STA457 Assignment 2

Depeng Ye 1002079500

28/10/2019

Problem 1

(1)

First read the file:

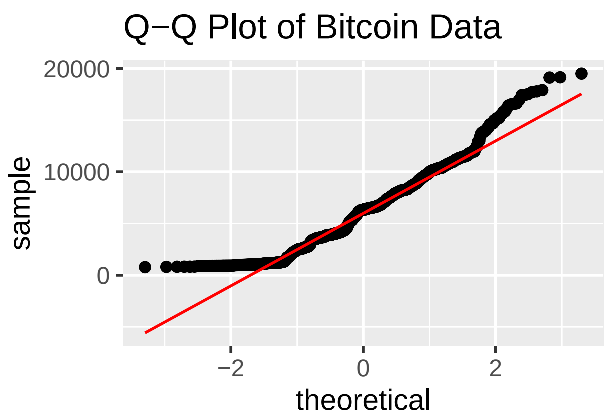
```
BTC = read.csv("BTC.csv", header = T)
#refine the closing prices (Close.USD) by removing the 1000 separators
BTC$Close.USD = as.numeric(gsub(",","",BTC$Close.USD))
```

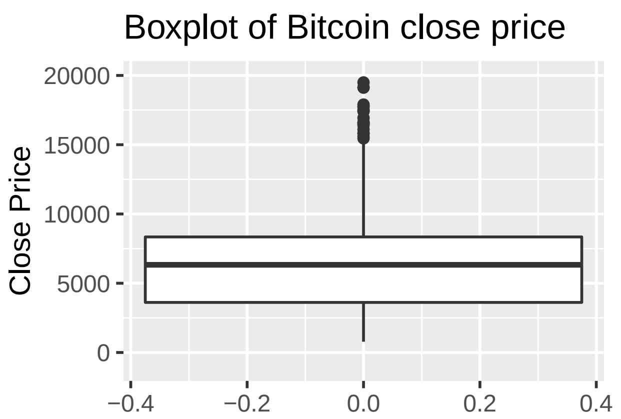
Notice that the time series is from January 1st, 2017 to October 18th, 2019. Looking into the data: the 2017.1.1 data is in 1021th row. Subsetting the BTC data as follows:

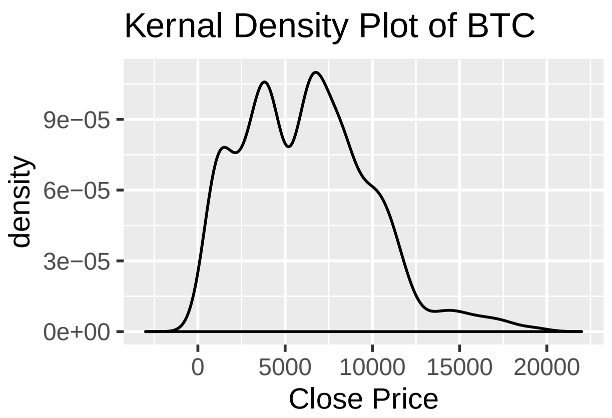
```
#Exacting the 3 years' data
BTC = BTC[1:1021,]
```

Then do the plot:

```
require(gridExtra)
#qq plot
p1<- ggplot(BTC, aes(sample = Close.USD)) + stat_qq() +
  stat_qq_line(col = 2) + labs(title = "Q-Q Plot of Bitcoin Data")
#boxplot
p2<- ggplot(BTC, aes(y = BTC$Close.USD)) + geom_boxplot() +
  labs(title = "Boxplot of Bitcoin close price", y = "Close Price") +
  ylim(c(-1000, 20000))
#kernal density plot
p3<- ggplot(data = BTC, aes(x = Close.USD)) + geom_density() +
  labs(title = "Kernal Density Plot of BTC", x = "Close Price") + xlim(c(-3000, 22000))
grid.arrange(p1, p2, p3, ncol = 2)
```







Considering that the Q-Q plot is has a convex pattern (and somewhat convex-concave) with the x-axis being theoretical quantile (i.e. normal quantile). It is easy to notice that the Bitcoin Close price is not normally distributed. Instead, the data distribution should appear to be light tailed and could have a right skewness as well.

Considering the boxplot, the first and the third quantile are in the smaller half of the data set, while the median is slightly closer to the third quantile. Also, there are several noticiable observatins beyond the whiskers in the higher price end. Hence, we can predict that the distribution is skewed to the right.

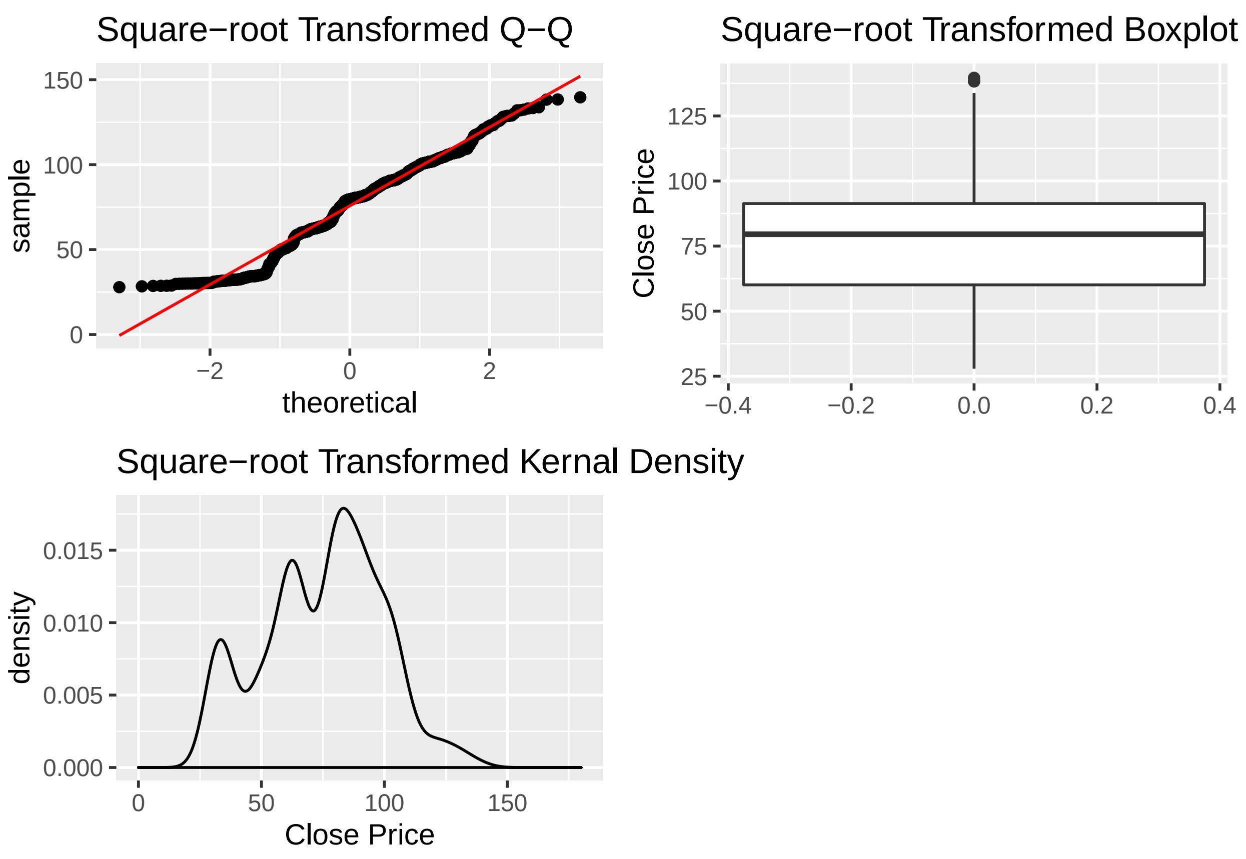
Looking into the Kernal density plot. Fist of all we can confirm that this data is right skewed. We could also see that the left tail is a lot heavier than the right tail for the reason that there are several observations in the higher price end of the data set. The distribution of BTC data is obviously not symmetric.

(2)

Perform the square root data transformation:

```
# Square-root Transformation
sqrt.BTC = sqrt(BTC$Close.USD)
sqrt.BTC = data.frame(sqrt.BTC)

require(gridExtra)
#qq plot
p4<- ggplot(sqrt.BTC, aes(sample = sqrt.BTC)) + stat_qq() +
  stat_qq_line(col = 2) + labs(title = "Square-root Transformed Q-Q")
#boxplot
p5<- ggplot(sqrt.BTC, aes(y = sqrt.BTC)) + geom_boxplot() +
  labs(title = "Square-root Transformed Boxplot", y = "Close Price")
```

```r
#kernal density plot
p6<- ggplot(data = sqrt.BTC, aes(x = sqrt.BTC)) + geom_density() +
  labs(title = "Square-root Transformed Kernal Density", x = "Close Price") +
  xlim(c(0, 180))

grid.arrange(p4, p5, p6, ncol = 2)
```



After the Square-root Transformation, the new data sqrt.BTC seems quite similar to the original data.

The Q-Q plot appears to be non-normal. To be exact, the smaller half is non-normal, while the larger half seems to be normal. It seems to have a convex-concave pattern, which means that the distribution is light tailed.

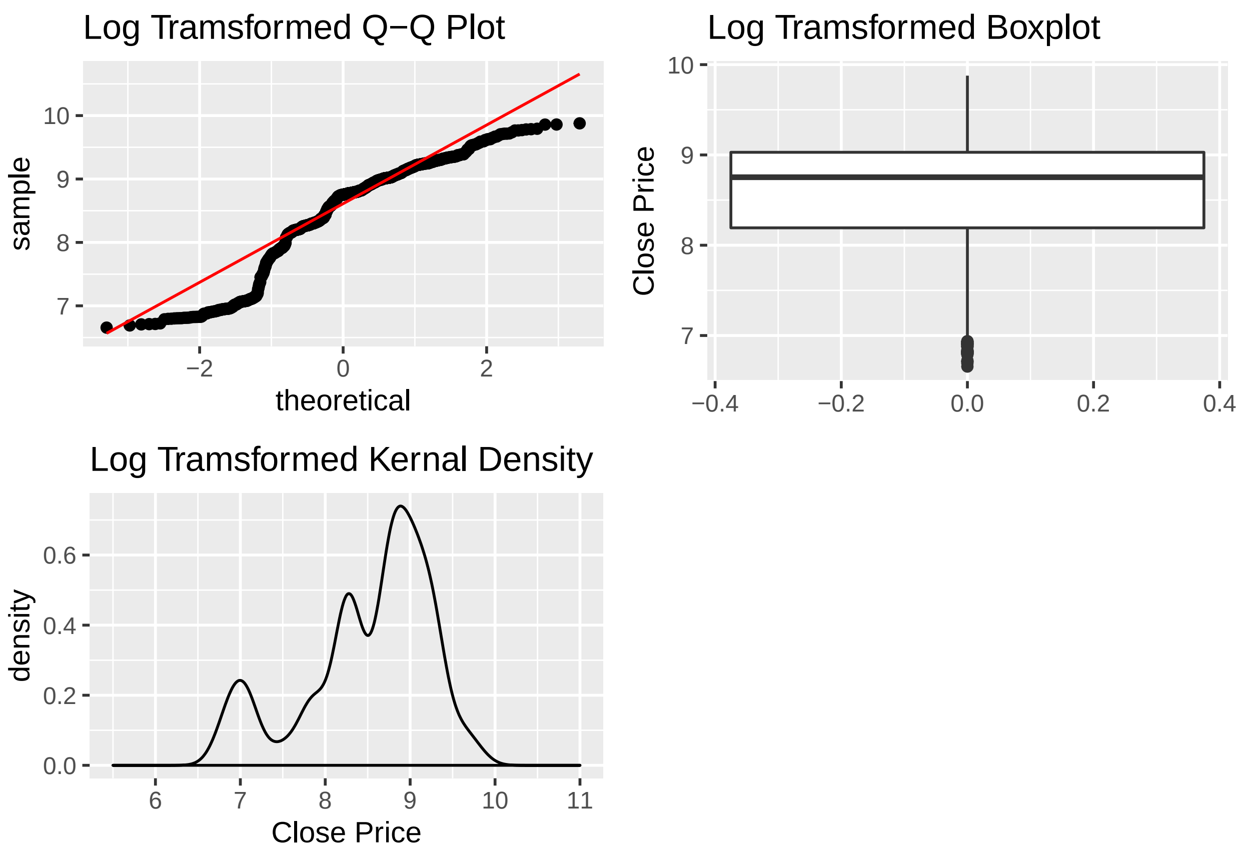
The boxplot seems to have the first and third quantile in the middle of this data set, but still having the median close to the third quantile, as well as several outlier obsevations in the higher price end of the square root transformed data set.

The kernal density estimate is light tailed but is more symmrtric when compared to the un-transformed data. The left tail appears to be lighter than right tail. The left half seems very fluctuating, while the right half can be regarded as normal distribution according to the Q-Q plot.

Perform the logarithm data transformation:

```r
#Logarithm Transformation
log.BTC = log(BTC$Close.USD)
log.BTC = data.frame(log.BTC)

require(gridExtra)
#qq plot
```

```r
p7<- ggplot(log.BTC, aes(sample = log.BTC)) + geom_qq() +
  stat_qq_line(col = 2) + labs(title = "Log Tramsformed Q-Q Plot")
#boxplot
p8<- ggplot(log.BTC, aes(y = log.BTC)) + geom_boxplot() +
  labs(title = "Log Tramsformed Boxplot", y = "Close Price")
#kernal density plot
p9<- ggplot(data = log.BTC, aes(x = log.BTC)) + geom_density() +
  labs(title = "Log Tramsformed Kernal Density", x = "Close Price") + xlim(c(5.5, 11))

grid.arrange(p7, p8, p9, ncol = 2)
```



Looking into the data after the logarithm transformation log.BTC.

First take a look at the Q-Q plot. It is non-normal. The pattern is more similar to convex-concave with the concave part a lot larger than the convex part, which implies to a light tailed and left skewed pattern in the data distribution.

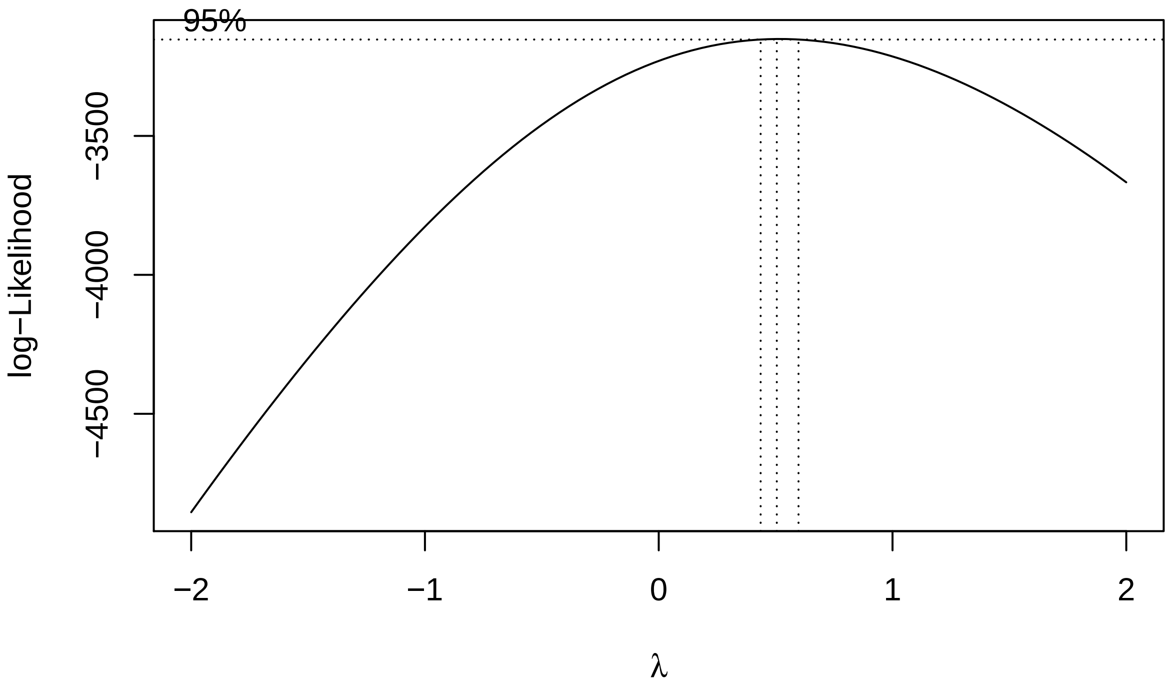
In the boxplot, the first and the third quatile is more on the larger end of the log-transformed BTC data. The median is close to the third quantile.

In the kernal density curve notice that the distribution is not quite symmetric, while there appears to be left skewed and light tailed. The left tail appears to be lighter than the right tail. The left tail is very fluctuating as well.

(3)

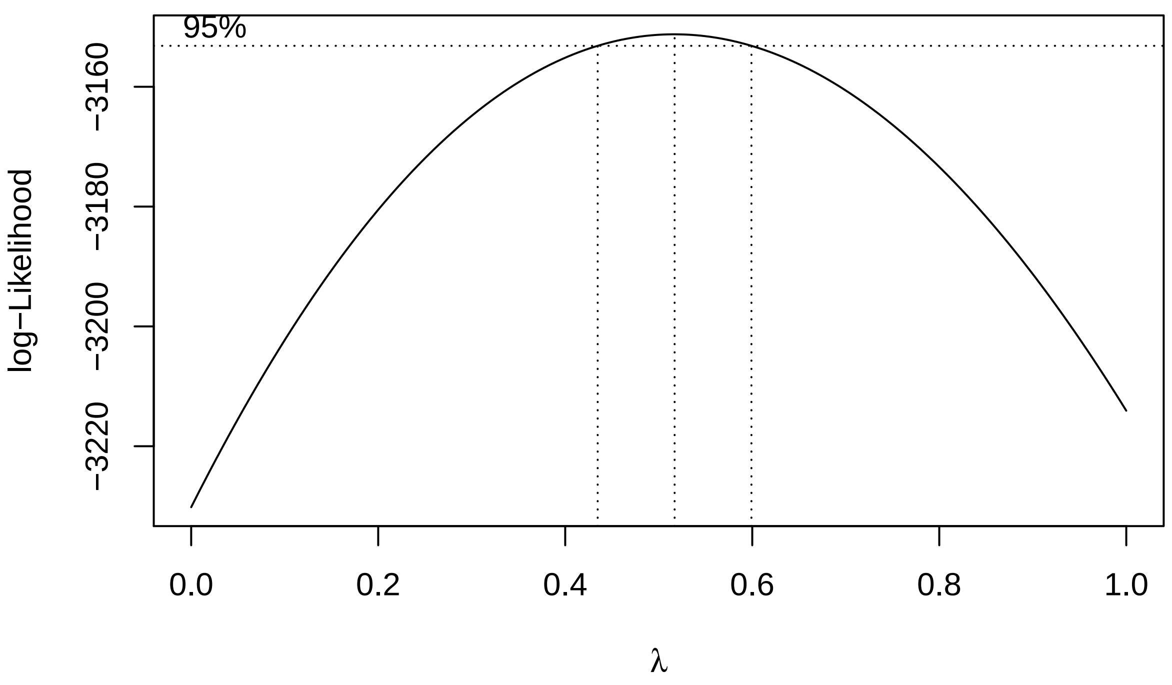
Box-cox Transformation

```r
#Box-cox transfermation
p10 <- boxcox(BTC$Close.USD ~ 1)
```



Zoom in and calculate λ:

```r
#find the MLE lambda using zoomed in box-cox plot
p11 <- boxcox(BTC$Close.USD~1,lambda=seq(0, 1, by = 1/1000), interp = F)
```



```
index = (p11$y == max(p11$y))
p11$x[index]
```

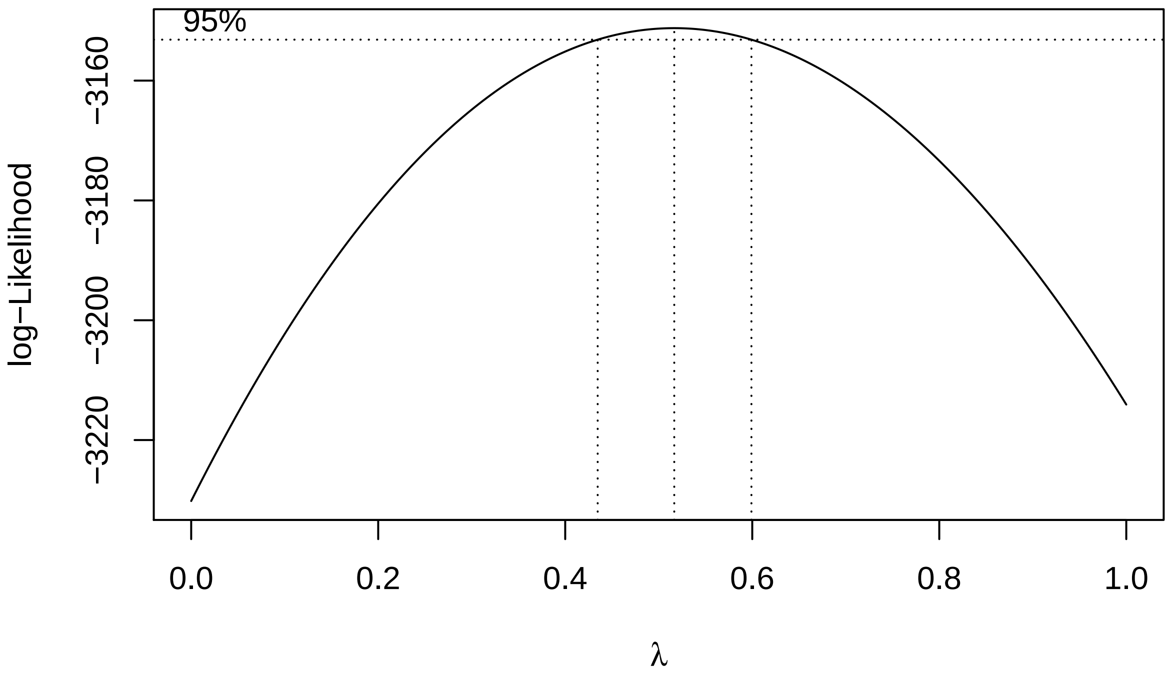
```
## [1] 0.517
```

Hence, the MLE of λ is 0.517.

(4)

Calculate the 99% Confidence Interval using the code below:

```
p12 <- boxcox(BTC$Close.USD~1,lambda=seq(0, 1, by = 1/10000), interp = F)
```



```
index2 = (p12$y > max(p12$y) - qchisq(0.99, df = 1) / 2)
min(p12$x[index2])
```

```
## [1] 0.4092
```

```
max(p12$x[index2])
```

```
## [1] 0.6252
```

Hence the 99% CI for λ is $(0.409, 0.625)$.

(5)

Fit a skewed t-distribution using package fGarch:

```
tFit = sstdFit(BTC$Close.USD, hessian = TRUE)
```

(6)

```
tFit$estimate
```

```
##         mean           sd           nu           xi
## 6.000046e+03 3.905672e+03 1.022011e+05 9.397228e+00
```

mean estimate is 6000.04, standard deviation estimate is 3905.67, DF estimate is 102201 and x_i estimate is 9.3972.

Problem 2

Read the data, and remove the NA rows.

```
LBMA_GOLD = read.csv("LBMA-GOLD.csv", header = T)
#removing the rows with missing value
LBMA_GOLD = na.omit(LBMA_GOLD)
```

Slicing the data to a time period of Jan 3rd, 2017 to Oct 18th, 2019 (we use from the first row to the 704th row of the data). Because of the na.omit operation does not change the label on rows, the 704th row still appears to be labeled as 708.

```
#row number 704 of data
LBMA_GOLD[704,]
```

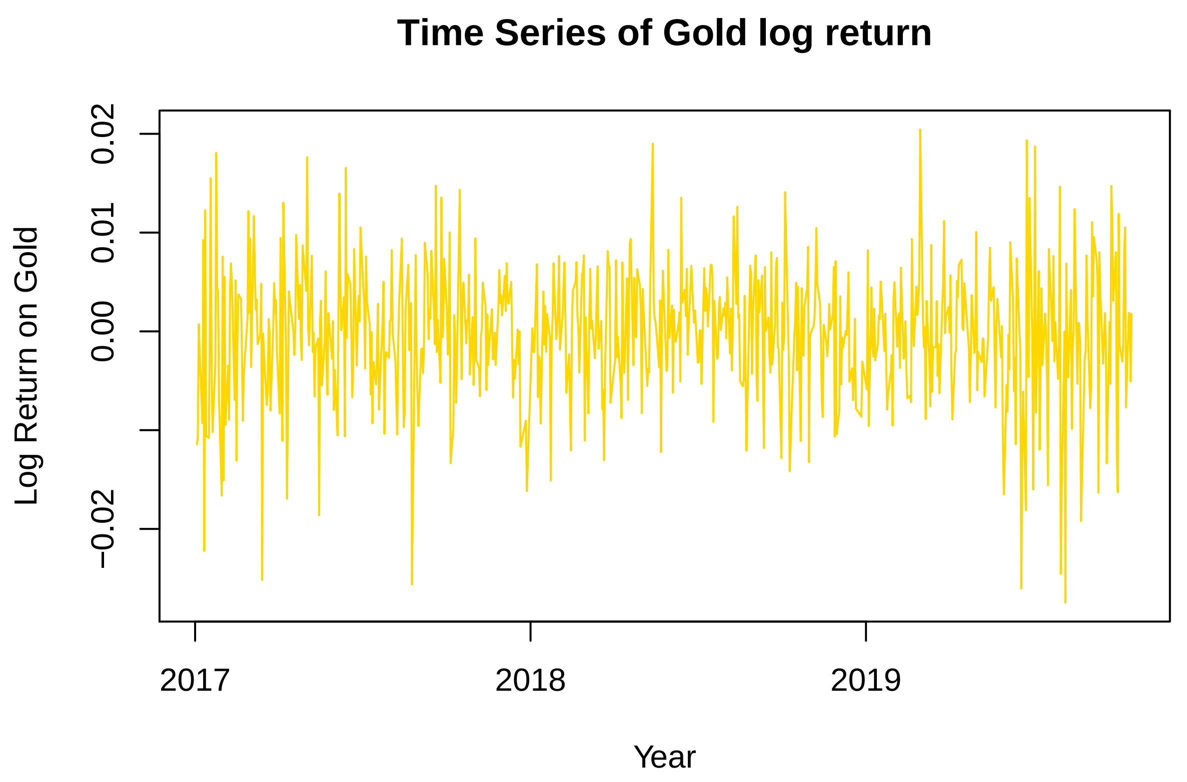
```
##           Date USD..PM.
## 708 2017-01-03     1151
```

```
#slicing
LBMA_GOLD = LBMA_GOLD[1:704,]
```

(1)

Calculate the log return and add to LBMA_GOLD data as column Y. For the last date in this data set, put in NA.

```
LBMA_GOLD$Y = diff(c(NA, log(LBMA_GOLD$USD..PM.)))
LBMA_GOLD$Date = as.Date(LBMA_GOLD$Date, format = "%Y-%m-%d")

#plot the time series of Gold Price log-return
plot(LBMA_GOLD$Y~LBMA_GOLD$Date, ylab = "Log Return on Gold",
     xlab = "Year", col = "gold", type = "l",
     main = "Time Series of Gold log return")
```



The time series of the log return of gold from the starting of 2017 to mid October of 2019 has been plotted above.

We can say that the series looks stationary.

The fluctuations in the series seem to be of a smaller constant interval in the center part of time period (from the beginning of 2018 to the first half of 2019). Nevertheless, in 2017 and the second half of 2019, the fluctuation seems to be larger than priorly the metioned time period but still lies in a noticiable range. Hence, we could say that in a short time period, the fluctuation is constant.
In the long run, the volatility of fluctuation is large but lies in a certain range.

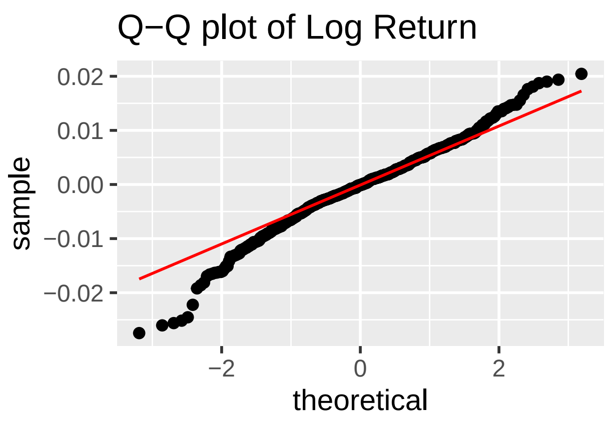
(2)

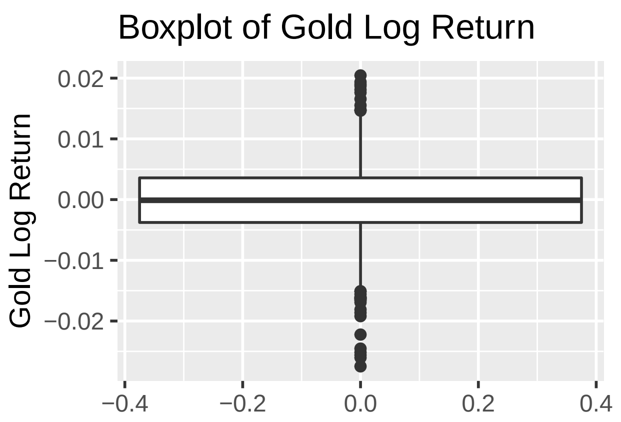
Removing the N/A row in LBMA_GOLD.

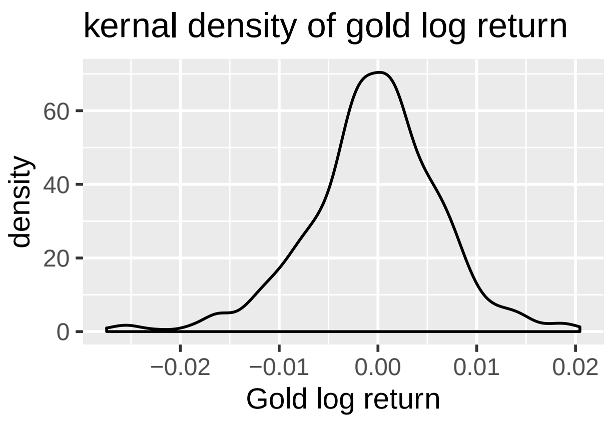
```r
LBMA_GOLD = na.omit(LBMA_GOLD)
```

Draw the plots.

```r
require(gridExtra)
p13 <- ggplot(LBMA_GOLD, aes(sample = LBMA_GOLD$Y)) + geom_qq() +
  stat_qq_line(col = 2) + labs(title = "Q-Q plot of Log Return")

p14<- ggplot(LBMA_GOLD, aes(y = LBMA_GOLD$Y)) + geom_boxplot() +
  labs(y = "Gold Log Return", title ="Boxplot of Gold Log Return")

p15<- ggplot(LBMA_GOLD, aes(x = LBMA_GOLD$Y)) + geom_density() +
  labs(title = "kernal density of gold log return", x = "Gold log return")

grid.arrange(p13, p14, p15, ncol = 2)
```







Taking a look at the Q-Q plot, we could know that the data is mostly normally distributed with a slight light tailed and slight left skewed pattern.
The box plot tells us that the first and third quantile are in the middle of the data. The median appears to be in the middle of first and third quantile. Also note that there are several observations out of the whiskers on both end of the data.
The kernal density plot tells us the distribution of Log Return on Gold is slightly light tailed and slightly skewed to the left when compared with normal distribution. But in general, it is symmetric.

(3)

```
log_return = LBMA_GOLD$Y
#Fit the standardized t-distribution
stFit = stdFit(log_return)
stFit
```

```
## $par
##          mean            sd            nu
## -0.0001597275  0.0067803631  5.5603947616
##
## $objective
## [1] -2538.391
##
## $convergence
## [1] 0
##
```

```
## $iterations
## [1] 32
##
## $evaluations
## function gradient
##       69      143
##
## $message
## [1] "relative convergence (4)"
```

```
#store the estimated value from stFit
mu = stFit$par["mean"]
sigma = stFit$par["sd"]
nu = stFit$par["nu"]

v = c(mean(log_return), sd(log_return), 5)
#create the log-likelihood function
loglik_std_t <- function(beta) {
  sum(- dstd(log_return, mean = beta[1], sd = beta[2], nu = beta[3], log = TRUE))
  }

MLE_std <- optim(v, loglik_std_t, hessian = T)
MLE_std
```

```
## $par
## [1] -0.0001599937  0.0067813205  5.5591797244
##
## $value
## [1] -2538.391
##
## $counts
## function gradient
##      216       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##              [,1]         [,2]        [,3]
## [1,] 1.823099e+07   448824.584  408.269814
## [2,] 4.488246e+05 22155310.434 2859.621935
## [3,] 4.082698e+02     2859.622    1.125411
```

Based on the output of optim() function, the MLE estimate of mean is -0.00016, MLE estimate of standard deviation is 0.0067, and the MLE estiamte of degrees-of-freedom is 5.559.

(4)

```r
AIC_std <- MLE_std$value * 2 + 2 * 3
AIC_std
```

```
## [1] -5070.782
```

```r
BIC_std <- MLE_std$value * 2 + log(length(log_return)) * 3
BIC_std
```

```
## [1] -5057.116
```

AIC based on the t-distribution estimate is -5070.782.
BIC based on the t-distribution estimate is -5057.116.

(5)

Calculate the skewed t-distribution MLE estimator using similar method as in **(3)**.

```r
v1 = c(mean(log_return), sd(log_return), 5, 1.5)
loglik_sstd_t <- function(beta) {
  sum(- dsstd(log_return, mean = beta[1], sd = beta[2], nu = beta[3],
              xi = beta[4], log = TRUE))
  }
MLE_sstd <- optim(v1, loglik_sstd_t, hessian = T)
MLE_sstd
```

```
## $par
## [1] -0.0003495992  0.0067689137  5.7695084589  0.9065083497
##
## $value
## [1] -2540.236
##
## $counts
## function gradient
##      237       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##                [,1]         [,2]         [,3]          [,4]
## [1,] 18317614.5968  2760143.581  300.7691839 -35268.578160
## [2,]  2760143.5810 22985817.343 2628.0551191   7866.168409
## [3,]      300.7692     2628.055    0.9485663      2.006786
## [4,]   -35268.5782     7866.168    2.0067857    550.962307
```

```r
AIC_sstd <- MLE_sstd$value * 2 + 2 * 4
AIC_sstd
```

```
## [1] -5072.471
```

```r
BIC_sstd <- MLE_sstd$value * 2 + log(length(log_return)) * 4
BIC_sstd
```

```
## [1] -5054.25
```

The MLE for skewed t-distribution are:
mean estimate: -0.00035 standard deviation estimate: 0.00068 degrees-of-freedom estimate: 5.76951 x_i estimate: 0.90651

AIC based on skewed t-distribution is -5072.471.
BIC based on skewed t-distribution is -5054.25.

(6)

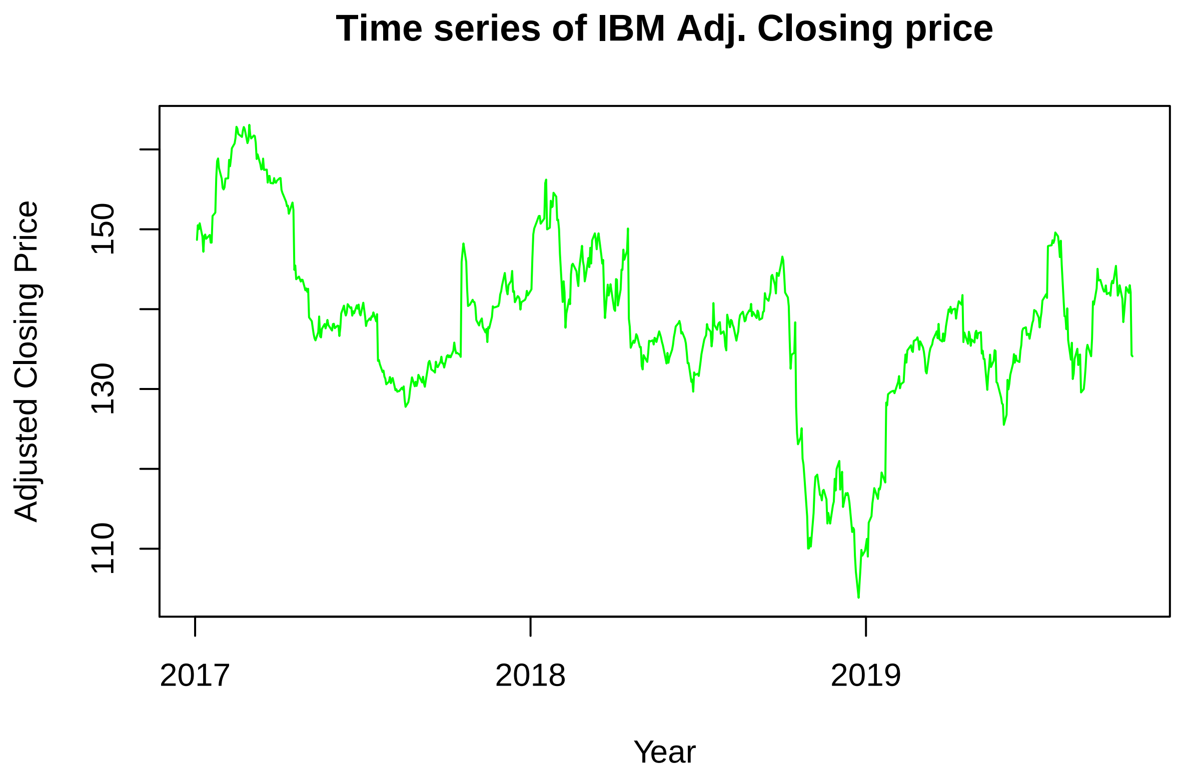
AIC_std > AIC_sstd implies that AIC selects skewed t-distribution.
BIC_std < BIC_sstd implies that BIC selects t-distribution without skewness.

This result proves the note in the lecture slides that "BIC tends to select simpler models (with less numbers of parameters) than AIC for data set with $n > 8$".

Problem 3

First read the data, and then exacting it the the time period of Jan 3rd, 2017 to Oct 18th, 2019.
Jan 3rd, 2017 observation appears in row 704, subsetting the data into 704 observations.

```r
IBM = read.csv("IBM.csv", header = T)
IBM = IBM[1:704,]
IBM$Date = as.Date(IBM$Date, "%Y-%m-%d")
plot(IBM$Adj.Close~IBM$Date, ylab = "Adjusted Closing Price",
     xlab = "Year", type = "l",
     main = "Time series of IBM Adj. Closing price", col = "green")
```



Notice that the time series of IBM adjusted close price is not stationary.

(1)

Calculate the sample mean, standard deviation, skewness and kurtosis.

```r
mean(IBM$Adj.Close)
```

```
## [1] 138.1832
```

```r
sd(IBM$Adj.Close)
```

```
## [1] 10.39769
```

```r
#skewness
skewness(IBM$Adj.Close)
```

```
## [1] -0.3204336
## attr(,"method")
## [1] "moment"
```

```r
# or use formula
Sk = (sum((IBM$Adj.Close - mean(IBM$Adj.Close))^3) /
    length(IBM$Adj.Close)) / sd(IBM$Adj.Close)^3
Sk
```

```
## [1] -0.3204336
```

```
#kurtosis
kurtosis(IBM$Adj.Close)
```

```
## [1] 0.9337797
## attr(,"method")
## [1] "excess"
```

```
#or use formula
Kur = (sum((IBM$Adj.Close - mean(IBM$Adj.Close))^4) /
    length(IBM$Adj.Close)) / sd(IBM$Adj.Close)^4
Kur
```

```
## [1] 3.93378
```

The sample mean of IBM dataset is 138.18, standard deviation is 10.40, skewness is -0.32, excess kurtosis is 0.934 and kurtosis is 3.934.

(2)

```
IBMFit = stdFit(IBM$Adj.Close)
IBMFit$par
```

```
##       mean         sd         nu
## 138.311938  13.542733   2.631856
```

t-distribution (IBMFit) has been fitted above. Mean estimate is 138.31, standard deviation estimate 13.54, degrees-of-freedom (nu) estimate 2.63.

(3)

Bootstrap the sample mean using **model free** first:

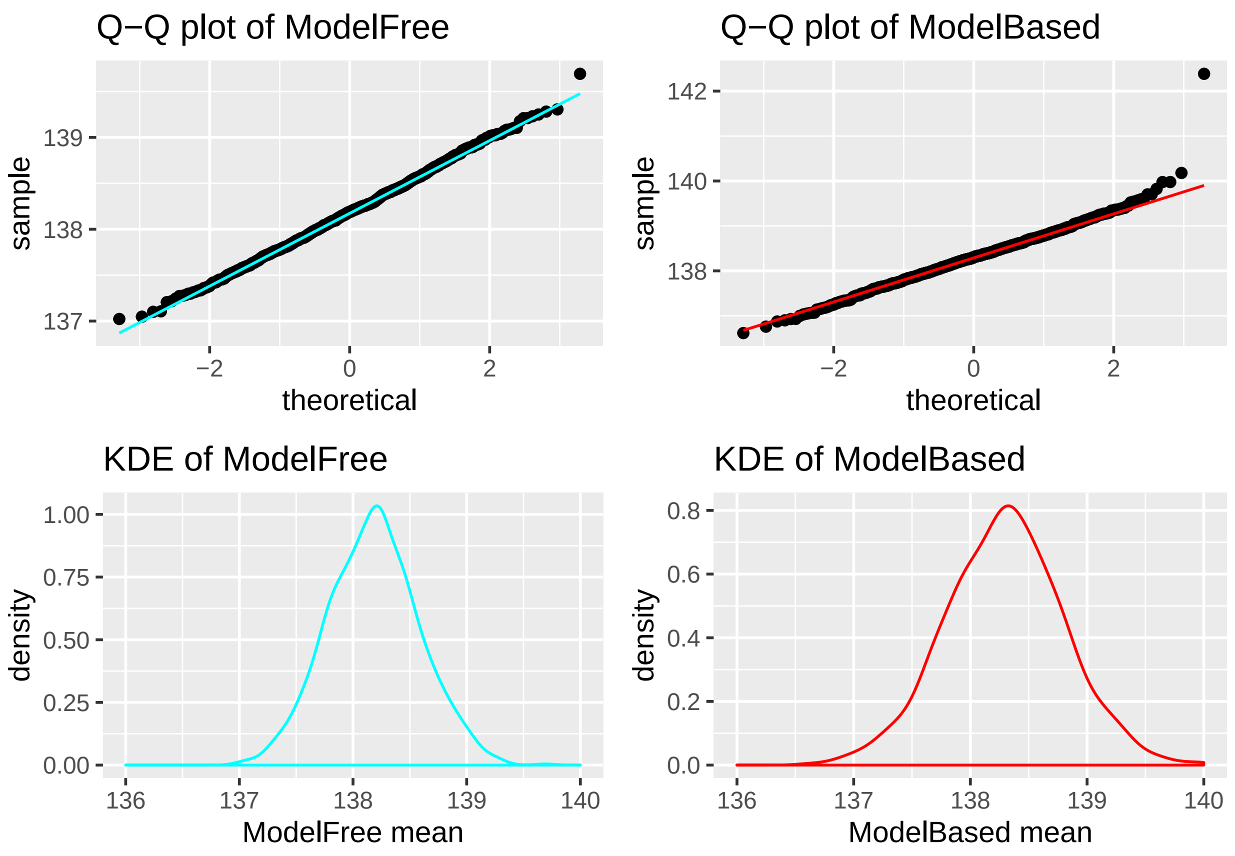
```
set.seed(723)
n = length(IBM$Adj.Close)
nboot = 1000
ModelFree = rep(0, nboot)
for (i in (1:nboot))
{
  ind0 = sample(1:n, replace = T)
  ModelFree[i] = mean(IBM$Adj.Close[ind0])
}
ModelFree_mean <- mean(ModelFree)
ModelFree_sd <- sd(ModelFree)
```

Then do the bootstrap **model based** on *t*-distribution.

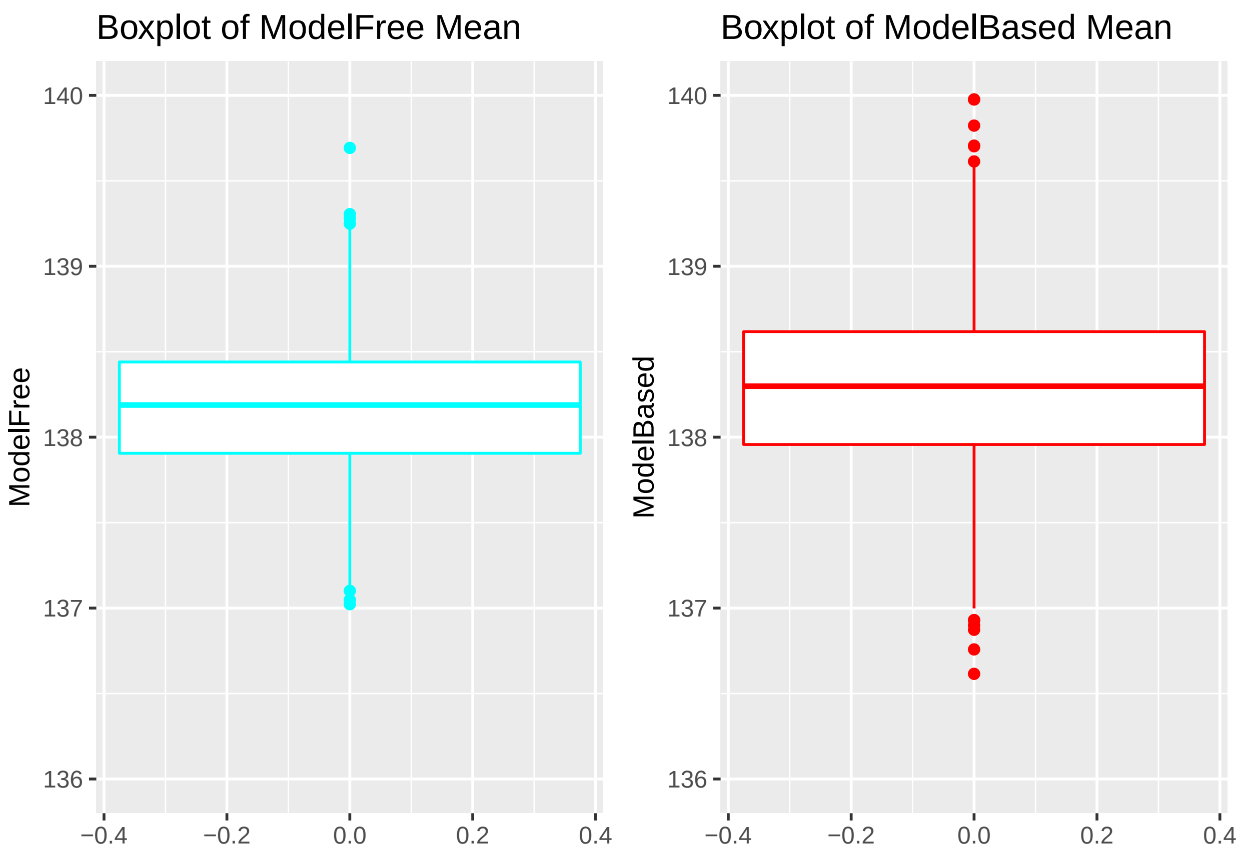
```
ModelBased = rep(0, nboot)
for (b in (1:nboot))
{
  ModelBased[b] = mean(rstd(n, mean = IBMFit$par["mean"],
                            sd = IBMFit$par["sd"], nu = IBMFit$par["nu"]))
}
ModelBased_mean <- mean(ModelBased)
ModelBased_sd <- sd(ModelBased)
```

(4)

```
ModelFree = data.frame(ModelFree)
ModelBased = data.frame(ModelBased)
require(gridExtra)
p16 <- ggplot(ModelFree, aes(sample = ModelFree)) + geom_qq() +
  geom_qq_line(col = "cyan") + labs(title = "Q-Q plot of ModelFree")
p17<- ggplot(ModelFree, aes(x = ModelFree)) + geom_density(col = "cyan") +
  labs(x = "ModelFree mean", title = "KDE of ModelFree") + xlim(c(136, 140))

p18 <- ggplot(ModelBased, aes(sample = ModelBased)) + geom_qq() +
  geom_qq_line(col = 2) + labs(title = "Q-Q plot of ModelBased")
p19 <- ggplot(ModelBased, aes(x = ModelBased)) + geom_density(col = 2) +
  labs(x = "ModelBased mean", title = "KDE of ModelBased") + xlim(c(136, 140))
grid.arrange(p16, p18, p17, p19, ncol = 2)
```



```r
p20 <- ggplot(ModelFree, aes(y = ModelFree)) + geom_boxplot(col = "cyan") +
  labs(title = "Boxplot of ModelFree Mean") + ylim(c(136, 140))
p21 <- ggplot(ModelBased, aes(y = ModelBased)) + geom_boxplot(col = 2) +
  labs(title = "Boxplot of ModelBased Mean") +  ylim(c(136, 140))
grid.arrange(p20, p21, ncol = 2)
```



Based on the Q-Q plot, KDE plot, and the box plot.

Comparing the Q-Q plot, it is safe to say that both bootstrap follows a normal distribution because both are having a Q-Q plot that follows closely to the guidline.

Comparing the box plot side-by-side, the first and third quantile range of the ModelFree is obviously smaller than ModelBased. The median of ModelBased bootstrap is larger than ModelFree.

(5)

Calculate the CI of both bootstrap.
Find the 95% CI of **model-free** bootstrap first:

```r
quantile(ModelFree$ModelFree, 0.025)
```

```
##     2.5%
## 137.4189
```

```r
quantile(ModelFree$ModelFree, 0.975)
```

```
##    97.5%
## 138.9931
```

Hence, the 95% CI of **model-free** bootstrap with digits = 5 is $(137.42, 138.99)$.
Calculate the 95% CI of **model-based** bootstrap:

```r
quantile(ModelBased$ModelBased, 0.025)
```

```
##     2.5%
## 137.2842
```

```r
quantile(ModelBased$ModelBased, 0.975)
```

```
##    97.5%
## 139.3078
```

Hence, the 85% CI of **model-based** bootstrap with degits = 5 is $(137.28, 139.31)$.

(6)

Compute the bias using formula

$$BIAS_{boot}(\hat{\theta}) = \bar{\hat{\theta}} - \hat{\theta}$$

.

```r
#sample mean
sample_mean <- mean(IBM$Adj.Close)
#Bias of Model Free bootstrap
Bias_ModelFree <- ModelFree_mean - sample_mean
Bias_ModelFree
```

```
## [1] -0.0007613152
```

```r
#Bias of Model Based bootstrap
Bias_ModelBased <- ModelBased_mean - sample_mean
Bias_ModelBased
```

```
## [1] 0.1166106
```

Based on the above calculation:
Bias of the sample mean of IBM based on **model-free** bootstrap is **-0.00076**.
Bias of sample mean of IBM based on **model-based** bootstrap is **0.11661**.

(7)

Calculate MSE of both bootstrap using the formula

$$MSE_{boot}(\theta) = BIAS^2_{boot}(\hat{\theta}) + s^2_{boot}(\hat{\theta})$$

```r
# Mean-Squared Error of Model-Free bootstrap
MSE_ModelFree <- Bias_ModelFree ^ 2 + ModelFree_sd ^ 2
MSE_ModelFree
```

```
## [1] 0.1589373
```

```r
# Mean-Squared Error of Model-Based bootstrap
MSE_ModelBased <- Bias_ModelBased ^ 2 + ModelBased_sd ^ 2
MSE_ModelBased
```

```
## [1] 0.2912905
```

Based on the calculation above:
Mean-Squared Error of **Model-Free** bootstrap is **0.1589**.
Mean-Squared Error of **Model-Based** bootstrap is **0.2913**.