# Computational Statistics Computer Lab 5 (Group 7)

Qinyuan Qi(qinqi464)     Satya Sai Naga Jaya Koushik Pilla (satpi345)

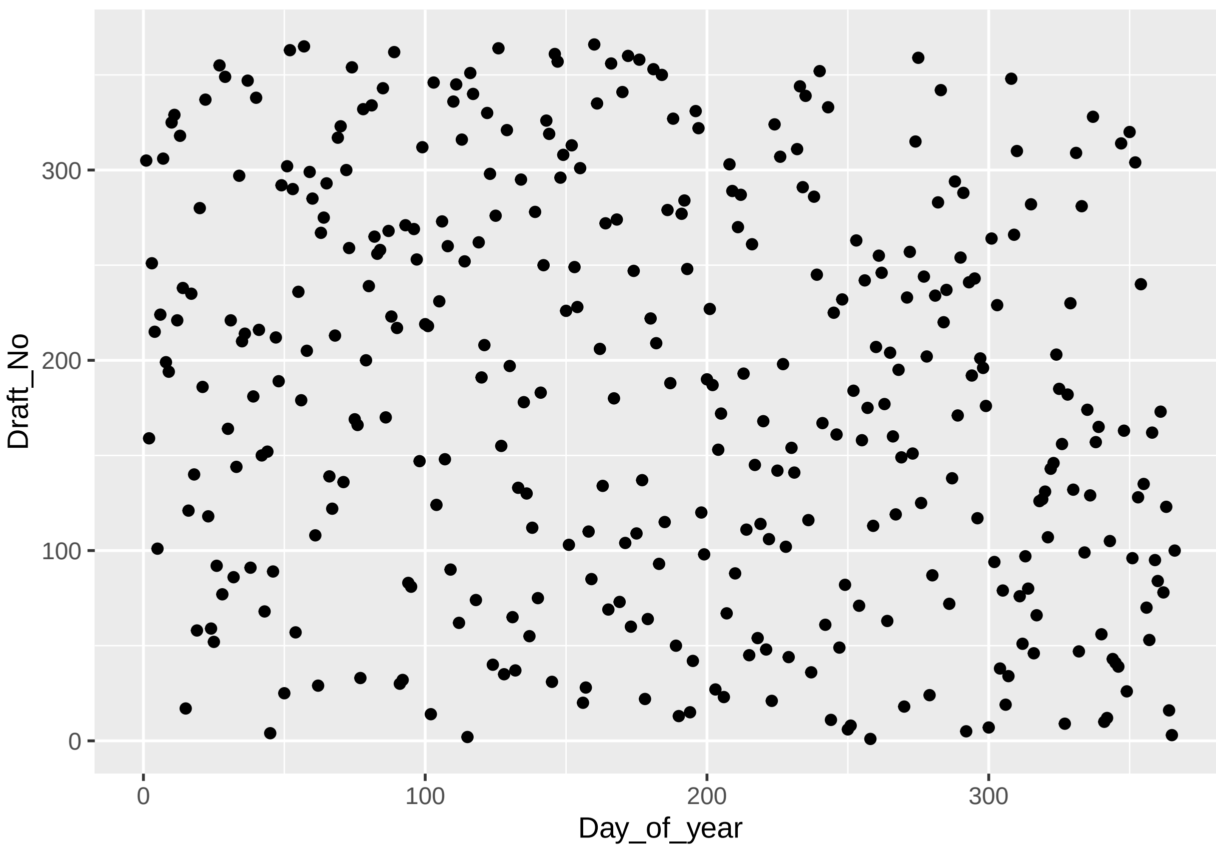
2024-01-27

## Question 1: Hypothesis testing (Solved by Qinyuan Qi)

**Answer:**

**(1.1) Make a scatter plot of Y versus X and conclude**

The plot is as follows, and according to the plot, the pattern is not very clear. So it seems to be random distributed.

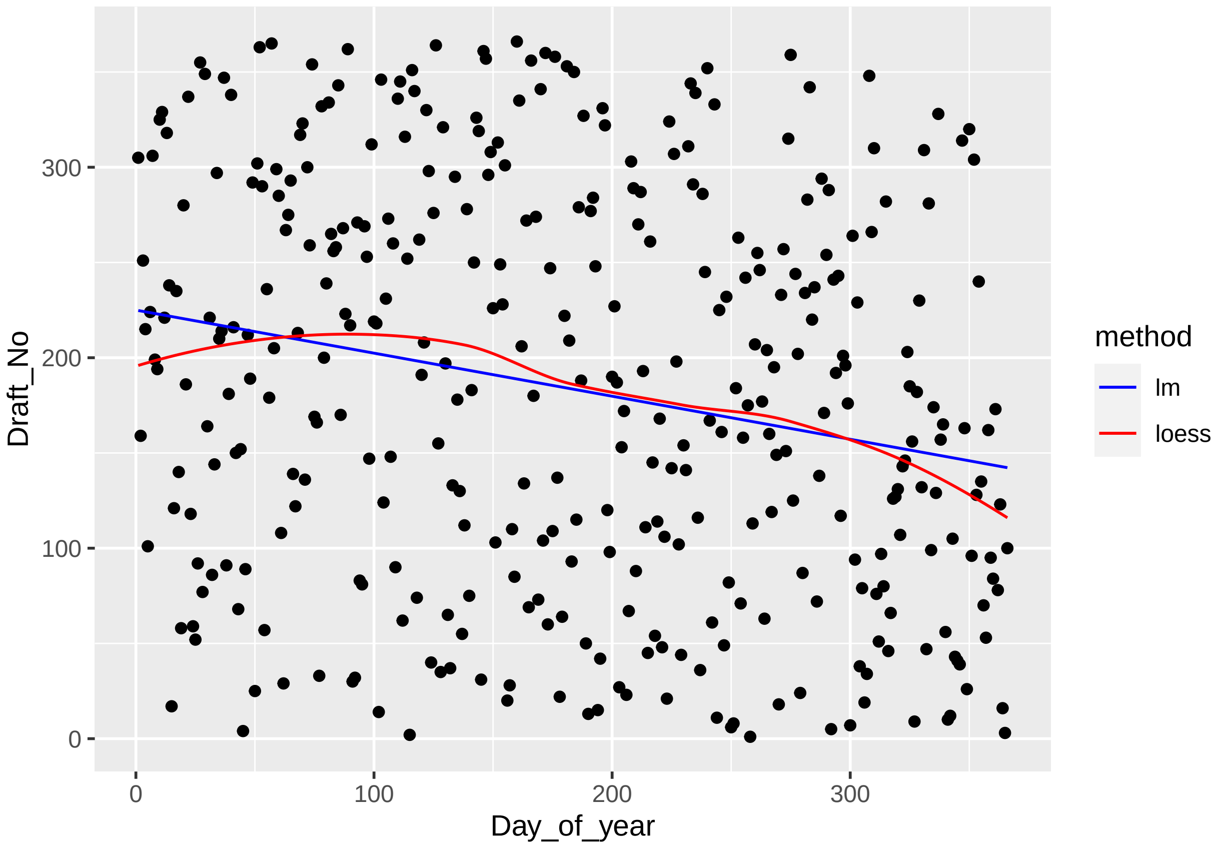



**(1.2) Fit a curve to the data and conclude**

Fitting the points using lm() and loess(), the plot is as below.

According to the plot, we can find that the lm fit line and loess fit line is not flat, meanwhile, when Day_of_Year value is greater than 100, the smoothness of loess line has a trend to increase.

So we can guess that the distribution of data may not be random.




**(1.3) Estimate S's distribution through a non–parametric bootstrap**

The bootstrap value is generated using the boot function.

We use the following formula to calc the p-value:

$$\hat{p} = \frac{\sum S \geqslant S0}{N}$$

The S and p-value for the loess regression are listed as follows.

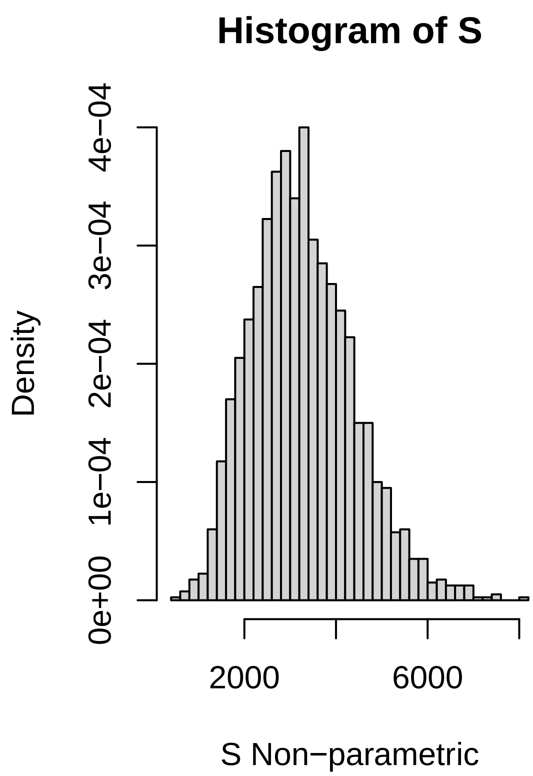
S value is very big compare to zero, and the p-value is 0 and 5e-04, which is less than 0.05, so we know that the data is not random.

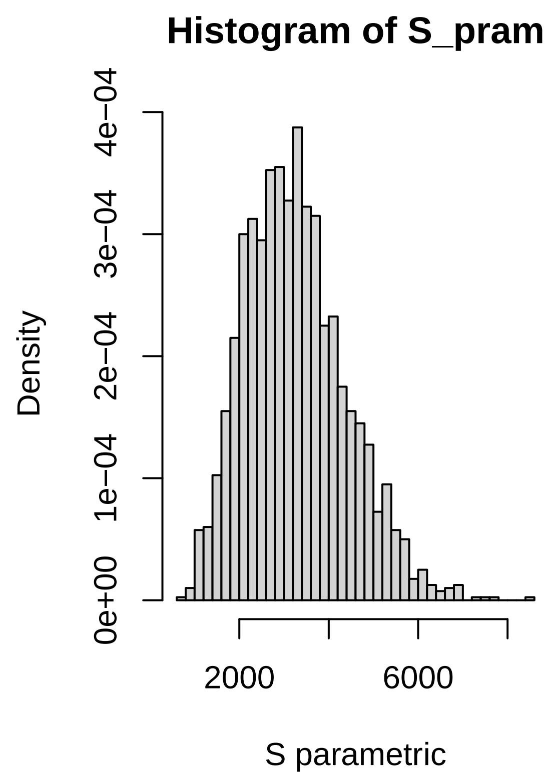
```
## S value for loess regression(non-parametric) =  3285.997
```

```
## p-value for loess regression(non-parametric) =  0
```

```
## S value for loess regression(parametric) =  3261.089
```

```
## p-value for loess regression(parametric) =  5e-04
```




**(1.4)**

The code will generate the value of S and its p–value, based on 2000 bootstrap samples. Accoring to the result, we reject null Hypothesis, which means dataset is not random.

```
## Reject null Hypothesis,dataset is not random
```

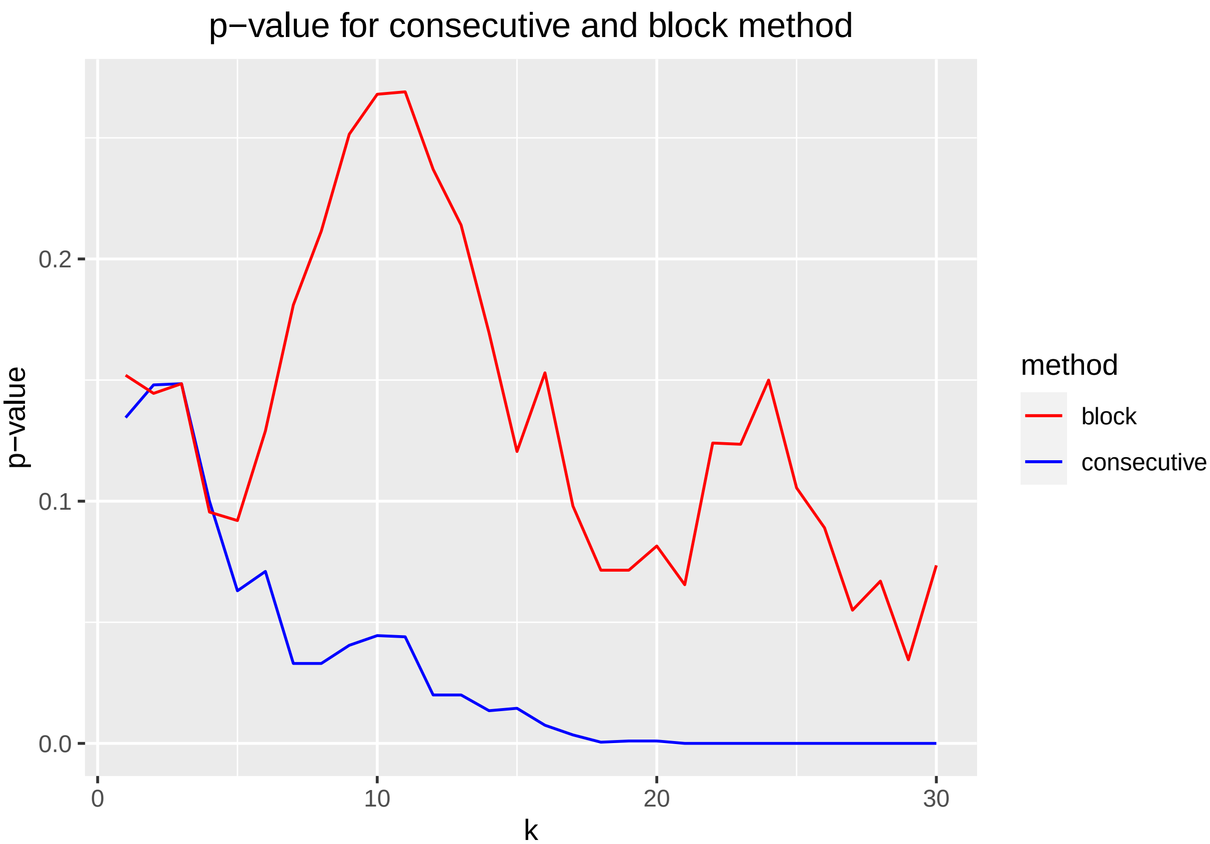
**(1.5)**

We will generate dataset of the same dimensions as the original data. please check the appendix for reference. we set k from 1 to 30, and test the hypothesis. Accoring to the output, we know that when k=7 we begin to reject null Hypothesis using consecutive method. and when k = 29 we begin to reject null Hypothesis using block method.

According to the plot, we find block method is much slower begin to reject null Hypothesis than consecutive method.

```
## Reject null Hypothesis using consecutive method when k= 7
## Reject null Hypothesis using consecutive method when k= 8
## Reject null Hypothesis using consecutive method when k= 9
## Reject null Hypothesis using consecutive method when k= 10
## Reject null Hypothesis using consecutive method when k= 11
## Reject null Hypothesis using consecutive method when k= 12
## Reject null Hypothesis using consecutive method when k= 13
## Reject null Hypothesis using consecutive method when k= 14
## Reject null Hypothesis using consecutive method when k= 15
## Reject null Hypothesis using consecutive method when k= 16
## Reject null Hypothesis using consecutive method when k= 17
## Reject null Hypothesis using consecutive method when k= 18
## Reject null Hypothesis using consecutive method when k= 19
## Reject null Hypothesis using consecutive method when k= 20
```

```
## Reject null Hypothesis using consecutive method when k= 21
## Reject null Hypothesis using consecutive method when k= 22
## Reject null Hypothesis using consecutive method when k= 23
## Reject null Hypothesis using consecutive method when k= 24
## Reject null Hypothesis using consecutive method when k= 25
## Reject null Hypothesis using consecutive method when k= 26
## Reject null Hypothesis using consecutive method when k= 27
## Reject null Hypothesis using consecutive method when k= 28
## Reject null Hypothesis using consecutive method when k= 29
## Reject null Hypothesis using block method when k= 29
## Reject null Hypothesis using consecutive method when k= 30
```




## Question 2: Bootstrap, jackknife and confidence intervals (Solved by Satya Sai Naga Jaya Koushik Pilla)

**Answer:**

**(2.1)**

we create a scatter plot of SqFt versus Price, and fit a line.Price = 69.30663 + 0.60457 * SqFt

```
################################### (2.1) ########################################
data <- read.csv("prices1.csv", sep = ";")

g_2_1 <- ggplot(data, aes(x = SqFt, y = Price)) + geom_point()

# fit a linear regression model
fit2_1_lm <- lm(Price ~ SqFt, data = data)
```

```r
summary(fit2_1_lm)
```

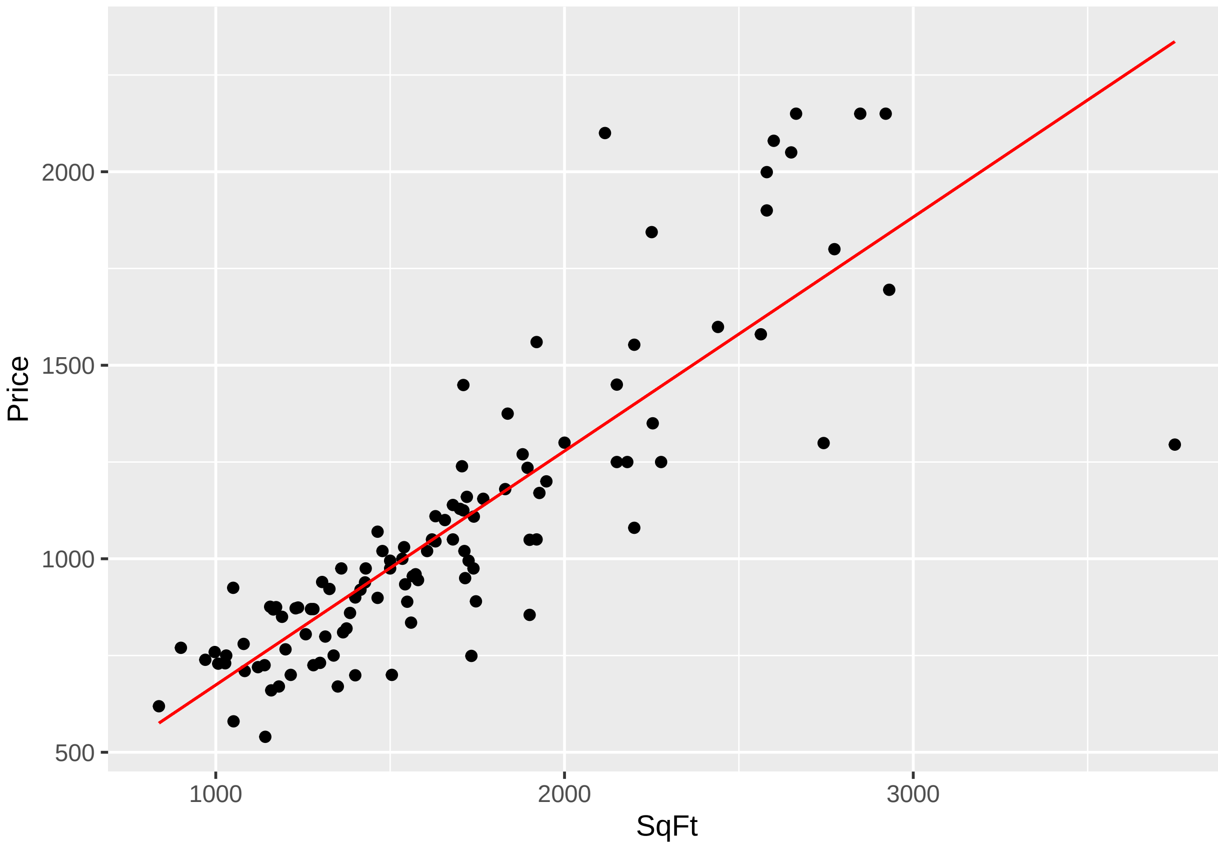
```
##
## Call:
## lm(formula = Price ~ SqFt, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1041.43   -99.12     1.99    59.26   751.43
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.30663   65.13461   1.064     0.29
## SqFt         0.60457    0.03713  16.282   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 206 on 108 degrees of freedom
## Multiple R-squared:  0.7105, Adjusted R-squared:  0.7079
## F-statistic: 265.1 on 1 and 108 DF,  p-value: < 2.2e-16
```

```r
minimal_x <- min(data$SqFt)
maximum_x <- max(data$SqFt)

x_2_1_lm <- seq(minimal_x, maximum_x, 0.1)
y_2_1_lm <- predict(fit2_1_lm, newdata = data.frame(SqFt = x_2_1_lm))

data_2_1 <- data.frame(SqFt = x_2_1_lm, Price = y_2_1_lm)

g_2_1 + geom_line(data = data_2_1, aes(x = x_2_1_lm, y = y_2_1_lm), color = "red")
```




According to the plotted line, it does not seem like a good fit. since the left bottom of the plot is very dense, but top right corner is very sparse.

**(2.2)**

According to the question, we need to min RSS as follows

$$RSS = \sum (y_i - f(f(x_i)))^2$$

And the code implementation as follows.

```
rss_fun <- function(par,SqFt,Price) {

  a1 <- par[1]
  a2 <- par[2]
  b <- par[3]
  c <- par[4]

  predicted <- ifelse(SqFt > c, b + a1 * SqFt + a2 * (SqFt - c), b + a1 * SqFt)

  rss <- sum((Price - predicted)^2)
  return(rss)
}

c_func <- function(c, SqFt, Price) {
  # Init
  init_pars <- c(2, 0.1, 50,c)
```

```r
  # Optimize using optim()
  result <- optim(par = init_pars, rss_fun, SqFt = SqFt, Price = Price)

  opt_pars <- result$par
  rss <- result$value

  return(list(c = opt_pars, params = opt_pars, rss = rss))
}


c <- 150
results <- c_func(c, SqFt = data$SqFt, Price = data$Price)
results
```

```
## $c
## [1]   0.68946673  -0.08460581  55.69699121 156.71143820
##
## $params
## [1]   0.68946673  -0.08460581  55.69699121 156.71143820
##
## $rss
## [1] 4584296
```

**(2.3)**

We estimate the distribution of c using bootstrap

```r
stat1 <- function(data,vn){
    data = as.data.frame(data[vn,])
    res <- c_func(c, SqFt = data$SqFt, Price = data$Price)
    return(res$param[4])
}

set.seed(12345)
res1 = boot(data, stat1, R=1000)
res1
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data, statistic = stat1, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 156.7114 2.875368    3.410341
```

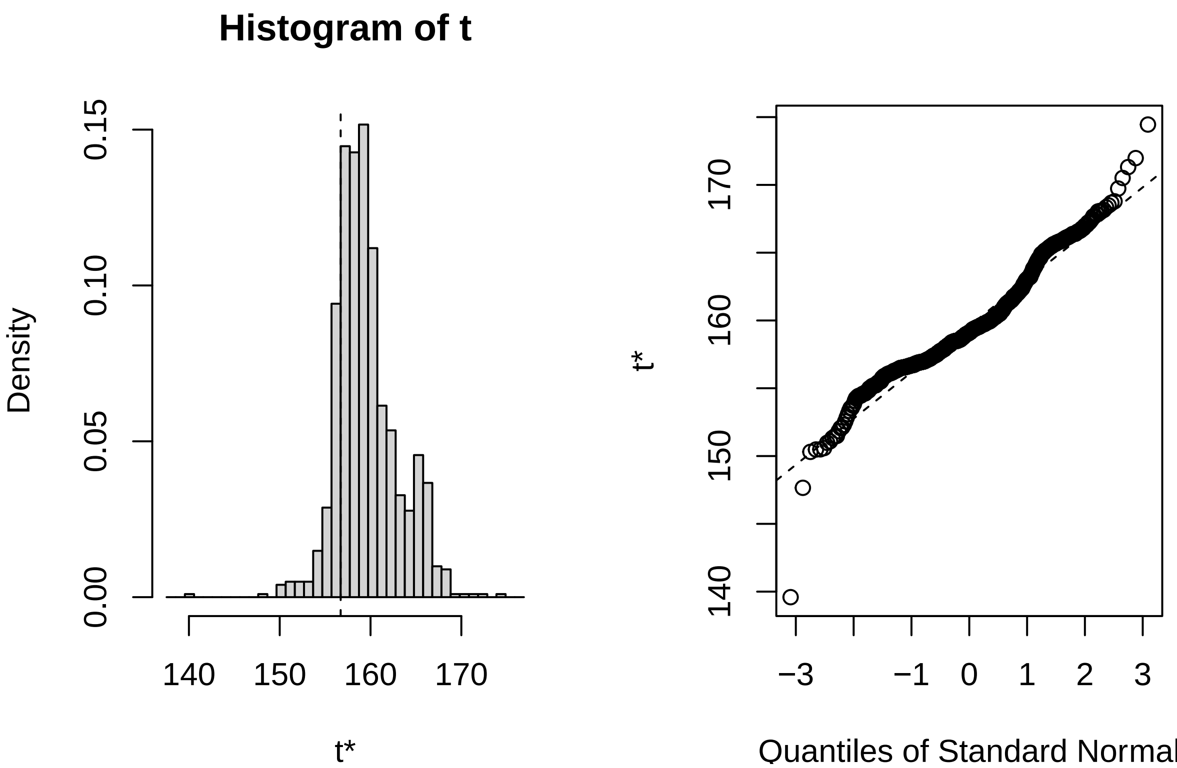
We calc the 95% confidence interval

```r
print(boot.ci(res1))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = res1)
##
## Intervals :
## Level      Normal              Basic
## 95%   (147.2, 160.5 )   (146.7, 159.2 )
##
## Level     Percentile            BCa
## 95%   (154.3, 166.8 )   (139.6, 159.0 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

```
# plot
plot(res1)
```




**(2.4)**

Estimate the variance of c using the jackknife:

```
# Jackknife Function
jackknife <- function(data, fun) {
  n <- length(data)
  indices <- 1:n
  estimates <- numeric(n)

  for (i in 1:n) {
    jackknife_sample <- data[-i]
    estimates[i] <- fun(jackknife_sample)
  }
```

```r
  bias_correction <- (n - 1) * mean(estimates) - fun(data)

  jackknife_var <- ((n - 1) / n) * sum((estimates - mean(estimates) - bias_correction)^2)

  return(list(estimates = estimates, jackknife_var = jackknife_var))
}

# not working here,c_values

jackknife_results <- jackknife(c_values, function(c_val) {
  c_func(c_val, SqFt, Price)$params[1]
})

# Print Jackknife Results
cat("Jackknife Estimates of c:", jackknife_results$estimates, "\n")
cat("Jackknife Variance of c:", jackknife_results$jackknife_var, "\n")
```

**(2.5)**

N/A

## Appendix: Code for this report

```
########################## Init code for question 1 ############################
rm(list = ls())
library(ggplot2)
library(boot)
set.seed(12345)
################################### (1.1) #######################################
data <- read.csv("lottery.csv", sep = ";")

# Remove Month column since there is a Mo.Number column which is same as Month
data <- data[,4:5]

g_1_1 <- ggplot(data, aes(x = Day_of_year, y =Draft_No )) + geom_point()
g_1_1
################################### (1.2) #######################################

minimal_x <- min(data$Day_of_year)
maximum_x <- max(data$Day_of_year)

x_1_2 <- seq(minimal_x, maximum_x, 0.1)

# fit the data using linear regression
fit1_2_lm <- lm(Draft_No ~ Day_of_year, data = data)
#summary(fit1_2_lm)
y_1_2_lm <- predict(fit1_2_lm, newdata = data.frame(Day_of_year = x_1_2))

# fit using loess function
fit1_2_loess <- loess( Draft_No ~ Day_of_year, data = data)
#summary(fit1_2_loess)
y_1_2_loess <- predict(fit1_2_loess, newdata = data.frame(Day_of_year = x_1_2))


# plot 2 predicted lines
data_1_2 <- data.frame(Day_of_year = x_1_2, y_lm = y_1_2_lm, y_loess = y_1_2_loess)

g_1_1 +
geom_line(data = data_1_2, aes(x = Day_of_year, y = y_lm,colour="lm")) +
geom_line(data = data_1_2, aes(x = Day_of_year, y = y_loess, colour = "loess")) +
scale_color_manual(name = "method", values = c("lm" = "blue", "loess" = "red"))
################################### (1.3) #######################################
# estimate S' dist using non-parametric bootstrap

bootstrap_sample_number <- 2000
S <- numeric(bootstrap_sample_number)
S_pram <- numeric(bootstrap_sample_number)
loess_mod <- loess(Draft_No ~ Day_of_year, data = data)
mean_draft_no <- mean(data$Draft_No)

observed_s <- sum(abs(predict(loess_mod) - mean_draft_no))

set.seed(12345)

for( b in 1:bootstrap_sample_number){
```

```r
  # generate bootstrap sample
  draft_no_sample <- sample(data$Draft_No, replace = TRUE)
  df_sample <- data.frame(Day_of_year = data$Day_of_year, Draft_No = draft_no_sample)
  # fit loess model
  loess_sample_mod <- loess(Draft_No ~ Day_of_year, data = df_sample)
  # calc S
  S[b] <- sum(abs(predict(loess_sample_mod) - mean_draft_no))
}

# print S value
cat("S value for loess regression(non-parametric) = ", mean(S), "\n")
# calc p-value
p_value <- mean(S >= observed_s)
cat("p-value for loess regression(non-parametric) = ", p_value, "\n")

# estimate S' dist using parametric bootstrap

set.seed(12345)
for( b in 1:bootstrap_sample_number){
  # generate bootstrap sample
  draft_no_sample <- sample(data$Draft_No, replace = TRUE)
  df_param_sample <- data.frame(Day_of_year = data$Day_of_year, Draft_No = draft_no_sample)
  # fit model
  loess_param_sample_mod <- loess(Draft_No ~ Day_of_year, data = df_param_sample)
  # get residuals
  redisuals <- rnorm(nrow(df_param_sample), mean = 0, sd = sd(resid(loess_param_sample_mod)))
  df_param_sample$Draft_No <- predict(loess_param_sample_mod) + redisuals
  # get mean
  mean_parm_draft_no <- mean(df_param_sample$Draft_No)

  # calc S_pram
  S_pram[b] <- sum(abs(predict(loess_param_sample_mod) - mean_parm_draft_no))
}

# print S value
cat("S value for loess regression(parametric) = ", mean(S_pram), "\n")
# calc p-value
p_value_param <- mean(S_pram >= observed_s)
cat("p-value for loess regression(parametric) = ", p_value_param, "\n")

par(mfrow = c(1,2))
hist(S, breaks = 50, freq = F, xlab = "S Non-parametric")
hist(S_pram, breaks = 50, freq = F, xlab = "S parametric")

test_hypothesis <- function(data, B = 2000) {
  set.seed(12345)
  s_value_loess <- numeric(B)
  loess_mod <- loess(Draft_No ~ Day_of_year, data = data)
  observed_s <- sum(abs(predict(loess_mod) - mean(data$Draft_No)))

  for(b in 1:B){
    # generate bootstrap sample
    draft_no_sample <- sample(data$Draft_No, replace = TRUE)
```

```r
    df_sample <- data.frame(Day_of_year = data$Day_of_year, Draft_No = draft_no_sample)
    # fit loess model
    loess_sample_mod <- loess(Draft_No ~ Day_of_year, data = df_sample)
    # calc S
    s_value_loess[b] <- sum(abs(predict(loess_sample_mod) - mean(df_sample$Draft_No)))
  }

  p_value_loess <- mean(s_value_loess >= observed_s)

  result <- list(
    s_loess_value = s_value_loess,
    p_loess_value = p_value_loess,
    observed_s = observed_s
  )
  return(result)
}

test_result <- test_hypothesis(data = data, B = 2000)
if (test_result$p_loess_value <= 0.05) {
  cat("Reject null Hypothesis,dataset is not random","\n")
} else {
  cat("Fail to Reject null Hypothesis,dataset is random","\n")
}
################################## (1.5.a) #######################################
# function yo generate biased dataset with same dimension as original dataset
gen_bias_data <- function(k, method = "consecutive"){

  bias_data <- data.frame(Day_of_year = data$Day_of_year, Draft_No = NA)

  if (method == "consecutive"){
    start_dates <- sample(1:(366 - k + 1),1)
    bias_date_indexs <- start_dates:(start_dates + k - 1)

    # assign them the end number of the lotty dataset
    bias_data$Draft_No[bias_date_indexs] <- (366 - k + 1):366

    # get other dates index
    other_dates <- setdiff(1:366, bias_date_indexs)

    # assign other dates with random number
    bias_data$Draft_No[other_dates] <- sample(1:(366 - k),
                                              size = length(other_dates),
                                              replace = TRUE)
  }else if (method == "blocks"){
    block_size <- floor(k / 3)
    remaining <- k - block_size * 3

    bias_date_indexs <- c()

    for(i in 1:block_size){
      start_dates <- sample(setdiff(1:366,bias_date_indexs),1)
      bias_date_indexs <- c(bias_date_indexs, start_dates:(start_dates + 2))
    }
```

```r
    if (remaining > 0){
      start_dates <- sample(setdiff(1:366,bias_date_indexs),1)
      bias_date_indexs <- c(bias_date_indexs, start_dates:(start_dates + remaining - 1))
    }

    # assign them the end number of the lotty dataset
    bias_data$Draft_No[bias_date_indexs] <- (366 - k + 1):366

    # get other dates index
    other_dates <- setdiff(1:366, bias_date_indexs)

    # assign other dates with random number
    bias_data$Draft_No[other_dates] <- sample(1:(366 - k),
                                              size = length(other_dates),
                                              replace = TRUE)
  }
  return(bias_data)
}
################################### (1.5.b) #######################################
# perform test on biased dataset
perform_bootstrap_test <- function(k, method){
  # generate biased dataset
  bias_data <- gen_bias_data(k, method)

  # test hypothesis
  test_result <- test_hypothesis(data = bias_data, B = 2000)

  return(test_result)
}
################################### (1.5.c) #######################################
consecutive_p_values <- numeric(30)
block_p_values <- numeric(30)

for(k in 1:30){
  consecutive_p_values[k] <- perform_bootstrap_test(k, "consecutive")$p_loess_value
  block_p_values[k] <- perform_bootstrap_test(k, "blocks")$p_loess_value
  if (consecutive_p_values[k] <= 0.05) {
    cat("Reject null Hypothesis using consecutive method when k=",k,"\n")
  }
  if (block_p_values[k] <= 0.05) {
    cat("Reject null Hypothesis using block method when k=",k,"\n")
  }

}
################################### (1.5.c) #######################################
p_value_df <- data.frame(k = 1:30, consecutive_p_values = consecutive_p_values,
                         block_p_values = block_p_values)

ggplot(data = p_value_df,) +
geom_line(aes(x = k, y = consecutive_p_values, colour = "consecutive")) +
geom_line(aes(x = k, y = block_p_values, colour = "block")) +
scale_color_manual(name = "method", values = c("consecutive" = "blue", "block" = "red")) +
xlab("k") + ylab("p-value") + ggtitle("p-value for consecutive and block method") +
```

```
theme(plot.title = element_text(hjust = 0.5))

########################## Init code for question 2 ############################
### Init Code
rm(list = ls())
library(ggplot2)
library(boot)
################################### (2.1) ########################################
data <- read.csv("prices1.csv", sep = ";")

g_2_1 <- ggplot(data, aes(x = SqFt, y = Price)) + geom_point()

# fit a linear regression model
fit2_1_lm <- lm(Price ~ SqFt, data = data)
summary(fit2_1_lm)

minimal_x <- min(data$SqFt)
maximum_x <- max(data$SqFt)

x_2_1_lm <- seq(minimal_x, maximum_x, 0.1)
y_2_1_lm <- predict(fit2_1_lm, newdata = data.frame(SqFt = x_2_1_lm))

data_2_1 <- data.frame(SqFt = x_2_1_lm, Price = y_2_1_lm)

g_2_1 + geom_line(data = data_2_1, aes(x = x_2_1_lm, y = y_2_1_lm), color = "red")

rss_fun <- function(par,SqFt,Price) {

  a1 <- par[1]
  a2 <- par[2]
  b <- par[3]
  c <- par[4]

  predicted <- ifelse(SqFt > c, b + a1 * SqFt + a2 * (SqFt - c), b + a1 * SqFt)

  rss <- sum((Price - predicted)^2)
  return(rss)
}

c_func <- function(c, SqFt, Price) {
  # Init
  init_pars <- c(2, 0.1, 50,c)

  # Optimize using optim()
  result <- optim(par = init_pars, rss_fun, SqFt = SqFt, Price = Price)

  opt_pars <- result$par
  rss <- result$value

  return(list(c = opt_pars, params = opt_pars, rss = rss))
}
```

```r
c <- 150
results <- c_func(c, SqFt = data$SqFt, Price = data$Price)
results
stat1 <- function(data,vn){
    data = as.data.frame(data[vn,])
    res <- c_func(c, SqFt = data$SqFt, Price = data$Price)
    return(res$param[4])
}

set.seed(12345)
res1 = boot(data, stat1, R=1000)
res1

print(boot.ci(res1))

# plot
plot(res1)
# Jackknife Function
jackknife <- function(data, fun) {
  n <- length(data)
  indices <- 1:n
  estimates <- numeric(n)

  for (i in 1:n) {
    jackknife_sample <- data[-i]
    estimates[i] <- fun(jackknife_sample)
  }

  bias_correction <- (n - 1) * mean(estimates) - fun(data)

  jackknife_var <- ((n - 1) / n) * sum((estimates - mean(estimates) - bias_correction)^2)

  return(list(estimates = estimates, jackknife_var = jackknife_var))
}

# not working here,c_values

jackknife_results <- jackknife(c_values, function(c_val) {
  c_func(c_val, SqFt, Price)$params[1]
})

# Print Jackknife Results
cat("Jackknife Estimates of c:", jackknife_results$estimates, "\n")
cat("Jackknife Variance of c:", jackknife_results$jackknife_var, "\n")
```