Computational Statistics Computer Lab 5 (Group 7)

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

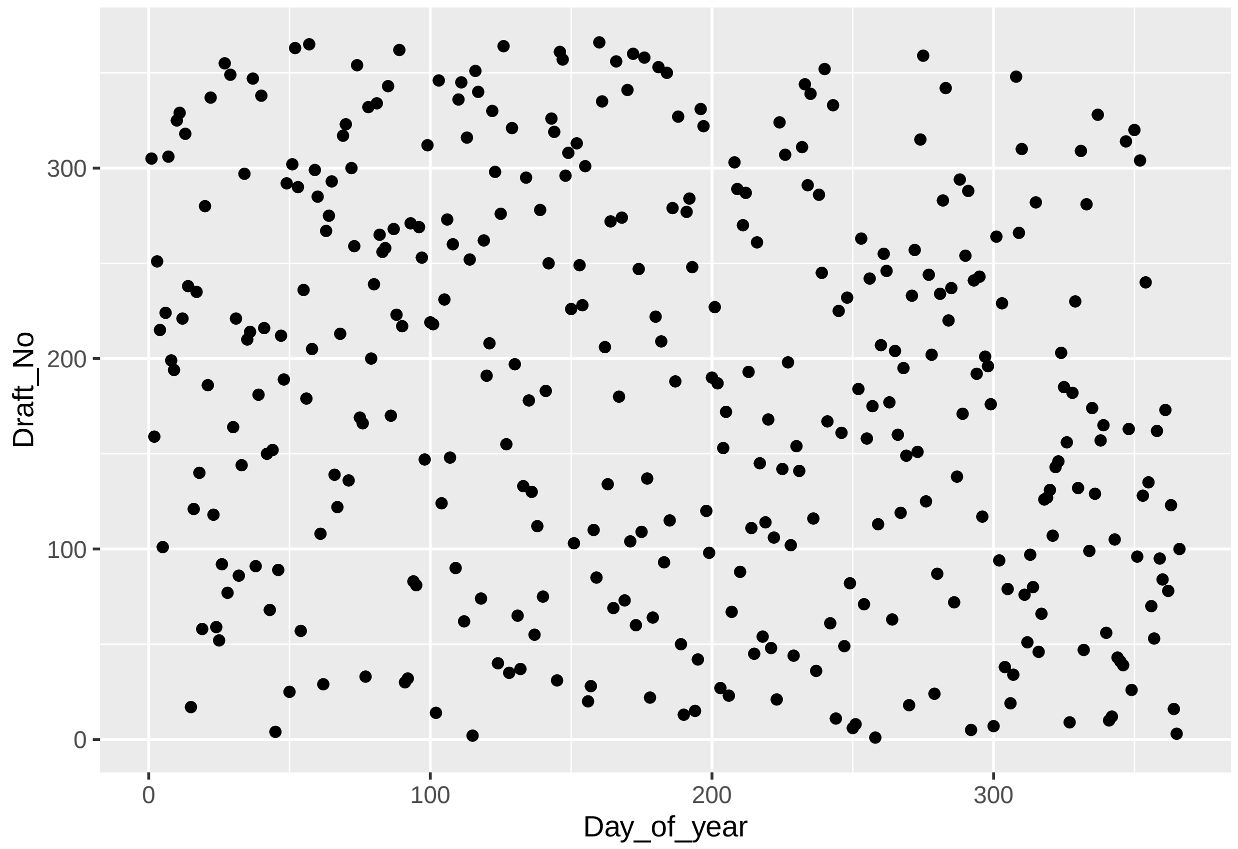
2023-12-12

Question 1: Hypothesis testing (Solved by Qinyuan Qi)

Answer:

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

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

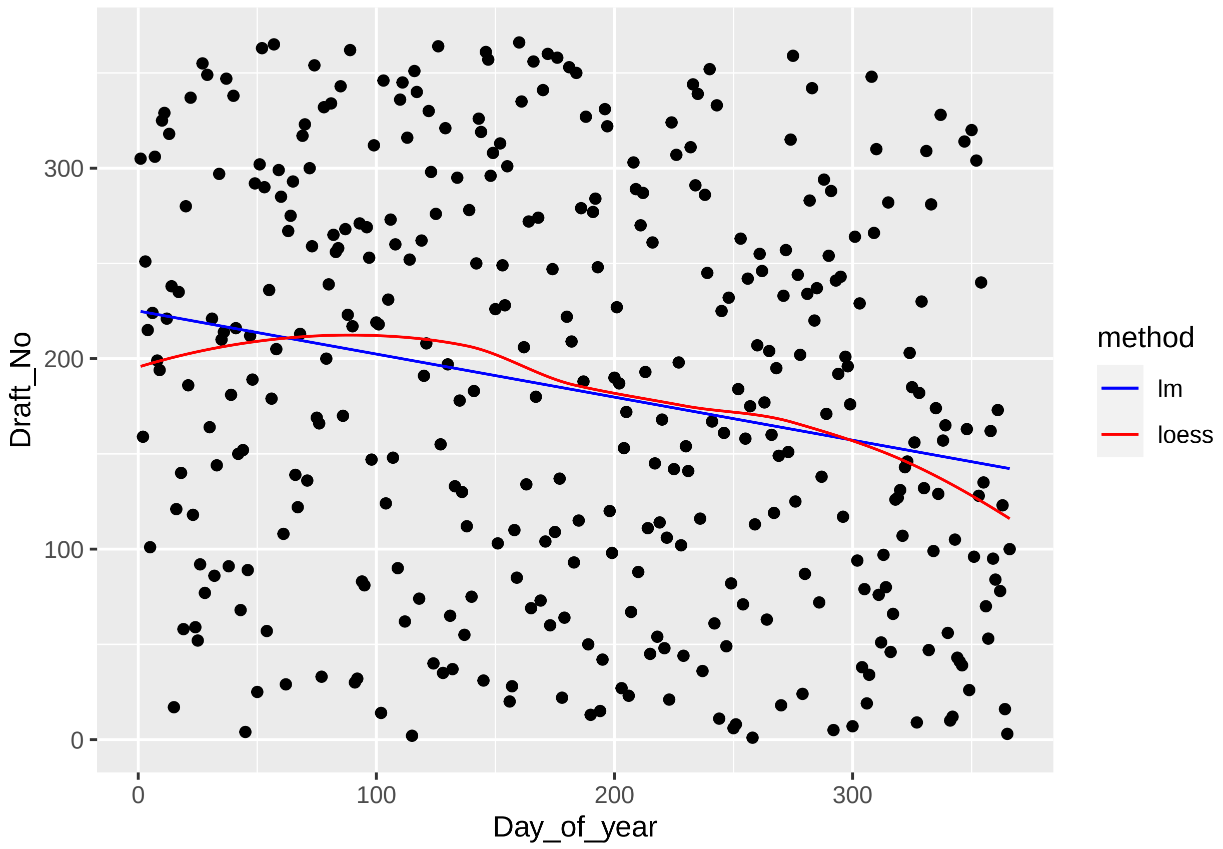


(1.2) Fit a curve to the data and conclude

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

According to the 2 curves, it seems that all the points are divided to 2 groups randomly. and we can guess that the points are randomly distributed.

According to the plot , we can not get the conclusion that which model(lm or loess) is better since only parameter (Day of year) is used. And according to the output of 2 models, output format is different, however, lm version's residual standard error is 103.2 and loess version's residual standard error is 103. according to this value , we can know that there have no big difference between these 2 versions.



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

The bootstrap value is generated using boot function. please check code in the appendix for detail.

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

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

The S value and p-value for lm and loess regression are generated by code.

Because S value is not close to zero, then this indicates some trend in the data, we throws suspicion on the randomness of the lottery dataset.

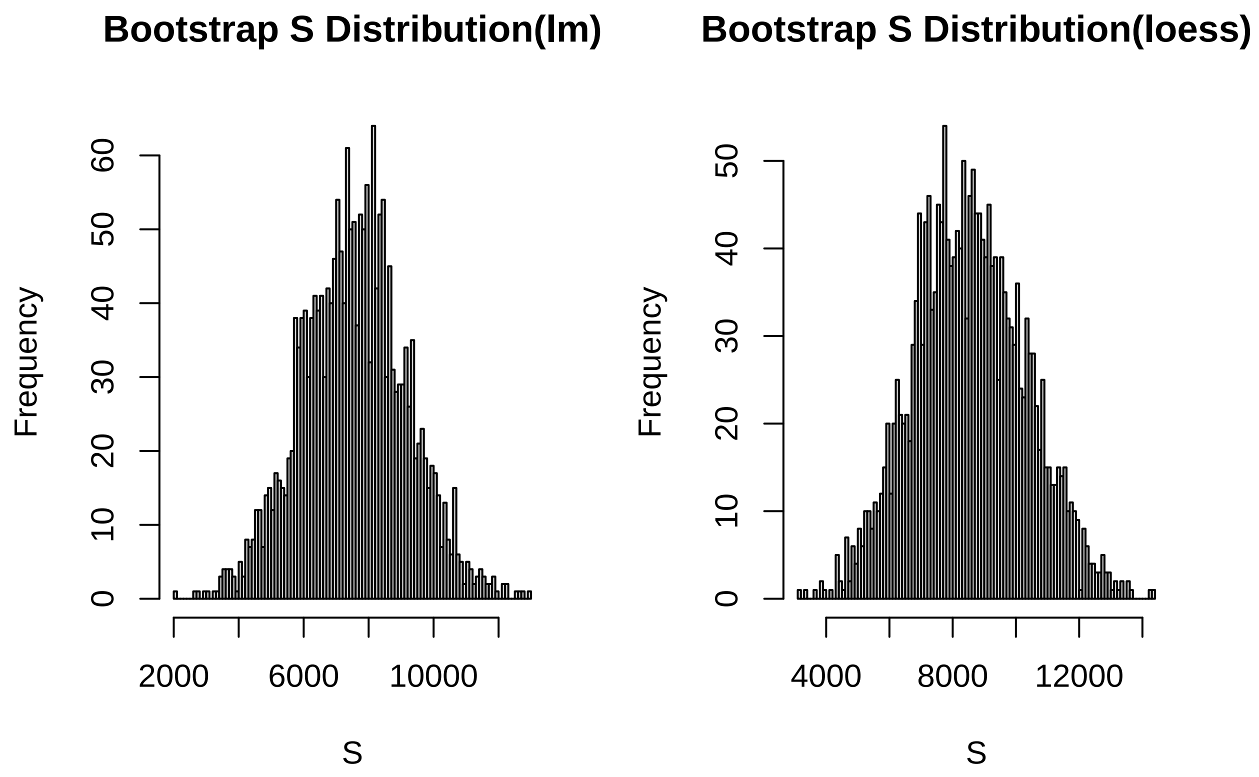
According to the plot, 2 plots are not symmetrical, also both of them not not follow the pattern of the known distributions, so we can say that it is not randomly distributed.

```
## mean of bootstrap sample statistics(lm) =  7551.223
```

```
## mean of bootstrap sample statistics(loess) =  8544.389
```

```
## p-value for linear regression =  0.492
```

```
## p-value for loess regression =  0.5525
```



(1.4)

The following code will generate the value of S and its p–value, based on 2000 bootstrap samples.

```
test_hypothesis <- function(data, B = 2000) {

  # bootstrap samples
  bootstrap_samples_lm <- boot(data = data, statistic = fun_lm_s, R = B,
                formula=Draft_No  ~ Day_of_year,parallel = "multicore")
  bootstrap_samples_loess <- boot(data = data, statistic = fun_loess_s, R = B,
                formula=Draft_No  ~ Day_of_year,parallel = "multicore")

  # p-values
  pvalue_lm = sum(boot_samples_lm$t >= boot_samples_lm$t0)/length(boot_samples_lm$t)
  pvalue_loess = sum(boot_samples_loess$t >= boot_samples_loess$t0)/length(boot_samples_loess$t)

  mean_boot_samples_lm <- mean(boot_samples_lm$t)
  mean_boot_samples_loess <- mean(boot_samples_loess$t)

  result <- list(
    s_lm_value = mean_boot_samples_lm,
    p_lm_value = pvalue_lm,
    s_loess_value = mean_boot_samples_loess,
    p_loess_value = pvalue_loess
  )
  return(result)
}
```

(1.5)

We will generate a dataset of the same dimensions as the original data.

```
#################################### (1.5.a) #########################################
#(a) generate a dataset of the same dimensions as the original data.

gen_df_1_5 <- function (k){
  df_1_5 <- data.frame(Day_of_year = 1:k, Draft_No = sample(min(data$Draft_No):max(data$Draft_No),1))

  df_1_5 <- rbind(data[k+1,],df_1_5)
  return(df_1_5)
}
```

If we set k=9, and alpha = 0.05, compare pvalue and alpha we can accept or reject hypothesis.

According to the result, in the case of lm, we fail to reject H0 hypothesis, which is Lottery is random.

```
#################################### (1.5.b) #########################################
alpha <- 0.05
k <- 9
data_random <- gen_df_1_5(k)
result <- test_hypothesis(data = data_random)
cat("P-value(lm):", result$p_lm_value, "\n")
```

```
## P-value(lm): 0.492
```

```
cat("P-value(loess):", result$p_loess_value, "\n")
```

```
## P-value(loess): 0.5525
```

If we set k=10 to 50, and skip all k %% 3 != 0,we got 13 Fail to Reject H0 Hypothesis, and not got any Reject case.

```
#max_k = dim(data)[1]
max_k = 50

for(k in 10:max_k){
  if (k %% 3 != 0) {
    next
  }
  data_random <- gen_df_1_5(k)

  res <- test_hypothesis(data = data_random)
  # we only test p_loess_value here.
  if(result$p_loess_value <= alpha ){
    cat("Reject H0 Hypothesis","\n")
  }else{
    cat("Fail to Reject H0 Hypothesis","\n")
  }
}
```

```
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
## Fail to Reject H0 Hypothesis
```

According to the above, we know that Lottery is random.

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

```r
################################### (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)
```

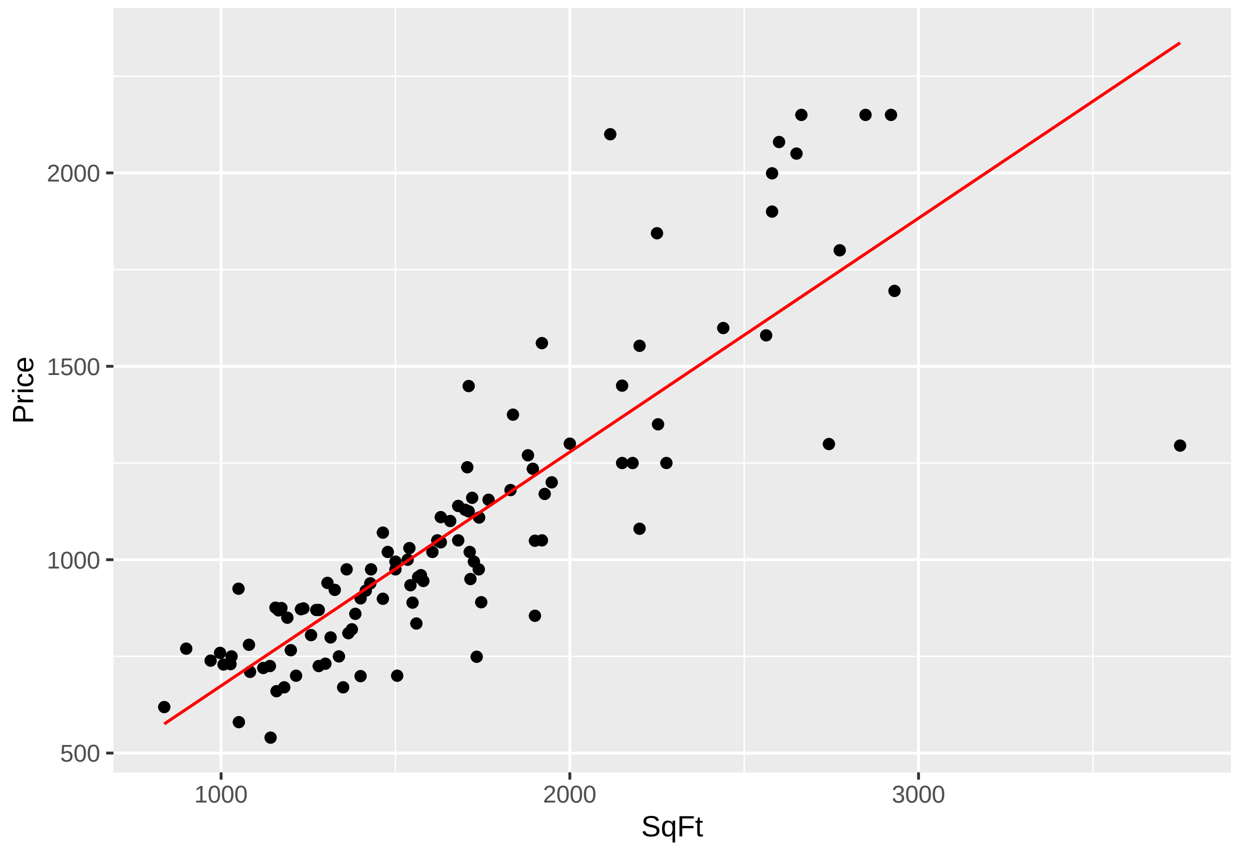
```
##
## Call:
## lm(formula = Price ~ SqFt, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1041.43   -99.12     1.99    59.26   751.43
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 69.30663   65.13461   1.064     0.29
## SqFt         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)
}
```

```r
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))
}


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
```

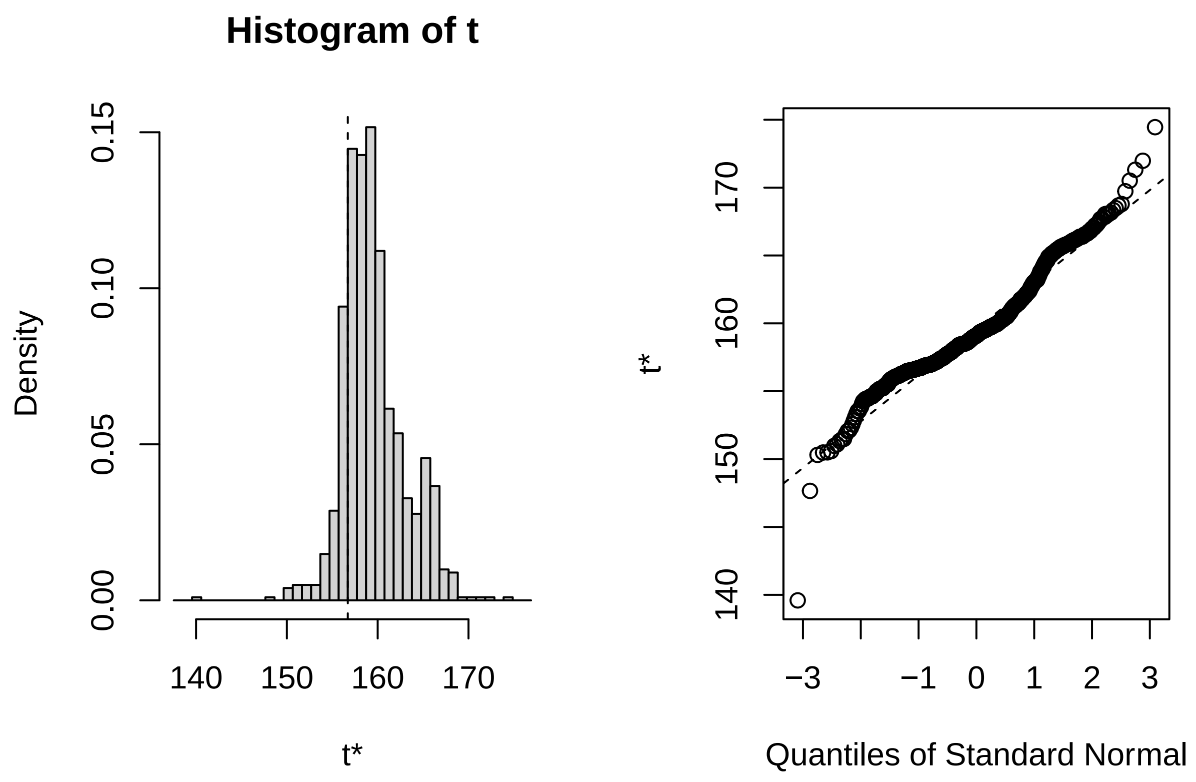
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data, statistic = stat1, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 156.7114 2.875368    3.410341
```

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
```

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



(2.4)

Estimate the variance of c using the jackknife:

```r
# 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))
}


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, se = TRUE)
#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) #######################################
# boot function fot lm
fun_lm_s <- function(formula,data,indices){
  d <- data[indices, ]
  lm_mod <- lm(formula, data = d)
  lm_val <- predict(lm_mod,
                    newdata = data.frame(Day_of_year = d$Day_of_year))
  s_lm <- sum(abs(lm_val - mean(d$Draft_No)))
  return(s_lm)
}

# boot function fot loess
```

```r
fun_loess_s <- function(formula,data,indices){
  d <- data[indices, ]
  loess_mod <- loess(formula, data = d, se = TRUE)
  loess_val <- predict(loess_mod,
                    newdata = data.frame(Day_of_year = d$Day_of_year))
  s_loess <- sum(abs(loess_val - mean(d$Draft_No)))
  return(s_loess)
}

bootstrap_sample_number <- 2000

# bootstrap for linear regression
boot_samples_lm <- boot(data, statistic = fun_lm_s, R = bootstrap_sample_number, formula=Draft_No  ~ Day

boot_samples_loess <- boot(data, statistic = fun_loess_s, R = bootstrap_sample_number, formula=Draft_No

# calculate mean of bootstrap sample statistics
mean_boot_samples_lm <- mean(boot_samples_lm$t)
cat("mean of bootstrap sample statistics(lm) = ", mean_boot_samples_lm, "\n")
#mean of bootstrap sample statistics(lm) =  7646.339

mean_boot_samples_loess <- mean(boot_samples_loess$t)
cat("mean of bootstrap sample statistics(loess) = ", mean_boot_samples_loess, "\n")

pvalue_lm = sum(boot_samples_lm$t >= boot_samples_lm$t0)/length(boot_samples_lm$t)
pvalue_loess = sum(boot_samples_loess$t >= boot_samples_loess$t0)/length(boot_samples_loess$t)

cat("p-value for linear regression = ", pvalue_lm, "\n")
cat("p-value for loess regression = ", pvalue_loess, "\n")

# plot 2 hist using 1*2 layout
par(mfrow = c(1,2))

# Plot the histogram of bootstrap sample statistics(lm)
hist(boot_samples_lm$t, main = "Bootstrap S Distribution(lm)", xlab = "S", breaks=100)

# Plot the histogram of bootstrap sample statistics(loess)
hist(boot_samples_loess$t, main = "Bootstrap S Distribution(loess)", xlab = "S", breaks=100)

test_hypothesis <- function(data, B = 2000) {

  # bootstrap samples
  bootstrap_samples_lm <- boot(data = data, statistic = fun_lm_s, R = B,
                formula=Draft_No  ~ Day_of_year,parallel = "multicore")
  bootstrap_samples_loess <- boot(data = data, statistic = fun_loess_s, R = B,
                formula=Draft_No  ~ Day_of_year,parallel = "multicore")

  # p-values
  pvalue_lm = sum(boot_samples_lm$t >= boot_samples_lm$t0)/length(boot_samples_lm$t)
  pvalue_loess = sum(boot_samples_loess$t >= boot_samples_loess$t0)/length(boot_samples_loess$t)

  mean_boot_samples_lm <- mean(boot_samples_lm$t)
  mean_boot_samples_loess <- mean(boot_samples_loess$t)
  result <- list(
    s_lm_value = mean_boot_samples_lm,
    p_lm_value = pvalue_lm,
    s_loess_value = mean_boot_samples_loess,
    p_loess_value = pvalue_loess
  )
  return(result)
}

################################### (1.5.a) #######################################
#(a) generate a dataset of the same dimensions as the original data.

gen_df_1_5 <- function (k){
  df_1_5 <- data.frame(Day_of_year = 1:k, Draft_No = sample(min(data$Draft_No):max(data$Draft_No),1))

  df_1_5 <- rbind(data[k+1,],df_1_5)
  return(df_1_5)
}

################################### (1.5.b) #######################################
alpha <- 0.05
k <- 9
data_random <- gen_df_1_5(k)
result <- test_hypothesis(data = data_random)
cat("P-value(lm):", result$p_lm_value, "\n")
cat("P-value(loess):", result$p_loess_value, "\n")

#max_k = dim(data)[1]
max_k = 50

for(k in 10:max_k){
  if (k %% 3 != 0) {
    next
  }
  data_random <- gen_df_1_5(k)

  res <- test_hypothesis(data = data_random)
  # we only test p_loess_value here.
  if(result$p_loess_value <= alpha ){
    cat("Reject H0 Hypothesis","\n")
  }else{
    cat("Fail to Reject H0 Hypothesis","\n")
  }
}

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

```r
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))
}


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)
```

```r
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))
}


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")
```