

Computational Statistics Computer Lab 5 (Group 7)

Qinyuan Qi(qinqi464)

Satya Sai Naga Jaya Koushik Pilla (satpi345)

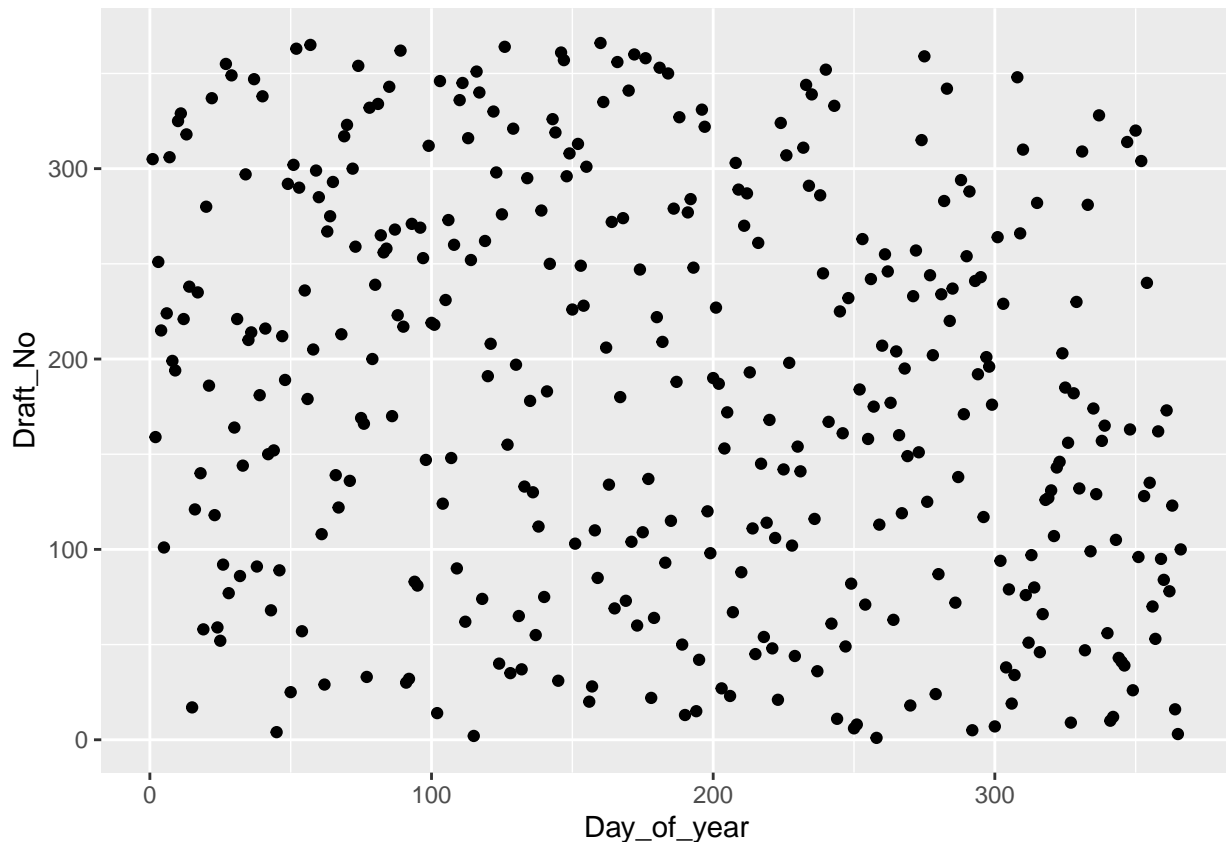
2024-01-28

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.

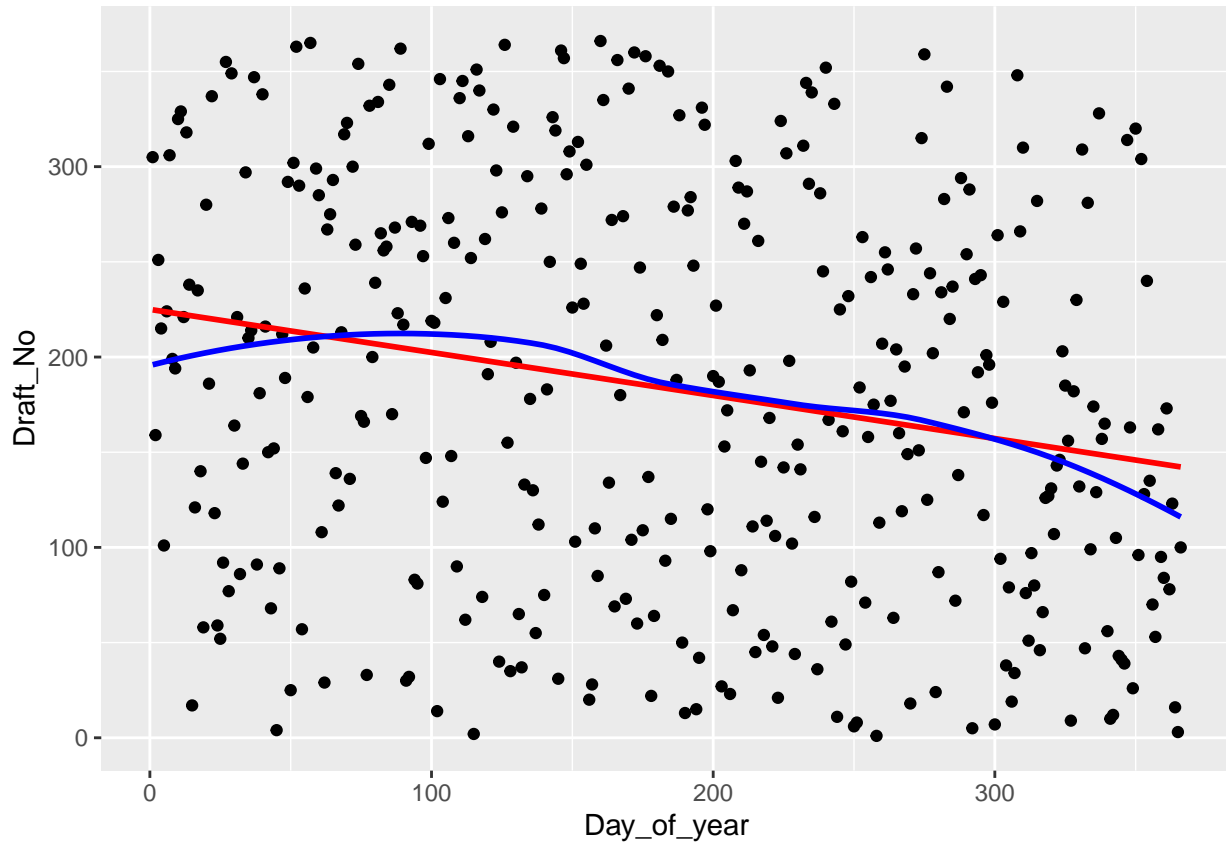


(1.2) Fit a curve to the data and conclude

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

According to the plot, we can find that the `lm` fit line(red) and `loess` fit line(blue) 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.

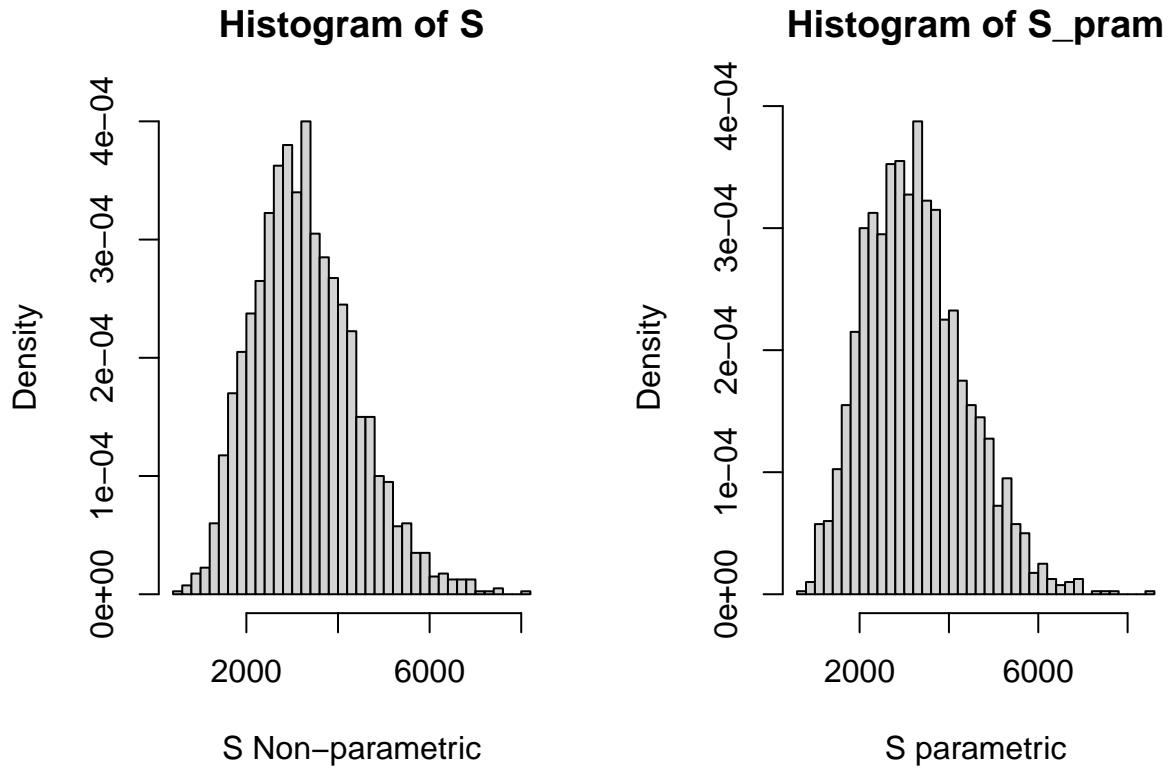
We use the following formula to calc the p-value:

$$\hat{p} = \frac{\sum S \geq S_0}{N}$$

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. According 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. According 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.

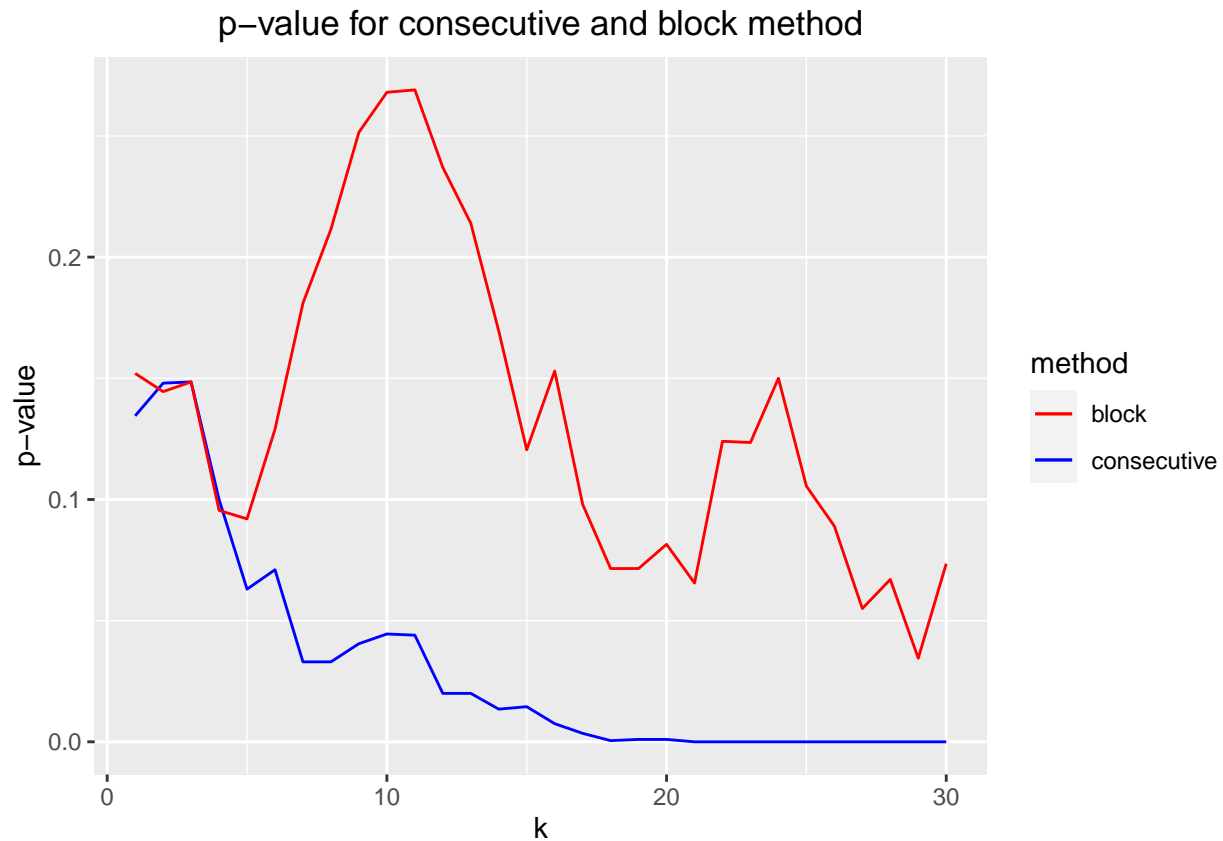
According to the plot, we find block method is much slower begin to reject null Hypothesis than consecutive method.

```
## 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

```

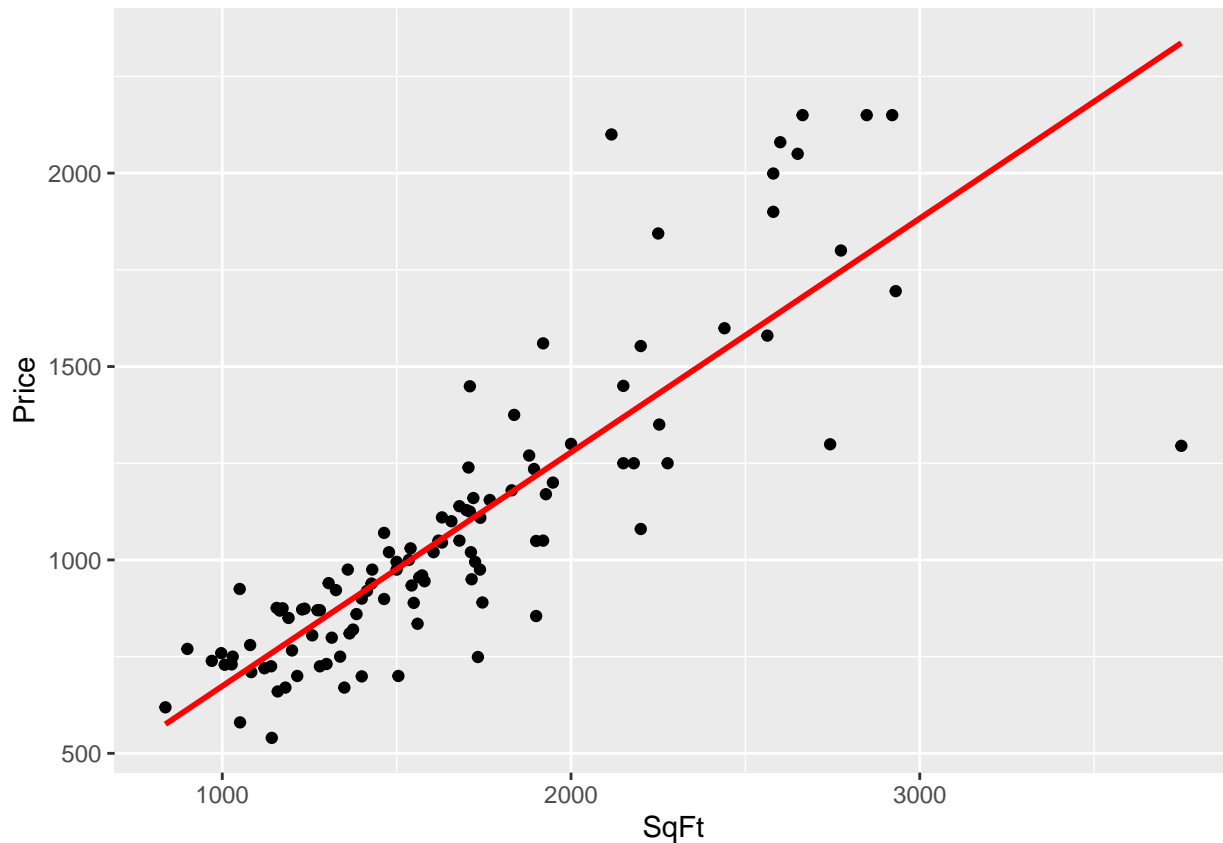


Question 2: Bootstrap, jackknife and confidence intervals (Solved by Satya Sai Naga Jaya Koushik Pilla)

Answer:

(2.1)

we create a scatter plot of SqFt versus Price, and fit a line using linear regression.



```
##
## Call:
## lm(formula = Price ~ SqFt, data = price_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
```

According to the output, the linear model is as follows:

$$\text{Price} = 69.30663 + 0.60457 * \text{SqFt}$$

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 and those points not close to the linear fit line.

(2.2)

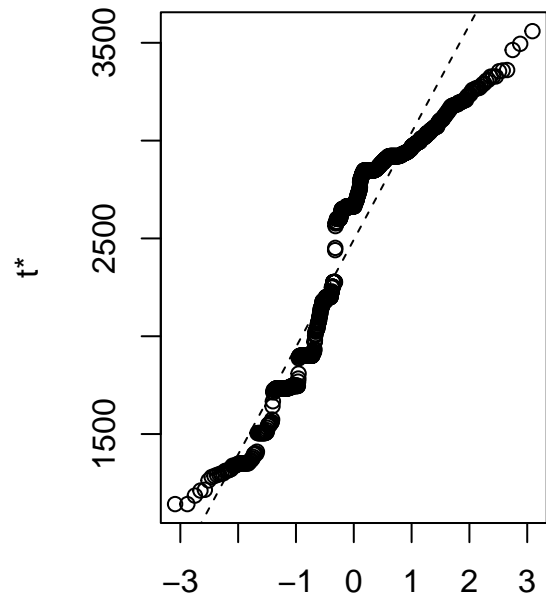
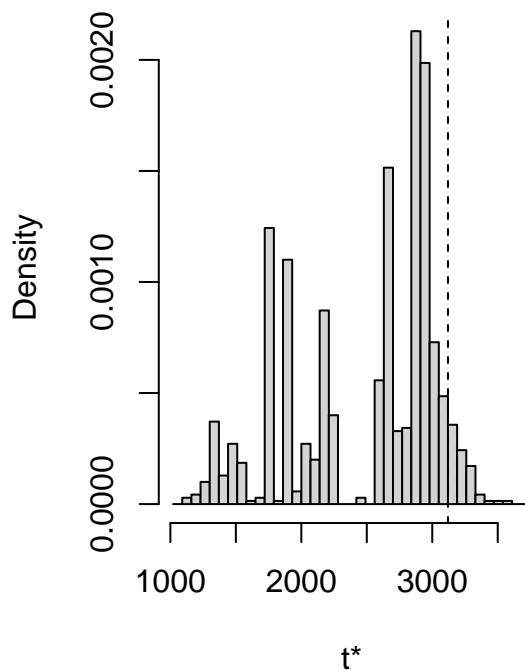
Please check code in the appendix for reference.

The result is as follows.

```
## The optimized b= -57.68284
## The optimized a1= 0.6871566
## The optimized a2= -1.942397
## The optimized c= 3119.801
## The optimized linear model is
## Price = -57.68284 +( 0.6871566 )* SqFt + ( -1.942397 )* (SqFt -( 3119.801 ))
## The optimized RSS= 3309413
```

(2.3)

Histogram of t



Quantiles of Standard Normal

```
## Bootstrap Variance of c: 301099.1
```

We estimate the distribution of c using bootstrap, the following is the result.

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_result, type = c("perc", "bca", "norm"))
##
## Intervals :
## Level      Normal          Percentile          BCa
## 95%  (2671, 4822 )  (1350, 3225 )  (3011, 3559 )
## Calculations and Intervals on Original Scale
## Warning : BCa Intervals used Extreme Quantiles
## Some BCa intervals may be unstable
```

According to the output of `boot.ci()`, we can find that the 95% confidence interval of c .
(1350, 3225) if we use bootstrap percentile.
(3011, 3559)if we use bootstrap BCa.
(2671, 4822) for normal approximation

(2.4)

Estimate the variance of c using the jackknife:

```
## Jackknife Variance of c: 32554.83
```

Compared with the bootstrap variance, the jackknife variance is little bigger than bootstrap variance. because jackknife only leave one observation out each time, when data is sensitive to outliers, jackknife method can create more variability.

While boot method resample with replacement, so it can reduce the effect of outliers.

From the scatter plot, we can find some outliers existed, which maybe the reason cause the higher variance of jackknife.

(2.5)

Compare the results of the confidence intervals from above, we find that BCa us the narrowest confidence interval, and normal is the widest confidence interval.

It's because wider confidence interval indicates higher uncertainty in the estimate of the parameter.

The BCa method is generally preferred when dealing with non-normal or skewed distributions, and it provides bias correction and accounts for skewness. In our case, several outliers existed, so the BCa method is more suitable.

The first-order normal approximation, however, assumes normality. since in our dataset, several outliers existed, so the normal approximation is not very accurate.

The percentile method is very sensitive to outliers, so it's not very accurate in our case.

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_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red") +
geom_smooth(method = "loess", formula = y ~ x, se = FALSE, color = "blue")
##### (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){
  # 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
  residuals <- rnorm(nrow(df_param_sample), mean = 0, sd = sd(resid(loess_param_sample_mod)))
  df_param_sample$Draft_No <- predict(loess_param_sample_mod) + residuals
  # get mean
  mean_parm_draft_no <- mean(df_param_sample$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$Draft_No, replace = TRUE)
    df_sample <- data.frame(Day_of_year = data$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 to 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_indexes <- start_dates:(start_dates + k - 1)

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

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

    # 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_indexes <- c()

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

```

```

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

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

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

# 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) #####
price_data <- read.csv("prices1.csv", sep = ";", header = TRUE)

ggplot(price_data, aes(x = SqFt, y = Price)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "red")

linear_model <- lm(Price ~ SqFt, data = price_data)
summary(linear_model)
rss_fun <- function(par,data) {

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

  SqFt <- data$SqFt
  Price <- data$Price

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

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

# Init
init_pars <- c(b = coef(linear_model)[1], a1 = coef(linear_model)[2], a2 = 0, c = 2000)

# Optimize using optim()
optim_result <- optim(par = init_pars, fn = rss_fun, data=price_data)

opt_pars <- optim_result$par
opt_rss <- optim_result$value

b = opt_pars[1]
a1 = opt_pars[2]
a2 = opt_pars[3]
c = opt_pars[4]

cat("The optimized b=", b, "\n")
cat("The optimized a1=", a1, "\n")
cat("The optimized a2=", a2, "\n")
cat("The optimized c=", c, "\n")

cat("The optimized linear model is", "\n")

```

```

cat("Price =", b, "+(", a1, ")* SqFt + (",
    a2, ")* (SqFt -(", c, "))\n")
cat("The optimized RSS=", opt_rss, "\n")
stat_c <- function(data, indices){
  boot_data = data[indices,]
  res <- optim(init_pars, rss_fun, data = boot_data)
  return(res$par[4])
}

set.seed(12345)
boot_result <- boot(price_data, statistic = stat_c, R=1000)
var_boot_c <- var(boot_result$t)

# plot boot_result
plot(boot_result)
cat("Bootstrap Variance of c:", var_boot_c, "\n")
boot.ci(boot_result, type=c("perc", "bca", "norm"))
# Jackknife Function
jackknife <- function(data, func) {
  n <- nrow(data)
  estimates <- numeric(n)

  for (i in 1:n) {
    data_sample <- data[-i,]
    optim_result <- optim(init_pars, func, data = data_sample)
    estimates[i] <- optim_result$par[4]
  }

  return(estimates)
}

# perform jackknife
jackknife_results <- jackknife(price_data, rss_fun)

# Print Jackknife Results
cat("Jackknife Variance of c:", var(jackknife_results), "\n")

```