypotest$fitted, statistic = statistic_1, R=B)

  # Plot histogram of statistic applied to the Bootstrap samples
  hist(result$t,
       freq = F,
       main = "Statistic S applied to bootstrap samples",
       xlab = "S Statistic value")
  # Add a red line for the observation of S on the original data
  abline(v=result$t0, col="red")

  return(list("Estimate" = result$t0, "p_value" = mean(result$t >= result$t0)))
}
```

```r
# Question 1.5

# Specify some k number of dates for special treatment
k <- 40

# Create a new df with k consecutive dates in the end of the year
df_consecutive_k <- data.frame("Draft_No" = c(sample(1:(366-k),size=366-k),
                                              (366-k+1):366),
                               "Day_of_year" = 1:366)


# Create a vector of dates in blocks of size floor(k/3)
dates_in_blocks <- split(1:366, ceiling(1:366/floor(k/3))) %>%
  sample() %>%
  unlist()

# Create a new df with dates in blocks of floor(k/3) dates
df_blocks_k <- data.frame("Draft_No" = dates_in_blocks,
                          "Day_of_year" = 1:366)


# Run Tests
cat("Lottery with", k, "consecutive dates at the end:\n")
test_null(df_consecutive_k, 2000)
cat("Lottery with blocks of", floor(k/3), "dates\n")
test_null(df_blocks_k, 2000)

data<-read.csv2("prices1.csv")
plot(data$SqFt,data$Price)

model<-lm(Price ~ SqFt, data = data)

lines(1:3750,model$coefficients[1]+model$coefficients[2]*(1:3750), col = "red")
#Price = b + a1*SqFt + a2*(SqFt - c)

sum(model$residuals^2)

min<-function(c,data){
  data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
  model2<-lm(Price ~ SqFt + para, data = data)
  sum(model2$residuals^2)
}


opt<-optim(2000,min,data= data,method = "Brent",lower = -500, upper = 4000)       #setting inital value
opt



#Using the optimum value of c found in the model:
data$para<-ifelse(data$SqFt>opt$par,data$SqFt-opt$par,0)
model3<-lm(Price ~ SqFt + para, data = data)
model3
```

```r
sum(model3$residuals^2) #RSS value

#110 rows
data<-read.csv2("prices1.csv")


value_c<-c()
for (i in 1:10000){
  #if (i %% 50 == 0) {print(sprintf("Iteration %d",i))}
  index<-sample(1:110,110,replace = TRUE)
  data2<-data[index,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
  }

  c_opt<-optim(2000,min,data= data2,method = "Brent",lower = -500, upper = 4000)
  value_c<-c(value_c,c_opt$par)
}

#value_c
hist(value_c)
#based on the plot, we can see 2 peaks, it looks like 2 Gaussian.



#Bootstrap bias correction
2*opt$par-sum(value_c)/10000


#Variance
sum((value_c-mean(value_c))^2)/9999

library(boot)


stat<-function(data,index){
  data2<-data[index,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
  }

  c_opt<-optim(2000,min,data= data2,method = "Brent",lower = -500, upper = 4000)
  c_opt$par
}


var<-boot(data = data,stat , R = 2000)
```

```r
plot(var)

#the value of c from the entire data
t0<-var$t0

#the values of c from the bootstrap data
t<-var$t

#Confidence Interval
suppressWarnings(boot.ci(var))
value_c_2<-c()
for (i in 1:110){
  if (i %% 50 == 0) {print(sprintf("Iteration %d",i))}
  data3<-data[-i,]

  min<-function(c,data){
    data$para<-ifelse(data$SqFt>c,data$SqFt-c,0)
    model2<-lm(Price ~ SqFt + para, data = data)
    sum(model2$residuals^2)
  }

  c_opt<-optim(2000,min,data= data3,method = "Brent",lower = -500, upper = 4000)
  value_c_2<-c(value_c_2,c_opt$par)
}

#value_c_2

#variance

term1<-nrow(data)*var$t0 - (nrow(data)-1)*(value_c_2)

term2<-sum(value_c_2)/nrow(data)

sum((term1-term2)^2)/(nrow(data)*(nrow(data)-1))

boot.ci(var)
```