STA457 Assignment 2

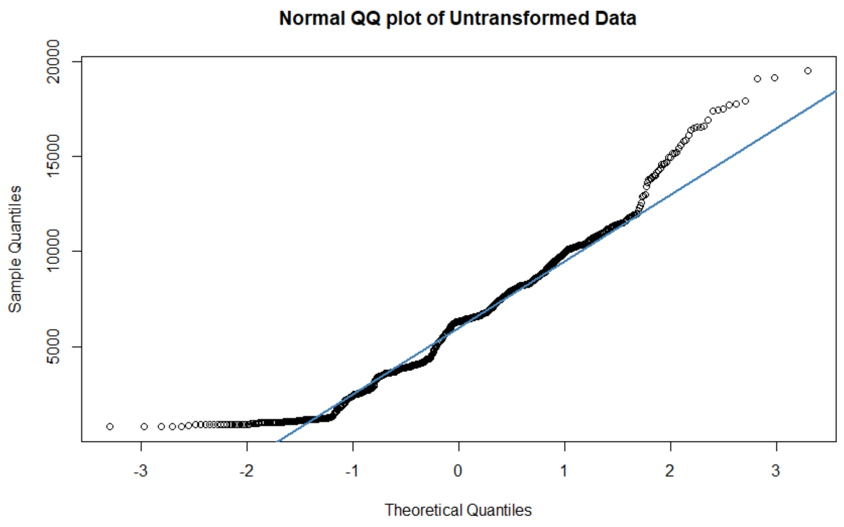
Brian Lee 1002750855

1 Problem 1

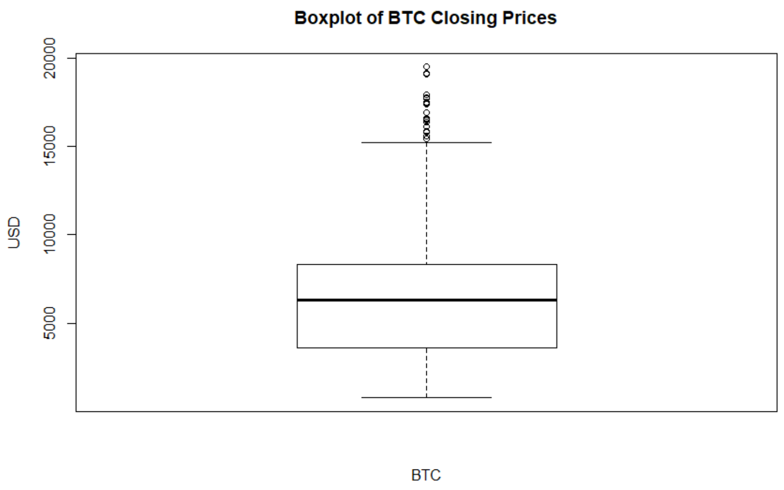
Note that the only data from 2017-2019 have been extracted and used. (Refer to appendix for code)

1.1

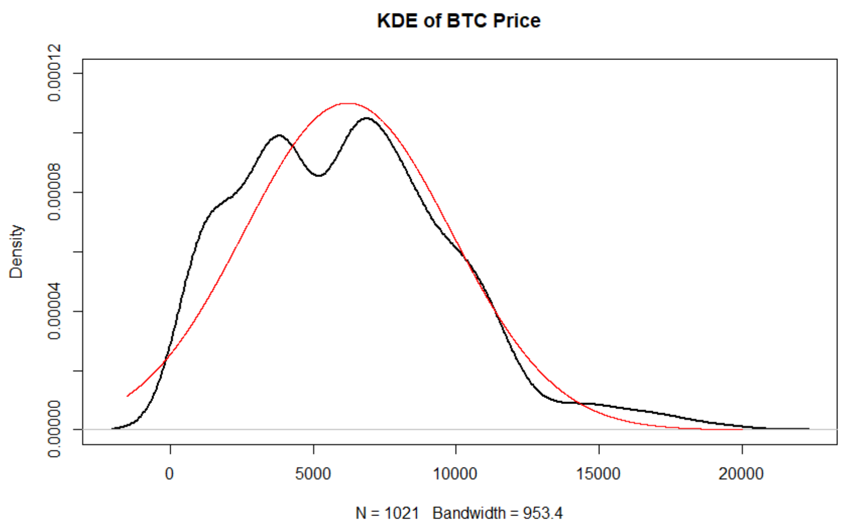
We first discuss the untransformed data.



We can observe deviation from linearity in the normal QQ plot, seeing significant deviations in the more extreme values, and demonstrating a general convex pattern which indicates left skew. This indicates that the data is not normal.



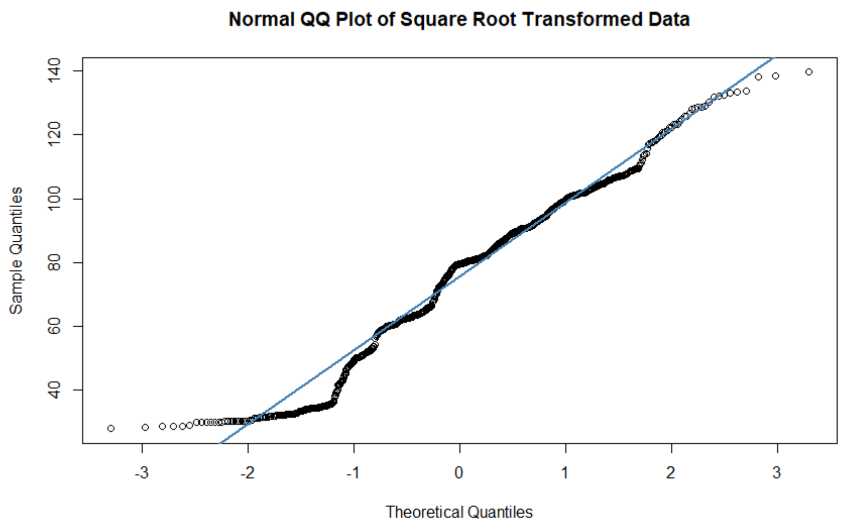
The boxplot further indicates the left-skewness of the data, as we see more mass concentrated on the bottom part beneath the line indicating the median. Furthermore, we can also see that large amount of extreme values/outliers in the boxplot, especially for higher extreme outliers.



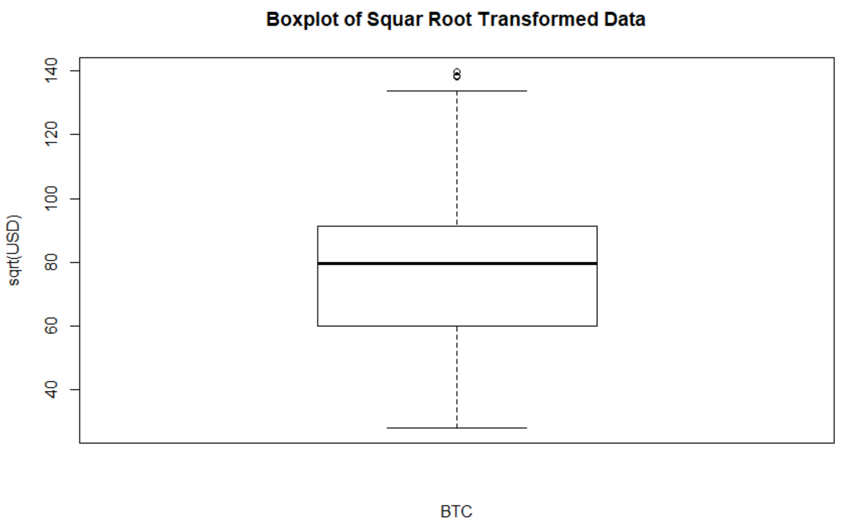
The black line above is the kernel density estimate, while the red line is a normal distribution with parameters given by the sameple mean and standard deviation. The kernel density estimate also confirms the left-skew of the data, and displays some bimodality with the two peaks (this is despite adjusting bandwith to smooth the curves). By comparing with the normal distribution, we see that the left skew has given it a very heavy left shoulder, while the left tail is light compared to the normal distribution, while the right tail is slightly heavier than the normal distribution, implying that the right tail is heavier than the left tail.

1.2

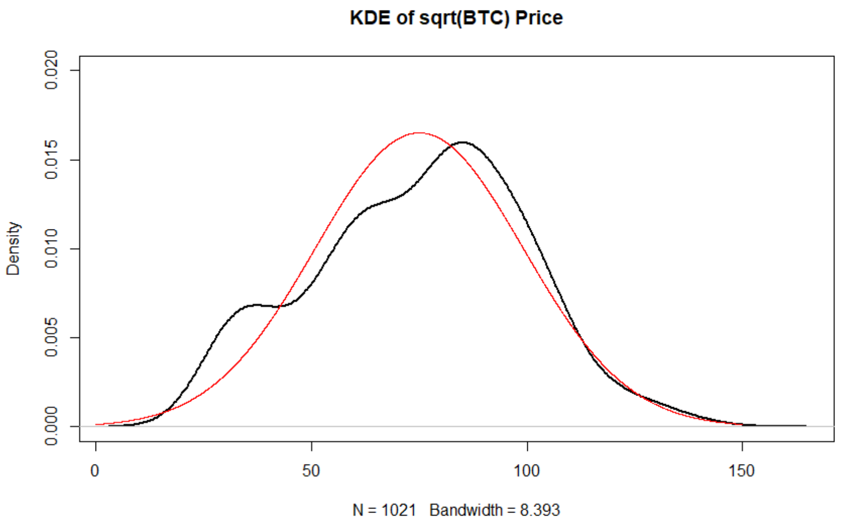
We now discuss the square-root transformed data.



QQ plot is closer to linearity with the square-root transformed data; in particular, the extreme values to the right (large value outliers) have been removed and the right 3/4th's of the data seems to follows a normal distribution fairly closely. However, the little 'valley' on the left side indicates that the left tail may be heavier than a normal distribution.

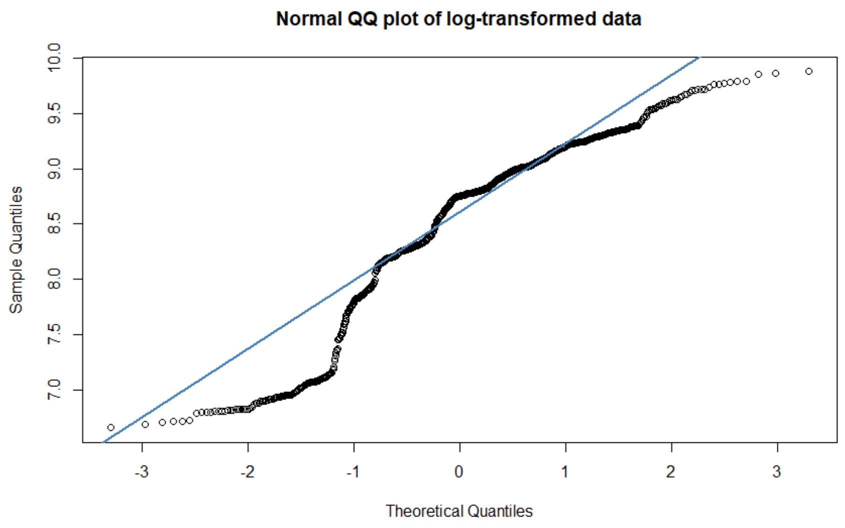


The boxplot appears more symmetric than in the untransformed case, with many outliers removed.

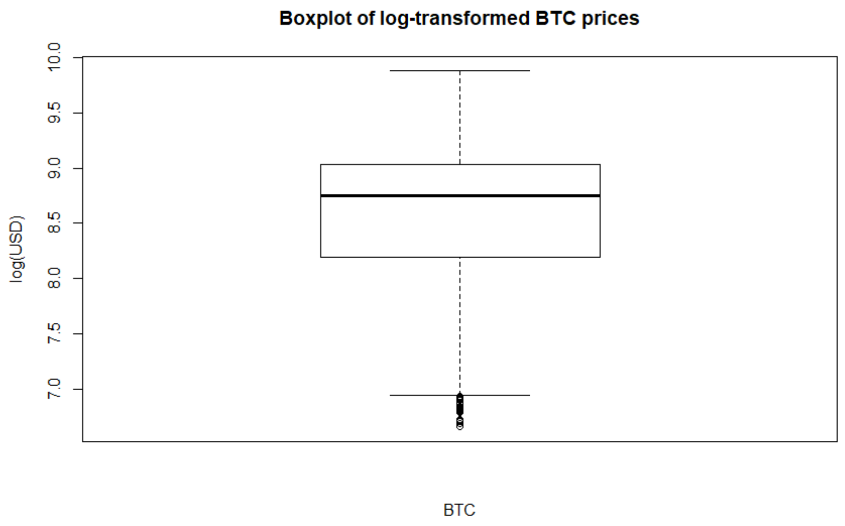


The black line is KDE while the red line is a fitted normal distribution. We see that the square root transformed data fits a normal distribution better than the untransformed data, although the concentration of mass at the lower values (heavy left tail) still deviates from normality significantly.

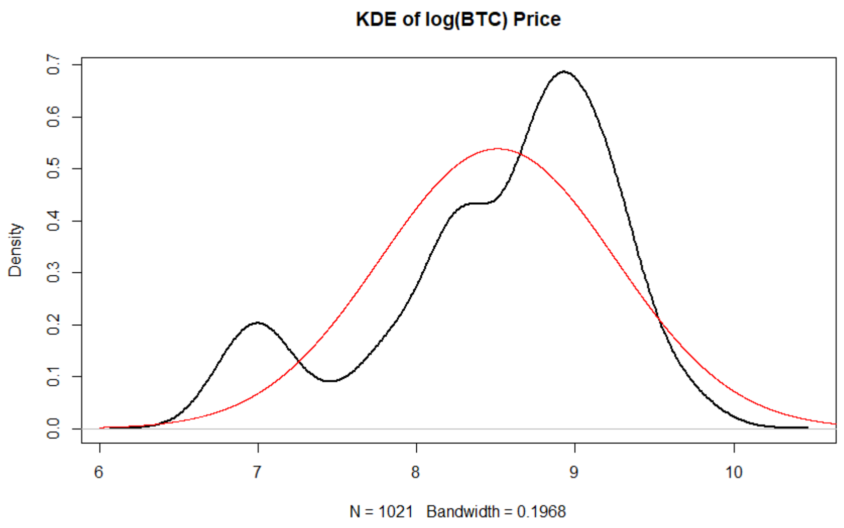
Now for the log-transformed data:



We see that the normal QQ plot of the log-transformed data deviates significantly from linearity indicating non-normality, with what appears to be a convex-concave pattern. This may be a sign of heavy tails, but also may be due to bimodality in the data, as we consider later in the KDE. We can see that there are a large number of outliers, especially in the left side (lower extreme values).



The boxplot again indicates asymmetry of the data, and shows a large number of outliers around the lower values.



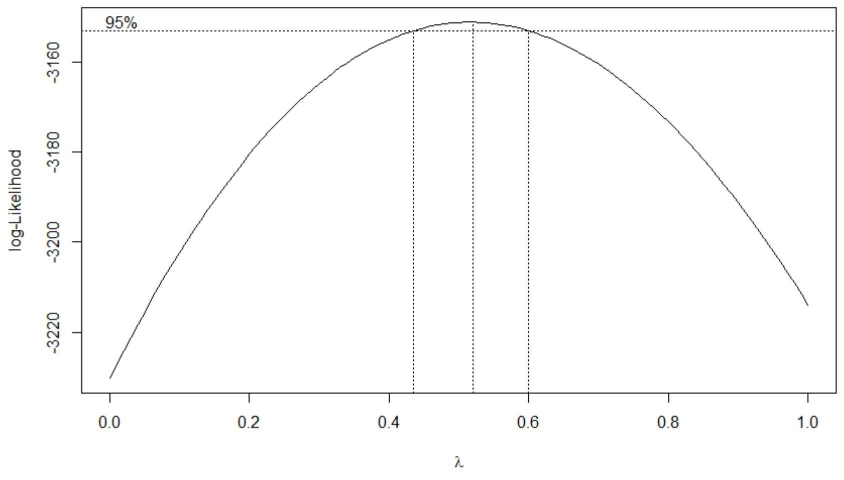
Again, the black line is the KDE of the log data and the red line is a fitted normal distribution. We can see that the data appears bimodal, with two peaks: a small one to the left, which explains the 'heavy left tail' indications that we've seen above, and a large one to the right. This also gives another explanation to the normal QQ plot: instead of heavy tails of a unimodal distribution, the bimodal behaviour explains the convex-concave pattern.

1.3

As suggested by the hint, we use the boxcox function and compute our estimate by optimizing.

```
bcPrice <- boxcox(prices~1, lambda=seq(0,1,1/100))
obj <- bcPrice$y
maxObj <- max(obj)
estimate <- match(maxObj, obj)/100
```

Using a partition of 100, maximum likelihood estimation gives us $\widehat{\lambda}_{MLE} \simeq 0.53$, which approximately matches the graphical representation given by the plot (MLE is given by the λ that maximizes below):



As a side note, using finer partitions with size going to ∞, we seem to converge to around $\widehat{\lambda}_{MLE} \simeq 0.5166$

1.4

A 99% confidence interval for lambda can be calculated by:

```
range(bcPrice$x[bcPrice$y > max(bcPrice$y)-qchisq(0.99,1)/2])
```

and gives us a confidence interval (when using partition size 100) of $[0.41, 0.62]$.

1.5

A skewed t-distribution can be fitted to the BTC price data by:

```
#Fit skew-t-distrib
skewFit <- sstdFit(prices)
```

1.6

Using our fitted skewed t-distribution above, we get the estimates of the parameters (by MLE):

```
> skewFit$estimate
mean          sd            nu            xi
6.000215e+03  3.905802e+03  1.022144e+05  9.397415e+00
```

2 Problem 2

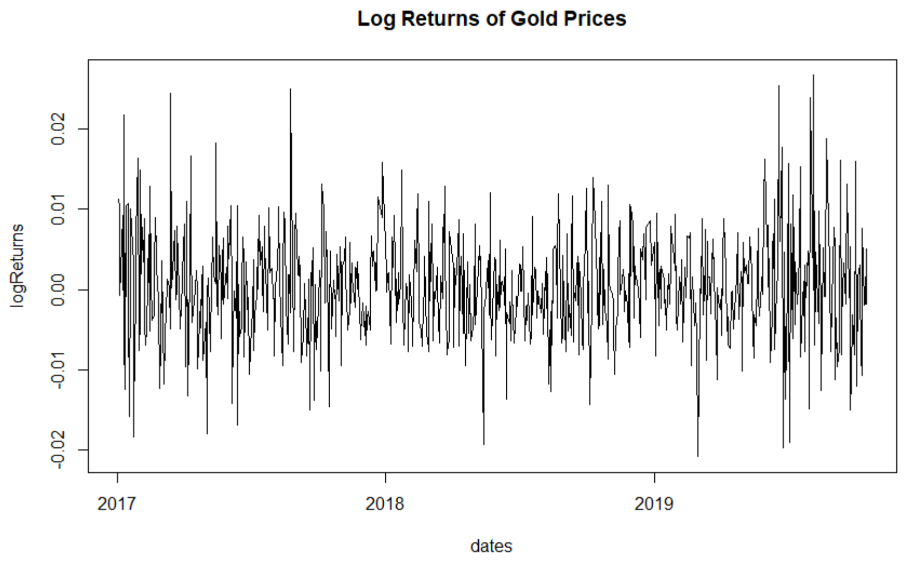
Note that the only data from 2017-2019 have been extracted and used. (Refer to appendix for code)

2.1

I used the code below to read the file, clean the data and compute the log return

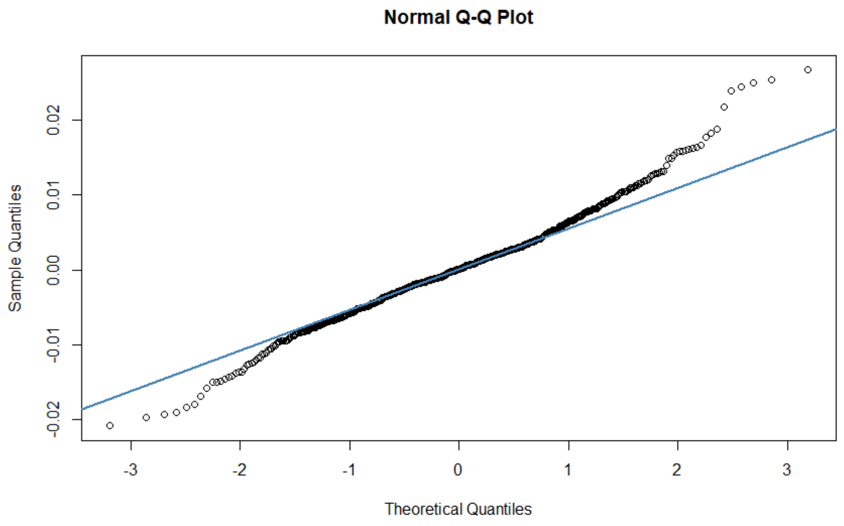
```
fileRead <- read.csv(file="LBMA-GOLD.csv")
goldData <- fileRead[-c(709:13093),]
price <- as.numeric(goldData$USD..PM.)
dates <- as.Date(goldData$Date)
dates <- dates[-c(1, 204,207,457,460)]
price <- price[-c(204,207,457,460)]

#Log returns
returns <- -diff(price)/price[-length(price)]
logReturns <- log(1+returns)
plot(dates, logReturns)
```

This gives us the following time series plot of log returns:

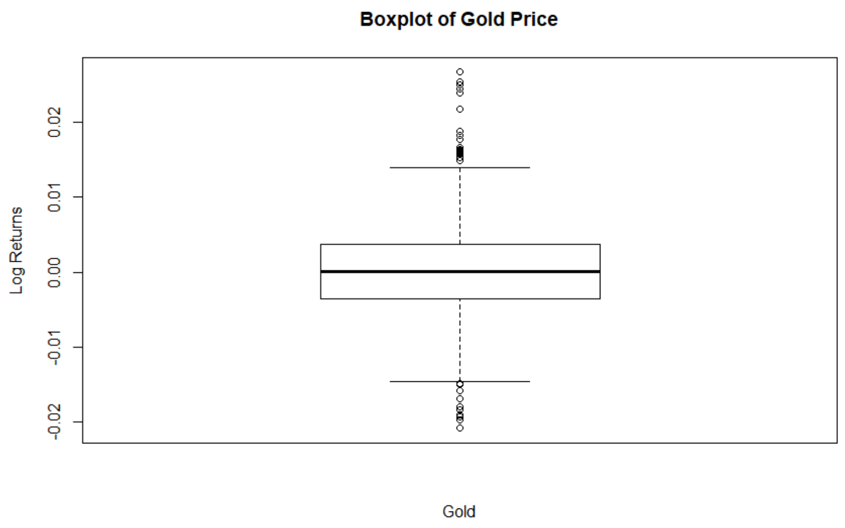


The time series, sans outliers, appears to have relatively constant variance, but the numerous large spikes in 2017 and end of 2019 seem to create a horizontal hourglass shape, indicating a degree of nonconstant variance, or some dependence on the conditional variance.

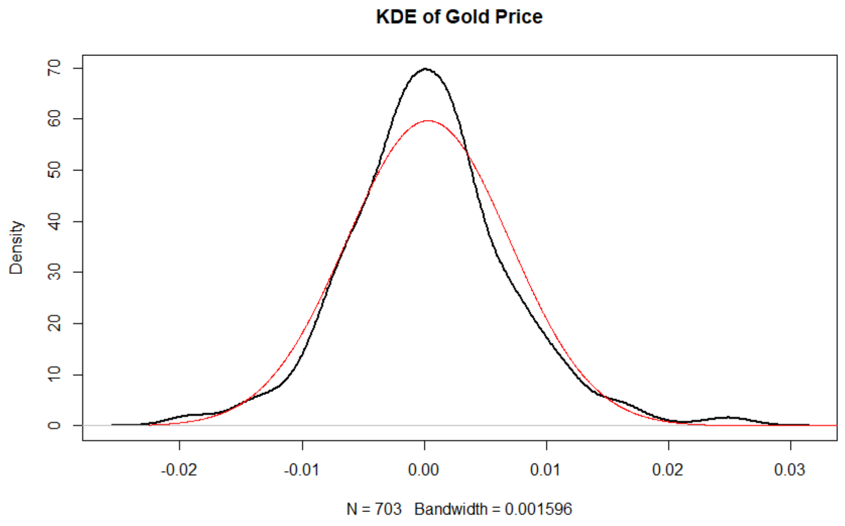
2.2



The normal QQ plot of the gold price data seems to follows linearity except at the tails, where a slight convex-concave pattern indicates heavier tails, and the data appears symmetric.



The boxplot confirms the observations made in the QQ plot, where we observe a large amount of outliers (heavier tails) and the boxplot appears symmetric.



The black line is the KDE and the red line is a fitted normal distribution. We see that indeed, the log returns data is symmetric, and appears to have slightly heavier tails than a normal distribution, and more mass in the center (although that is a feature of the KDE, thus may not be a feature of the data). This indicates that perhaps a t-distribution may be an appropriate fit to the data, as the data is symmetric and heavier-tailed than a normal distribution.

2.3

We fit a t-distribution to the data using the following code

```
Y=logReturns
loglike_std = function(x) {
        f = -sum(dstd(Y, x[1], x[2], x[3], log=TRUE))
        f}
start = c(mean(Y), sd(Y), 4)
```

```
fit_std = optim(start, loglike_std, method = "L-BFGS-B",
lower = c(-0.1, 0.001, 2.1),
upper = c(0.1, 1, 20), hessian = TRUE)
cat("MLE = ", round(fit_std$par, digits = 5))
```

Which gives us the estimates of the parameters:

```
> fit_std$par
[1] 0.0001241357 0.0073183376 4.0049559935
```

So the fit estimates $\mu = 0.000124$, $\sigma = 0.00732$, $\nu = 4.00496$

2.4

```
minus_logL_std = fit_std$value
AIC_std = 2*minus_logL_std+2*length(fit_std$par)
BIC_std = 2*minus_logL_std+log(length(logReturns))*length(fit_std$par)
```

Thus fitting this model, we get

$$AIC_{std} = -5069.832 \text{ and } BIC_{std} = -5056.166$$

2.5

The following code fits the skewed t-distribution to the data.

```
Y=logReturns
loglike_sstd = function(x) {
f = -sum(dsstd(Y, x[1], x[2], x[3], x[4], log=TRUE))
f}
start = c(mean(Y), sd(Y), 4, 1.5)
fit_sstd = optim(start, loglike_sstd, method = "L-BFGS-B",
lower = c(-0.1, 0.001, 2.1, -0.99),
upper = c(0.1, 1, 20, 0.99), hessian = TRUE)
cat("MLE = ", round(fit_sstd$par, digits = 5))
```

This model estimates the parameters to be (where ξ is the skew parameter)

$$\mu = 0.00009822 \quad \sigma = 0.00732 \quad \nu = 4.005 \quad \xi = 0.99$$

2.6

We calculate the AIC and BIC for the skew t-distribution fit model

```
minus_logL_sstd = fit_sstd$value
AIC_sstd = 2*minus_logL_sstd+2*length(fit_sstd$par)
BIC_sstd = 2*minus_logL_sstd+log(length(logReturns))*length(fit_sstd$par)
```

This gives us

$$AIC_{skew} = -5067.216 \text{ and } BIC_{skew} = -5048.994$$

So we have

$$AIC_{std} < AIC_{skew} \text{ and } BIC_{std} < BIC_{skew}$$

Thus both the AIC and the BIC select the standard t-distribution.

3 Problem 3

Note that the only data from 2017-2019 have been extracted and used. (Refer to appendix for code)

3.1

Computing these values with R gives us:

$$\text{sample mean} = 138.183 \quad \text{sample standard deviation} = 10.398$$

$$\text{kurtosis} = 3.945 \quad \text{skew} = -0.3211$$

3.2

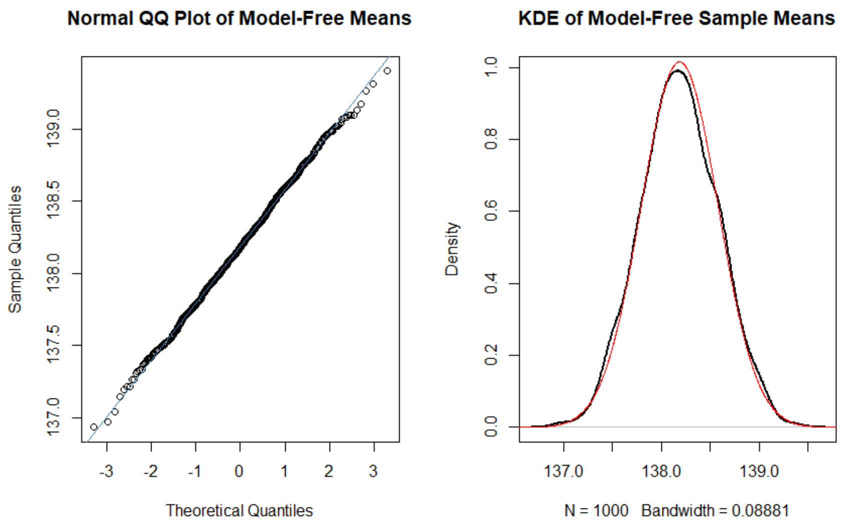
The R code and result are as follows:

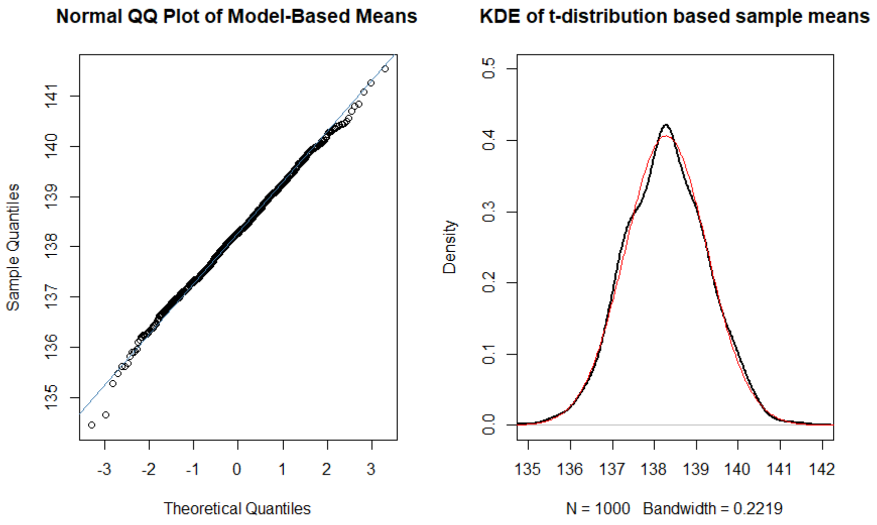
```
> tFit = stdFit(price)
> tFit$par
mean        sd         nu
138.311938  13.542733  2.631856
```

3.3

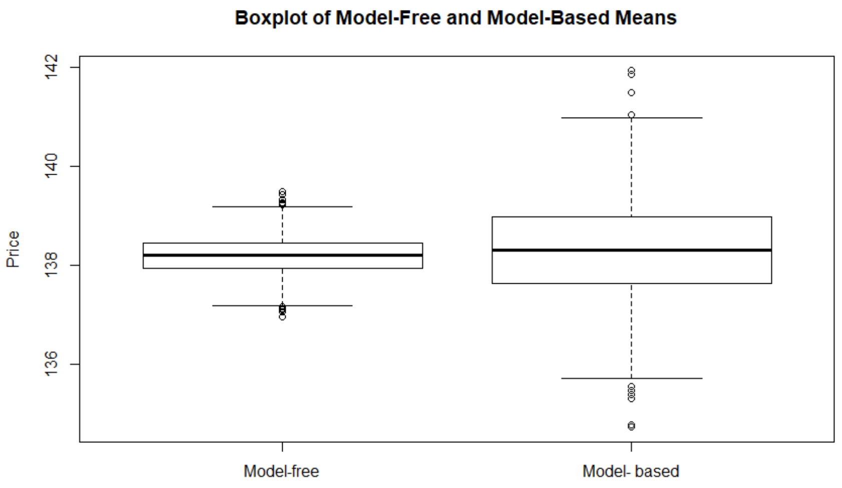
```
##Model-free
avs <- c()
for (ct in 1:1000){
rsample <- sample(price, 704, replace=T)
avs <- c(avs, mean(rsample))
}
mfMean <- mean(avs)

##t-distribution model-based
tavgs <- c()
for (ct1 in 1:1000){
tsample <- rt(704, df=2.631856)*13.542733+138.311938
tavgs <- c(tavgs, mean(tsample))
}
mbMean <- mean(tavgs)
```

3.4





We see that there are no marked differences in the QQ plot and the shape of the KDE of the model free and model-based sample means, as they both seeem to closely follow a normal distribution. However, we note that the KDE of the model-free mean is a much sharper, taller, 'narrow' peak than the model-based one, while the model-based curve is more spread out, indicating larger variance (look at scale).



The boxplots support the above observation, as the model-based means have much larger spread/variance and more extreme values compared to the model-free means.

3.5

The above plots indicate that the means come from a normal distribution, hence we have confidence intervals given by

$$\widehat{\mu} \pm s_{boot}(\widehat{\mu}) z_{\alpha/2}$$

Using

```
#Model-free
cimf_low <- mfMean - sd(avs)*qnorm(0.95, mean=0, sd=1)
cimf_up <- mfMean + sd(avs)*qnorm(0.95, mean=0, sd=1)
```

```
#Model-based
cimb_low <- mbMean - sd(tavgs)*qnorm(0.95, mean=0, sd=1)
cimb_up <- mbMean + sd(tavgs)*qnorm(0.95, mean=0, sd=1)
```

We get that the confidence interval is:

$$\text{Model-free confidence interval: } [137.54, 138.83] \quad \text{or} \quad 138.1856 \pm 0.6456$$

$$\text{Model-based confidence interval: } [136.63, 139.96] \quad \text{or} \quad 138.2964 \pm 1.6659$$

3.6

We compute the bias

```
mfBias <- mfMean - priceMean
mbBias <- mbMean - priceMean
```

Which gives us the result:

$$\text{BIAS}_{\text{model-free}} = 0.00249 \quad \text{BIAS}_{\text{model-based}} = 0.1132$$

3.7

We estimate the MSE by:

```
mfMSE <- sd(avs)^2+mfBias^2
mbMSE <- sd(tavgs)^2+mbBias^2
```

Which gives us the result

$$\text{MSE}_{\text{model-free}} = 0.154 \quad \text{MSE}_{\text{model-based}} = 1.0386$$

4 Appendix

4.1 Problem 1 entire code

```
library("MASS")
library("fGarch")
fileRead <- read.csv(file = "BTC.csv")
btcData <- fileRead[-c(1022:2365),]

prices <- as.numeric(gsub(",","", btcData$Close.USD))
priceMean <- mean(prices)
priceSd <- sd(prices)

sqrt.Price  <- sqrt(prices)
sqrt.PriceMean <- mean(sqrt.Price)
sqrt.PriceSd <- sd(sqrt.Price)

log.Price <- log(prices)
log.PriceMean <- mean(log.Price)
log.PriceSd <- sd(log.Price)


#Untransoformed Data
#QQ plot
qqnorm(prices, main="Normal QQ plot of Untransformed Data")
qqline(prices, col = "steelblue", lwd=2)

#Boxplot
boxplot(prices, main = "Boxplot of BTC Closing Prices", ylab = "USD", xlab = "BTC")

#KDE
d <- density(prices, adjust=1.5)
plot(d, main="KDE of BTC Price", ylim=c(0,0.00012), lwd = 2)
xval = seq(-1500, 20000, length=10000)
yval = dnorm(xval, mean=priceMean, sd=priceSd)
lines(xval, yval, col="red", lwd = 0.5)

#Sqrt-transformed data
#QQ plot
qqnorm(sqrt.Price, main="Normal QQ Plot of Square Root Transformed Data")
qqline(sqrt.Price, col = "steelblue", lwd=2)

#Boxplot
boxplot(sqrt.Price, main = "Boxplot of Square Root Transformed Data", ylab = "sqrt(USD)",

#KDE
d1 <- density(sqrt.Price, adjust=1.6)
plot(d1, main="KDE of sqrt(BTC) Price", ylim=c(0,0.02), lwd=2)
xval1 = seq(0, 150, length=10000)
yval1 = dnorm(xval1, mean=sqrt.PriceMean, sd=sqrt.PriceSd)
lines(xval1, yval1, col="red", lwd=0.5)


#Log-transformed data
```

```
#QQ plot
qqnorm(log.Price, main="Normal QQ plot of log-transformed data")
qqline(log.Price, col = "steelblue", lwd=2)

#Boxplot
boxplot(log.Price, main = "Boxplot of log-transformed BTC prices", ylab = "log(USD)", xlab

#KDE
d2 <- density(log.Price, adjust=1.4)
plot(d2, main="KDE of log(BTC) Price", lwd=2)
xval2 = seq(6, 11, length=10000)
yval2 = dnorm(xval2, mean=log.PriceMean, sd=log.PriceSd)
lines(xval2, yval2, col="red", lwd=0.5)


#Boxcox
bcPrice <- boxcox(prices~1, lambda=seq(0,1,1/100))
obj <- bcPrice$y
maxObj <- max(obj)
estimate <- match(maxObj, obj)/100

#CI
range(bcPrice$x[bcPrice$y > max(bcPrice$y)-qchisq(0.99,1)/2])


#Fit skew-t-distrib
skewFit <- sstdFit(prices)
```

4.2 Problem 2 entire code

```
library("fGarch")

fileRead <- read.csv(file="LBMA-GOLD.csv")
goldData <- fileRead[-c(709:13093),]
price <- as.numeric(goldData$USD..PM.)
dates <- as.Date(goldData$Date)
dates <- dates[-c(1, 204,207,457,460)]
price <- price[-c(204,207,457,460)]

#Log returns
returns = -diff(price)/price[-length(price)]
logReturns = log(1+returns)
plot(dates, logReturns, main="Log Returns of Gold Pric")

lreturnMean = mean(logReturns)
lreturnSd = sd(logReturns)

#QQ plot
qqnorm(logReturns)
qqline(logReturns, col = "steelblue", lwd=2)

#Boxplot
boxplot(logReturns, main="Boxplot of Gold Price", ylab="Log Returns", xlab="Gold")
```

```
#KDE
d <- density(logReturns, adjust=1.2)
plot(d, main="KDE of Gold Price", lwd=2)
xval = seq(-0.0225, 0.29, length=10000)
yval = dnorm(xval, mean=lreturnMean, sd=lreturnSd)
lines(xval, yval, col="red", lwd=0.5)

#Fitting T-Distribution for Real
Y=logReturns
loglike_std = function(x) {
f = -sum(dstd(Y, x[1], x[2], x[3], log=TRUE))
f}
start = c(mean(Y), sd(Y), 4)
fit_std = optim(start, loglike_std, method = "L-BFGS-B",
lower = c(-0.1, 0.001, 2.1),
upper = c(0.1, 1, 20), hessian = TRUE)
cat("MLE =", round(fit_std$par, digits = 5))
minus_logL_std = fit_std$value
AIC_std = 2*minus_logL_std+2*length(fit_std$par)
BIC_std = 2*minus_logL_std+log(length(logReturns))*length(fit_std$par)


#Fitting Skew-T distribution for Real
Y=logReturns
loglike_sstd = function(x) {
f = -sum(dsstd(Y, x[1], x[2], x[3], x[4], log=TRUE))
f}
start = c(mean(Y), sd(Y), 4, 1.5)
fit_sstd = optim(start, loglike_sstd, method = "L-BFGS-B",
lower = c(-0.1, 0.001, 2.1, -0.99),
upper = c(0.1, 1, 20, 0.99), hessian = TRUE)
cat("MLE =", round(fit_sstd$par, digits = 5))
minus_logL_sstd = fit_sstd$value
AIC_sstd = 2*minus_logL_sstd+2*length(fit_sstd$par)
BIC_sstd = 2*minus_logL_sstd+log(length(logReturns))*length(fit_sstd$par)
```

4.3 Problem 3 entire code

```
library(fGarch)

fileRead <- read.csv("IBM.csv")
IBMdata <- fileRead[-c(705:1258),]
price <- IBMdata$Adj.Close

###1
priceMean <- mean(price)
ppriceSd <- sd(price)
skew <- skewness(price)
kurt <- kurtosis(price)



###2
tFit = stdFit(price)
tFit$par
```

```
###3
##Model-free
avs <- c()
for (ct in 1:1000){
rsample <- sample(price, 704, replace=T)
avs <- c(avs, mean(rsample))
}
mfMean <- mean(avs)

##t-distribution model-based
tavgs <- c()
for (ct1 in 1:1000){
tsample <- rt(704, df=2.631856)*13.542733+138.311938
tavgs <- c(tavgs, mean(tsample))
}
mbMean <- mean(tavgs)


###4

#model-free
qqnorm(avs, main="Normal QQ Plot of Model-Free Means")
qqline(avs, col = "steelblue", lwd =1)
#model-free
d <- density(avs, adjust=1)
plot(d, main="KDE of Model-Free Sample Means", lwd=2)
xval = seq(135, 150, length=10000)
yval = dnorm(xval, mean=mean(avs), sd=sd(avs))
lines(xval, yval, col="red", lwd=0.5)

#model-based
qqnorm(tavgs, main="Normal QQ Plot of Model-Based Means")
qqline(tavgs, col="steelblue", lwd=1)
#model-based
d1 <- density(tavgs, adjust=1)
plot(d1, main="KDE of t-distribution based sample means", ylim=c(0, 0.5), xlim=c(135,142),
xval1 = seq(-135, 150, length=10000)
yval1 = dnorm(xval1, mean=mean(tavgs), sd=sd(tavgs))
lines(xval1, yval1, col="red", lwd=0.5)

#Boxplots
boxplot(avs, tavgs, main="Boxplot of Model-Free and Model-Based Means", ylab="Price", name


###5
#Model-free
cimf_low <- mfMean - sd(avs)*qnorm(0.95, mean=0, sd=1)
cimf_up <- mfMean + sd(avs)*qnorm(0.95, mean=0, sd=1)

#Model-based
cimb_low <- mbMean - sd(tavgs)*qnorm(0.95, mean=0, sd=1)
cimb_up <- mbMean + sd(tavgs)*qnorm(0.95, mean=0, sd=1)
```

```
###6
mfBias <- mfMean - priceMean
mbBias <- mbMean - priceMean

###7
mfMSE <- sd(avs)^2+mfBias^2
mbMSE <- sd(tavgs)^2+mbBias^2
```