A#2

jonathanwai

03/11/2019

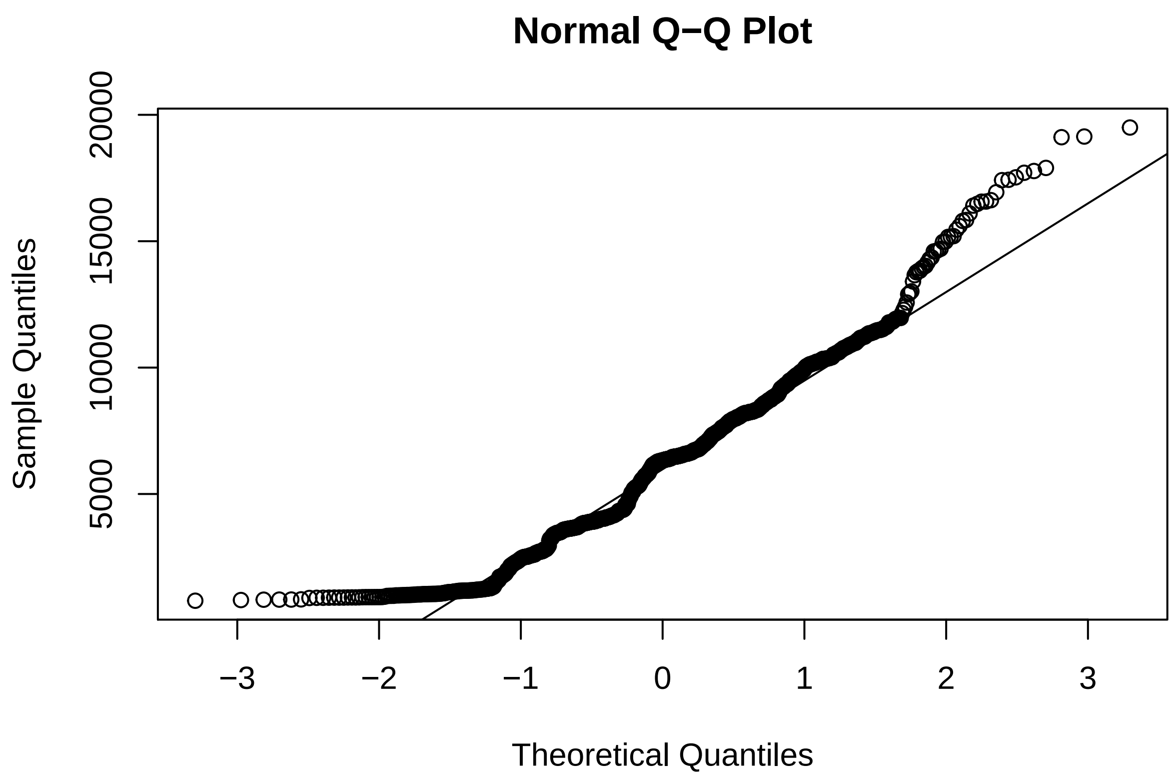
```
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

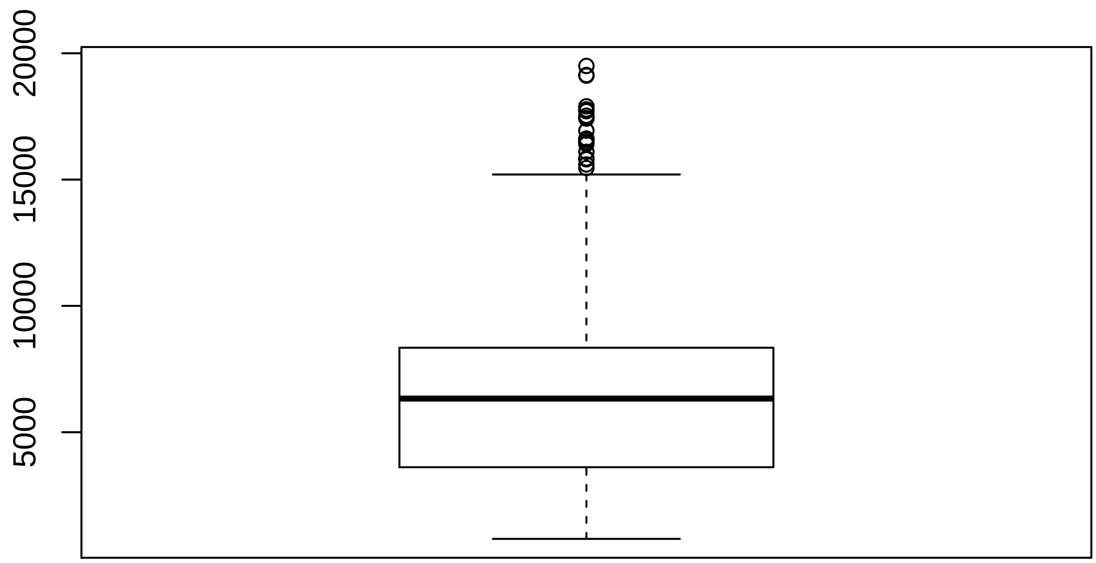
This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

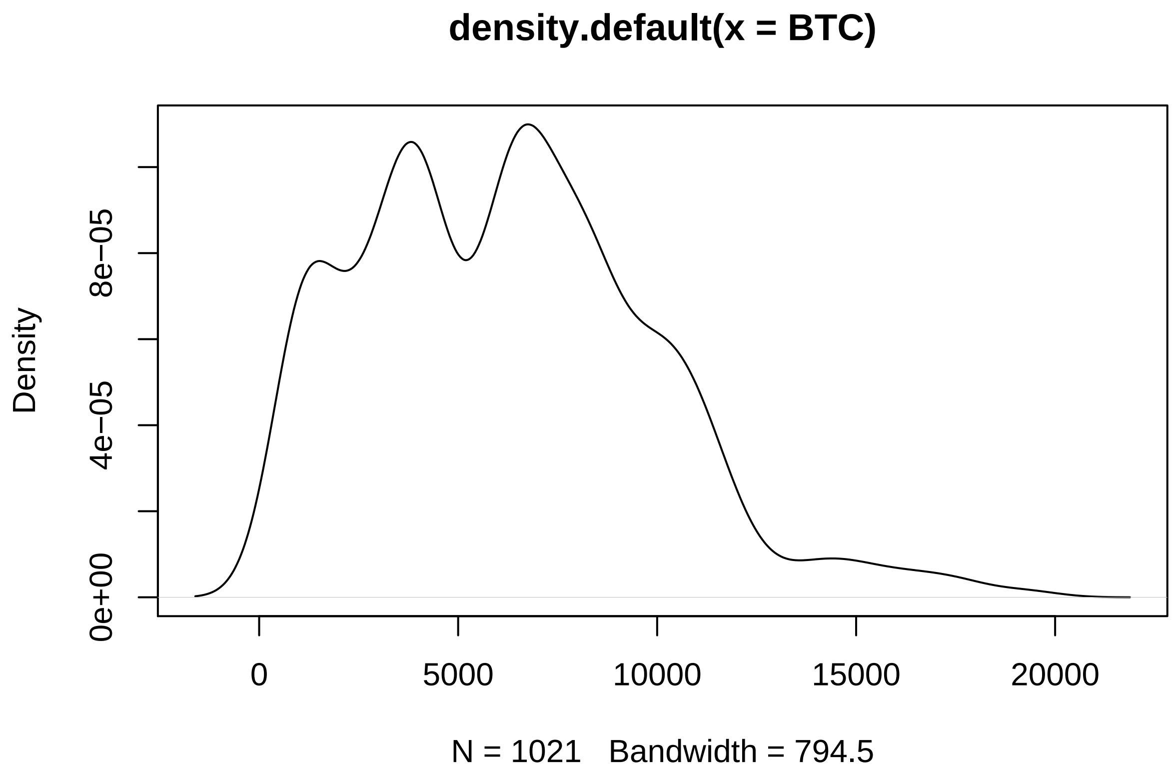
```
#1
library(readxl)
BTCNEW <- read_excel("C:\\Users\\Jonathan\\Desktop\\BTCNEW.xlsx")
BTC = BTCNEW$BTC
qqnorm(BTC)
qqline(BTC)
```



```r
boxplot(BTC)
```

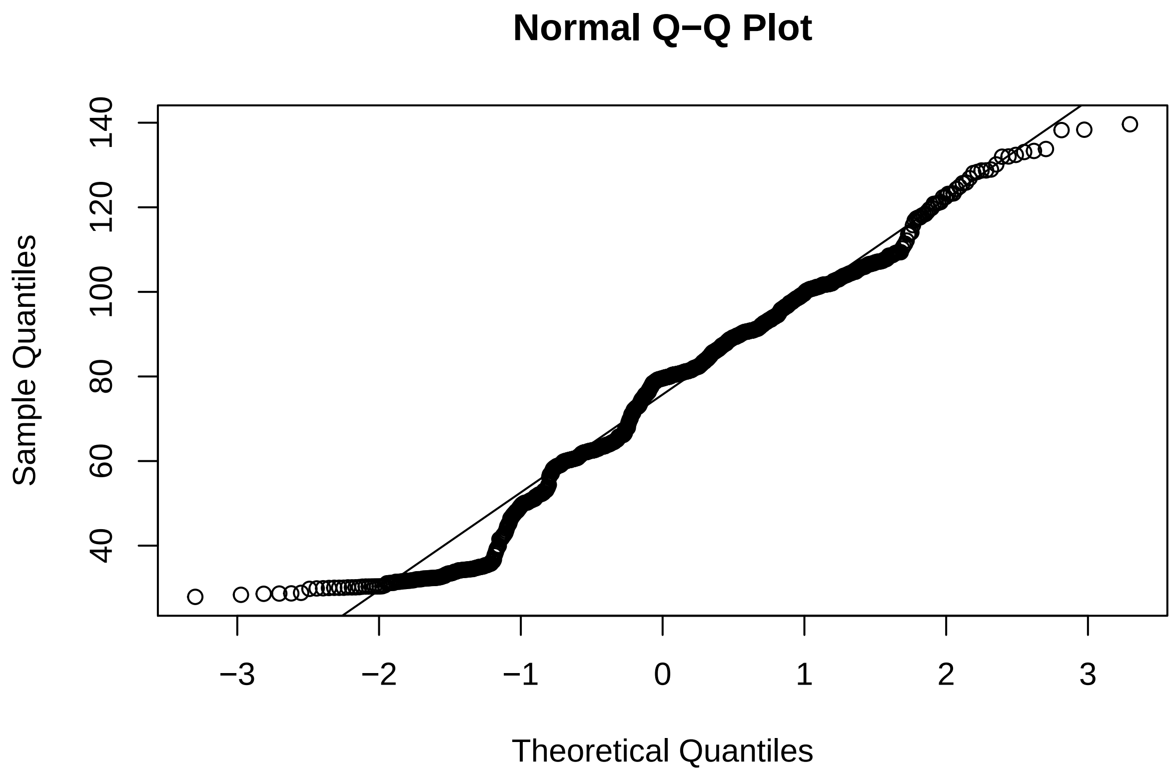



```r
plot(density(BTC))
```

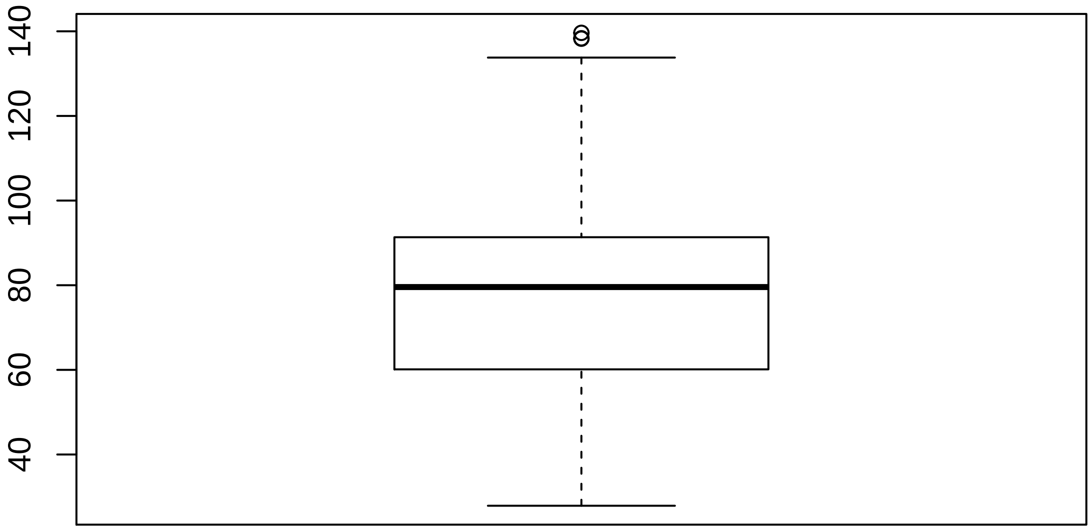



```
# Data appears to be non-normal and not normally distributed.
# Light tailed to the left
# Not symmetric
# Skewed to the right
# Light tailed to the left and heavy tailed to the right
# Right tail is heavier
```

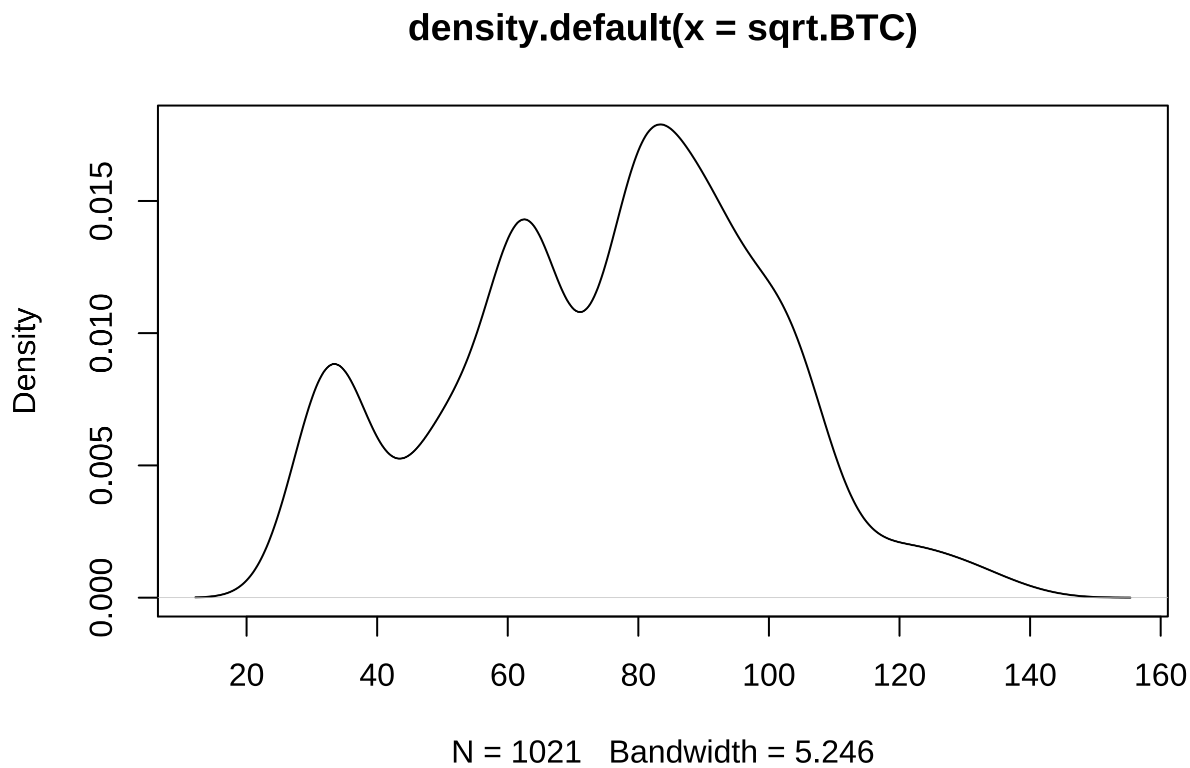
```
#2
sqrt.BTC=sqrt(BTC)
qqnorm(sqrt.BTC)
qqline(sqrt.BTC)
```




```
boxplot(sqrt.BTC)
```

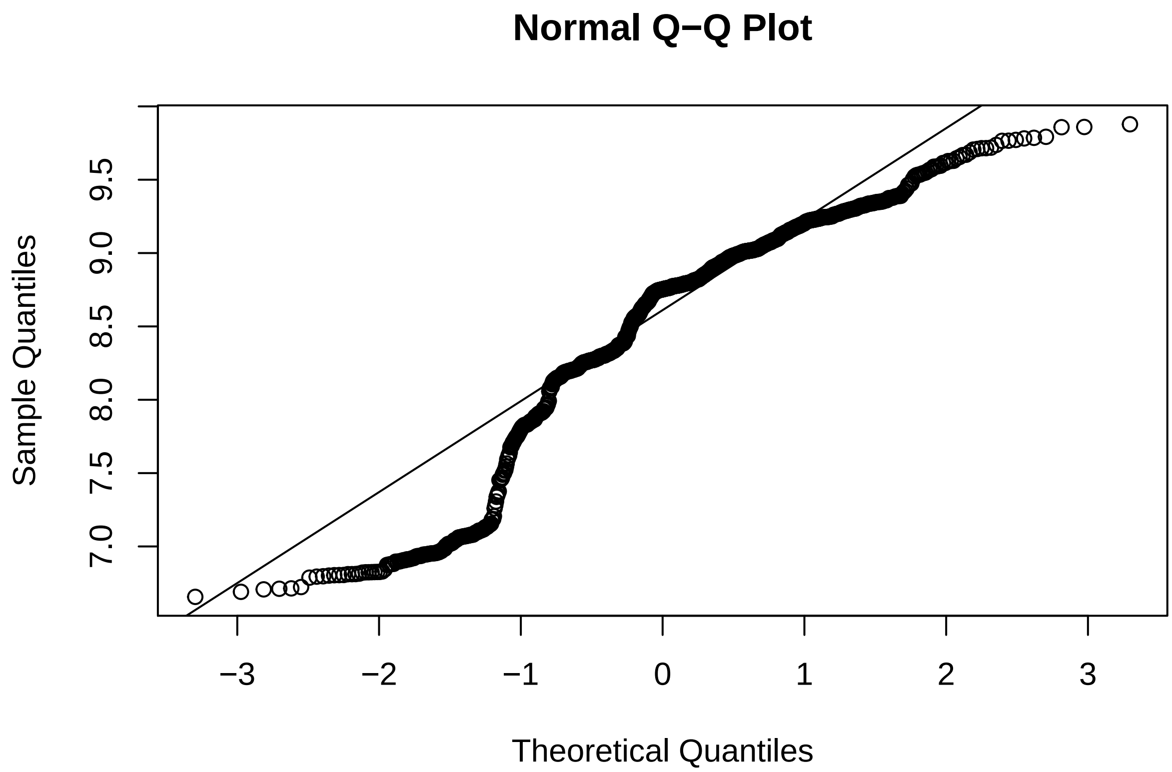



```r
plot(density(sqrt.BTC))
```

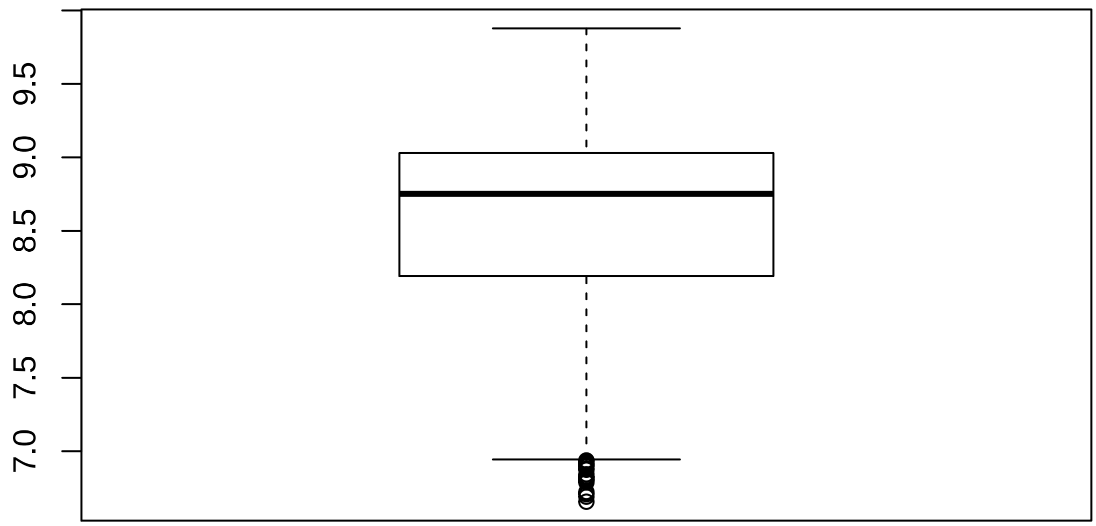



```
# Not normally distributed
# Slightly right skewed?
# Not symmetrically distributed, skewed to the right slightly
# Skewed to the right
# Light tailed compared to normal distribution
# Right tail is heavier
```

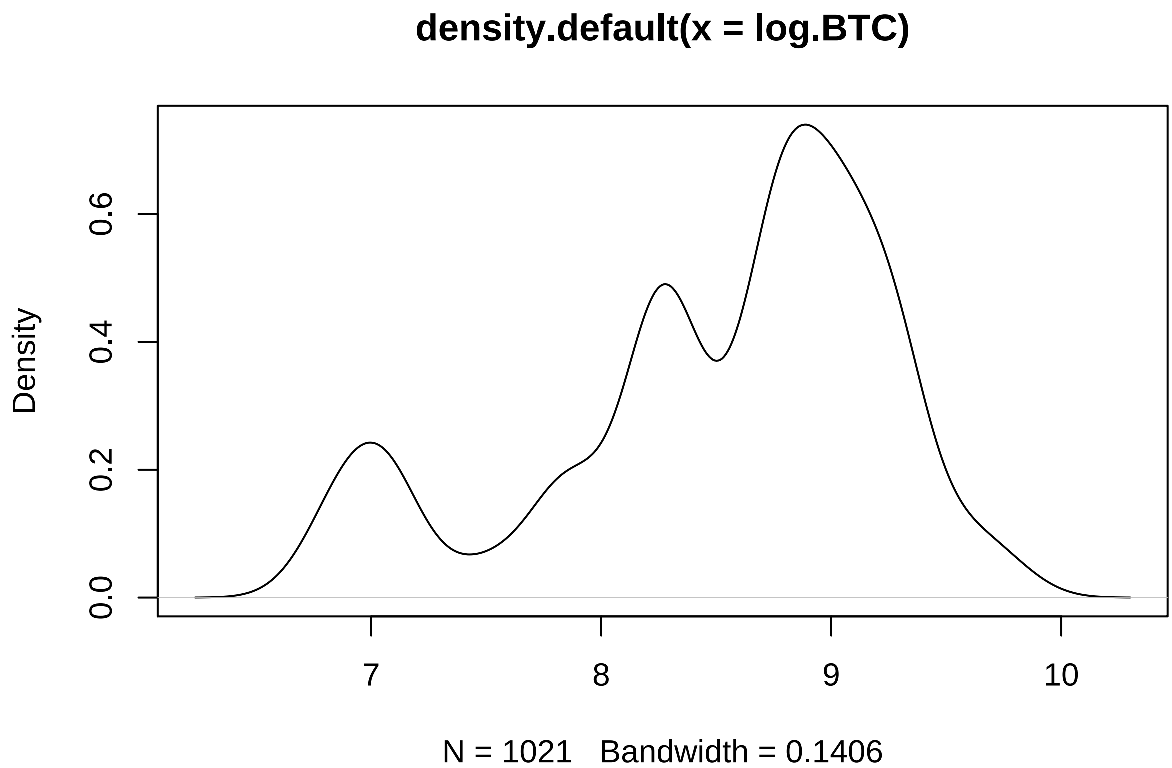
```
#2
log.BTC=log(BTC)
qqnorm(log.BTC)
qqline(log.BTC)
```




```r
boxplot(log.BTC)
```

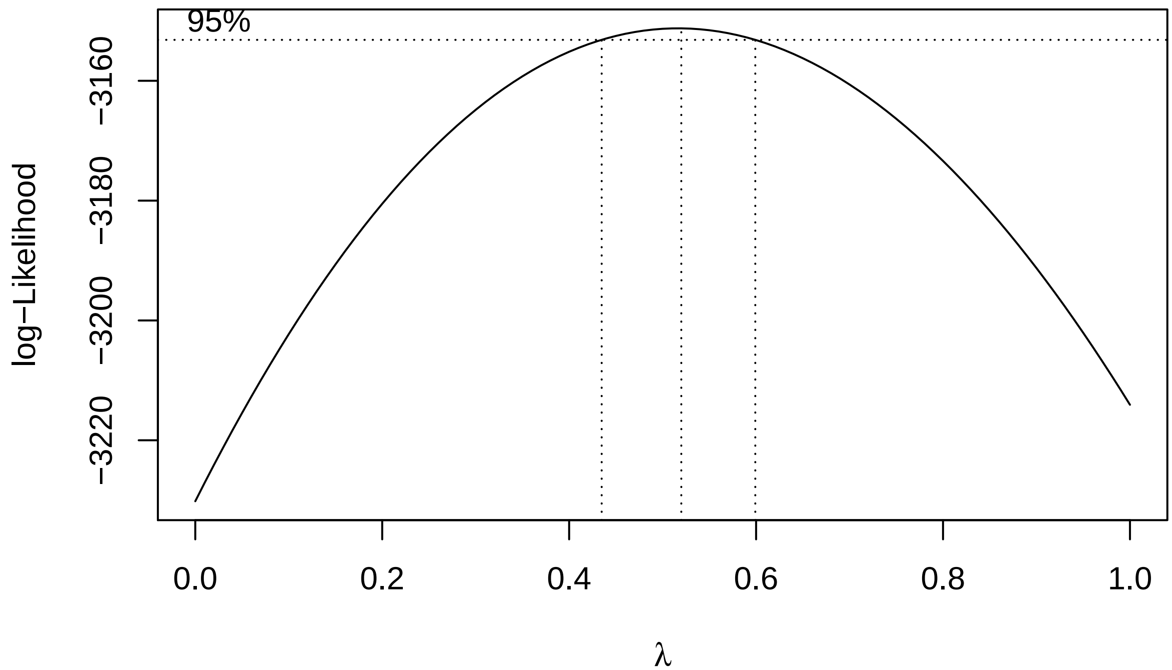



```r
plot(density(log.BTC))
```




```r
# Not normally distributed
# Left Skewed
# Not symmetrically distributed slightly skewed to the left
# Skewed to the left
# Right tail is lighter than left tail
# Left tail is heavier (More Flat)
```

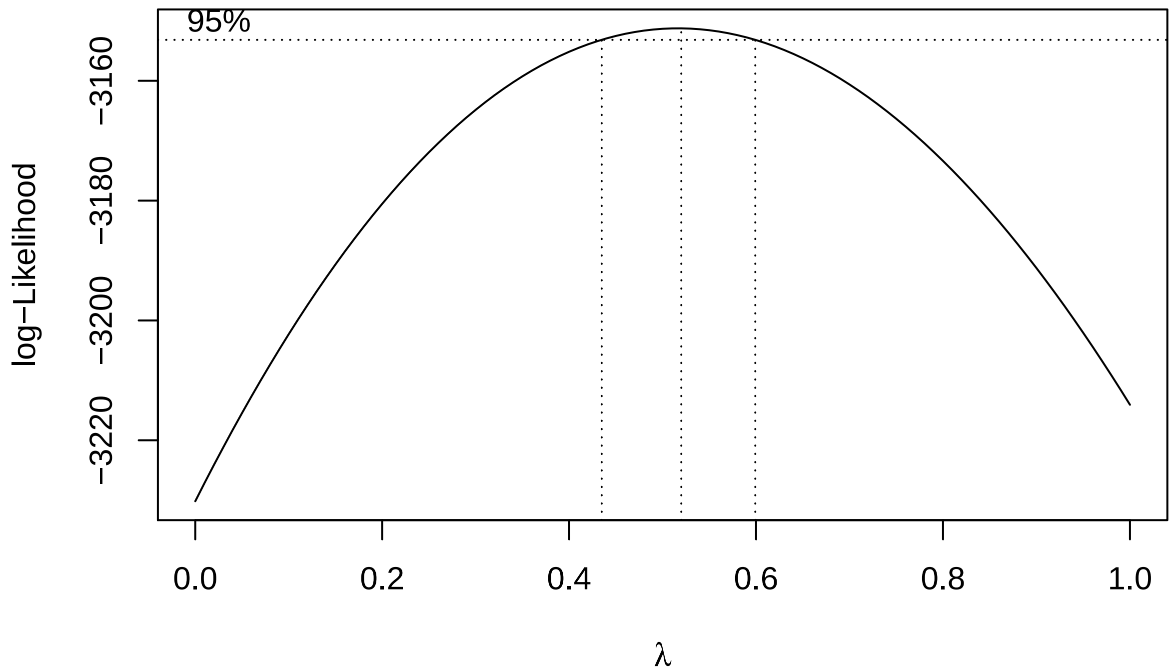
```r
#3
library(MASS)
Box = boxcox(BTC~1, lambda=seq(0,1,1/100), ploti=TRUE)
```




```
Results=data.frame(Box$x,Box$y)
Box$x[which.max(Box$y)]
```

```
## [1] 0.52
```

```
#4
library(MASS)
Box=boxcox(BTC~1, lambda=seq(0, 1, 1/100), plotit=TRUE)
```




```
Results=data.frame(Box$x, Box$y)
Results2 = Results[with(Results, order(-Results$Box.y)),]
Box$x[which.max(Box$y)]
```

```
## [1] 0.52
```

```
Box$x[Box$y > max(Box$y) - 1/2 * qchisq(.99,1)]
```

```
##  [1] 0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54
## [15] 0.55 0.56 0.57 0.58 0.59 0.60 0.61 0.62
```

```
#The confidence interval for lambda is (0.41,0.62)
```

```
#5
library(fGarch)
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

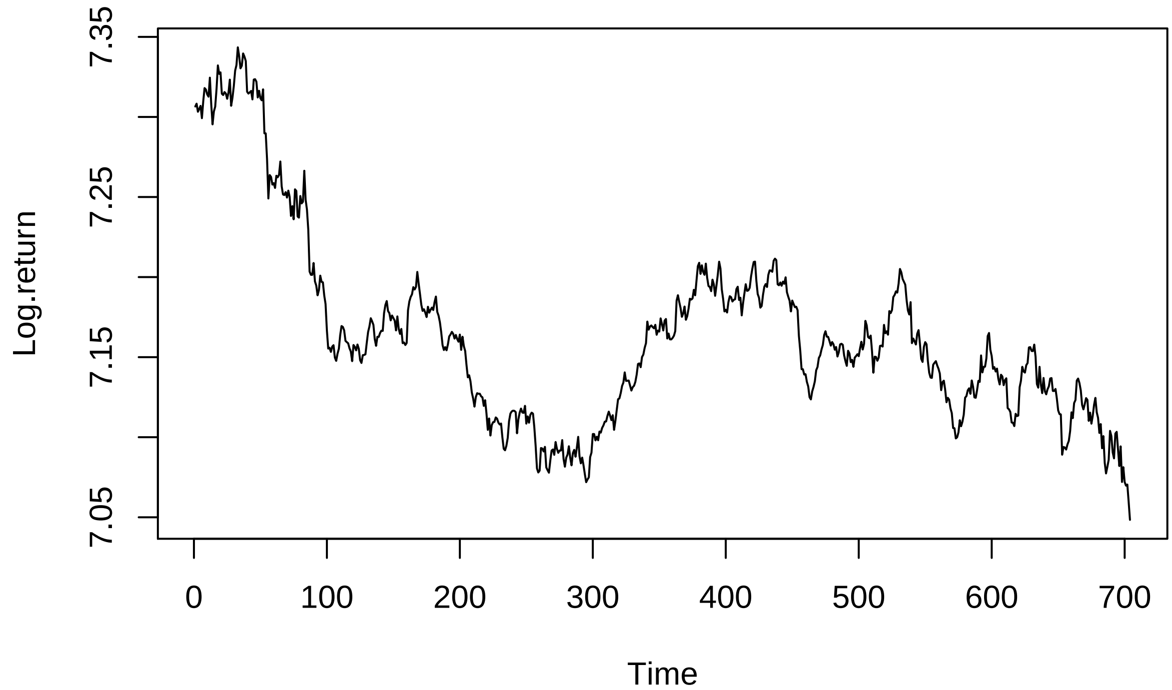
```
## Loading required package: fBasics
```

```r
sstdFit(BTC)
```

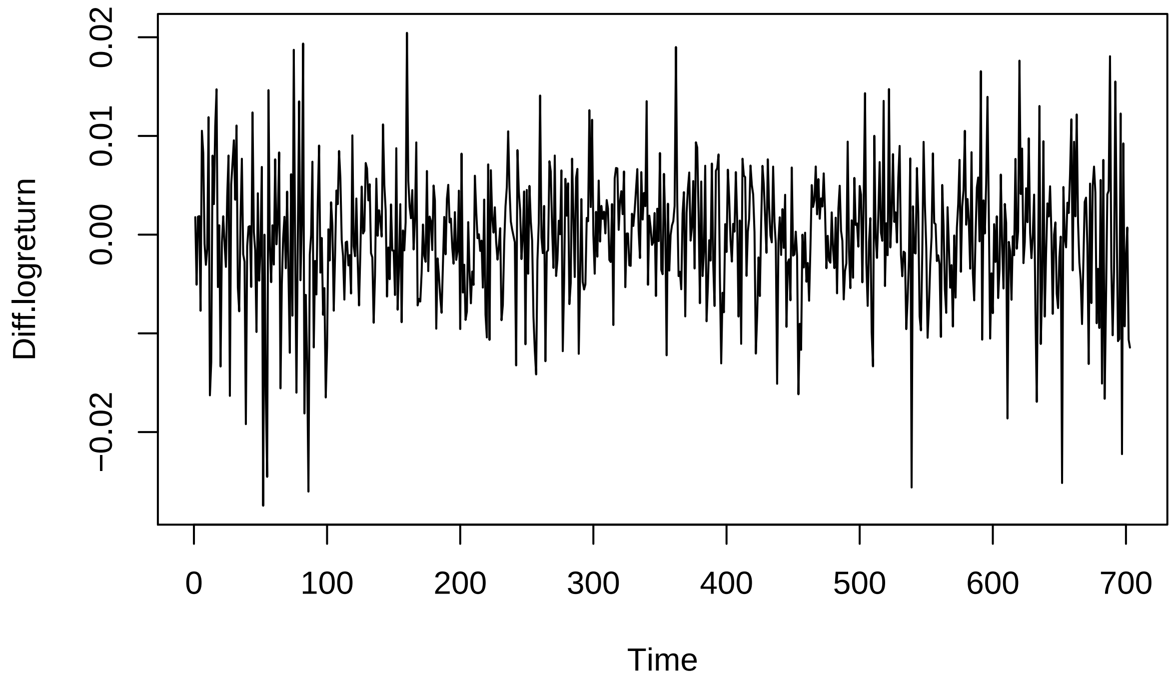
```
## $minimum
## [1] 9708.812
##
## $estimate
##         mean           sd           nu           xi
## 6.000215e+03 3.905802e+03 1.022144e+05 9.397415e+00
##
## $gradient
##          mean             sd             nu             xi
## -8.639512e-07  2.181214e-06 -1.370715e-08 -4.315482e-06
##
## $code
## [1] 1
##
## $iterations
## [1] 67
```

```r
#6
#The MLE estimate of the mean is 6000
#The MLE estimate of the standard deviation is 3905.7
#The MLE estimate of the shape parameter is 102200
#The MLE estimate of xi is 9.3972
```

```r
#Question 2
#1
library(readxl)
LBMA_GOLD <- read_excel("C:\\Users\\Jonathan\\Desktop\\LBMA-GOLD.xlsx")
Log.return = log(LBMA_GOLD$Price)
plot.ts(Log.return)
```

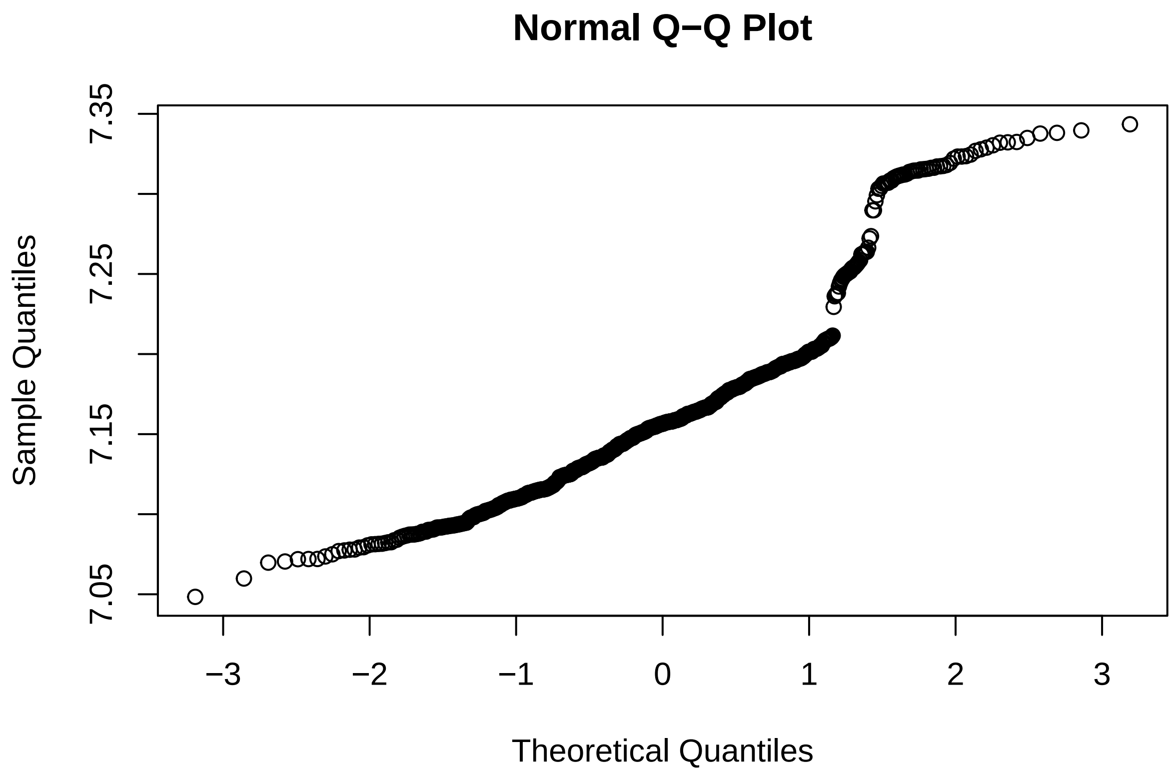



```r
Diff.logreturn = diff(Log.return)
plot.ts(Diff.logreturn)
```

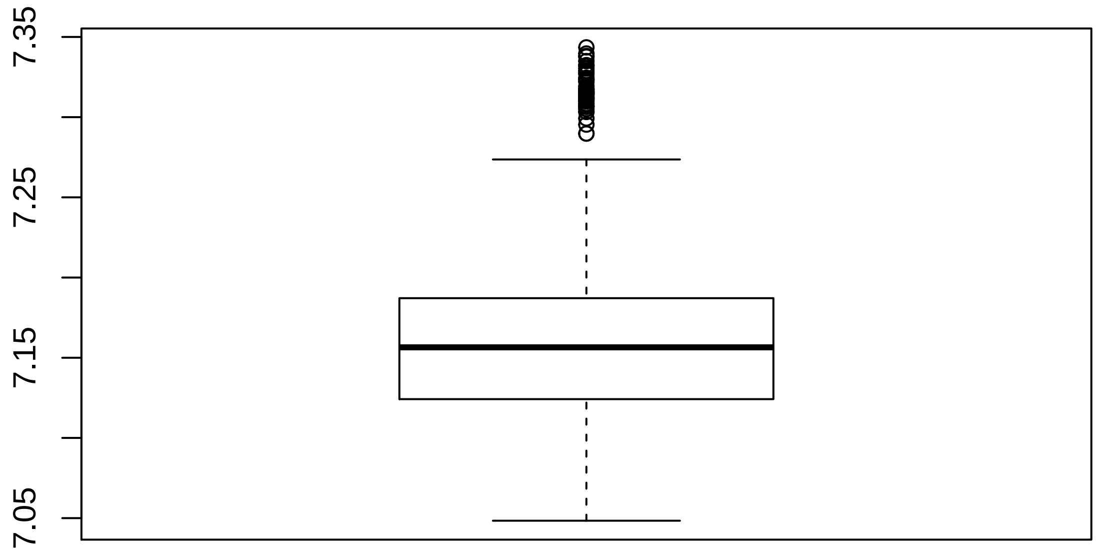



```
#The series of log returns appear to be nonstationary with
#more fluctuations at the beginning and the end of the
#observation period.
```

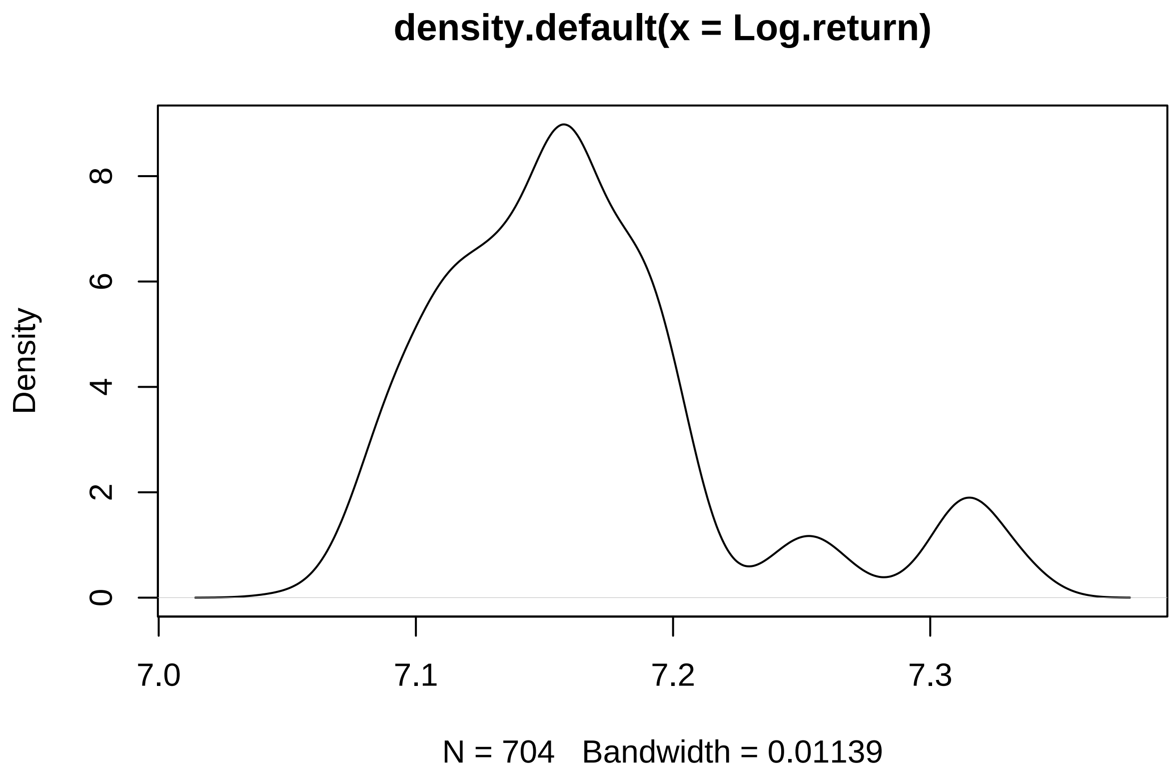
```
#2
qqnorm(Log.return)
```




```r
boxplot(Log.return)
```




```r
plot(density(Log.return))
```




```
#The qq plot shows a convex-concave pattern, indicating that
#the series are light tailed compared to a normal distribution.
#Additionally, the boxplot and the density plot show that
#the distribution of log returns is skewed to the right.
```

```
#3
library(fGarch)
objective_function = function(x){
  f = -sum(log(dstd(Log.return,x[1],x[2],x[3])))
}
st_vec=c(mean(Log.return), sd(Log.return),5)
fitstd=optim(st_vec,objective_function)
```

```
## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced

## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced

## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced

## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced

## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced

## Warning in log(dstd(Log.return, x[1], x[2], x[3])): NaNs produced
```

```r
fitstd$value
```

```
## [1] -1025.932
```

```r
fitstd$par
```

```
## [1] 7.15395087 0.06768246 3.05700318
```

```r
stdFit(Log.return)
```

```
## $par
##       mean         sd         nu
## 7.15395634 0.06767725 3.05765139
##
## $objective
## [1] -1025.932
##
## $convergence
## [1] 0
##
## $iterations
## [1] 19
##
## $evaluations
## function gradient
##       31       77
##
## $message
## [1] "relative convergence (4)"
```

```r
#The MLE estimate of the mean is 7.153951
#The MLE estimate of the standard deviation is 0.067682
#The MLE estimate of the degrees of freedom is 3.057003
```

```r
#4
n=length(Log.return)
AIC=2*3+2*fitstd$value
BIC=log(n)*3+2*fitstd$value
```

```r
#5
objective_function = function(x){
  f = -sum(log(dsstd(Log.return,x[1],x[2],x[3],x[4])))
}
st_vec=c(mean(Log.return), sd(Log.return),5,5)
fitsstd=optim(st_vec,objective_function)
```

```
## Warning in log(dsstd(Log.return, x[1], x[2], x[3], x[4])): NaNs produced

## Warning in log(dsstd(Log.return, x[1], x[2], x[3], x[4])): NaNs produced

## Warning in log(dsstd(Log.return, x[1], x[2], x[3], x[4])): NaNs produced

## Warning in log(dsstd(Log.return, x[1], x[2], x[3], x[4])): NaNs produced
```

```r
fitsstd$value
```

```
## [1] -1063.415
```

```r
fitsstd$par
```

```
## [1] 7.16335696 0.05970341 8.24472240 1.92239365
```

```r
#The MLE estimate of the mean is 7.163357
#The MLE estimate of the standard deviation is 0.059701
#The MLE estimate of the shape parameter is 8.242401
#The MLE estimate of xi is 1.922331
```

```r
#6
AIC=2*3+2*fitsstd$value
BIC=log(n)*3+2*fitsstd$value
#Since both AIC and BIC are smaller for the
#skewed t distribution, the skewed t distribution
#is a betterfit for the log return data.
```

```r
#1
library(readxl)
IBM <- read_excel("C:\\Users\\Jonathan\\Desktop\\IBM.xlsx")
Price=IBM$Price
Mean=mean(Price)
Sddev=sd(Price)
Skew=skewness(Price)
Kurt=kurtosis(Price)
```

```r
#2
stdFit(Price)
```

```
## $par
##       mean         sd         nu
## 138.311938  13.542733   2.631856
##
## $objective
## [1] 2617.344
##
## $convergence
## [1] 0
##
## $iterations
## [1] 15
##
## $evaluations
## function gradient
##       18       61
##
## $message
## [1] "relative convergence (4)"
```

```r
#The MLE estimate of the mean is 138.3119
#The MLE estimate of the standard deviation is 13.5427
#The MLE estimate of the shape parameter is 2.6319
```

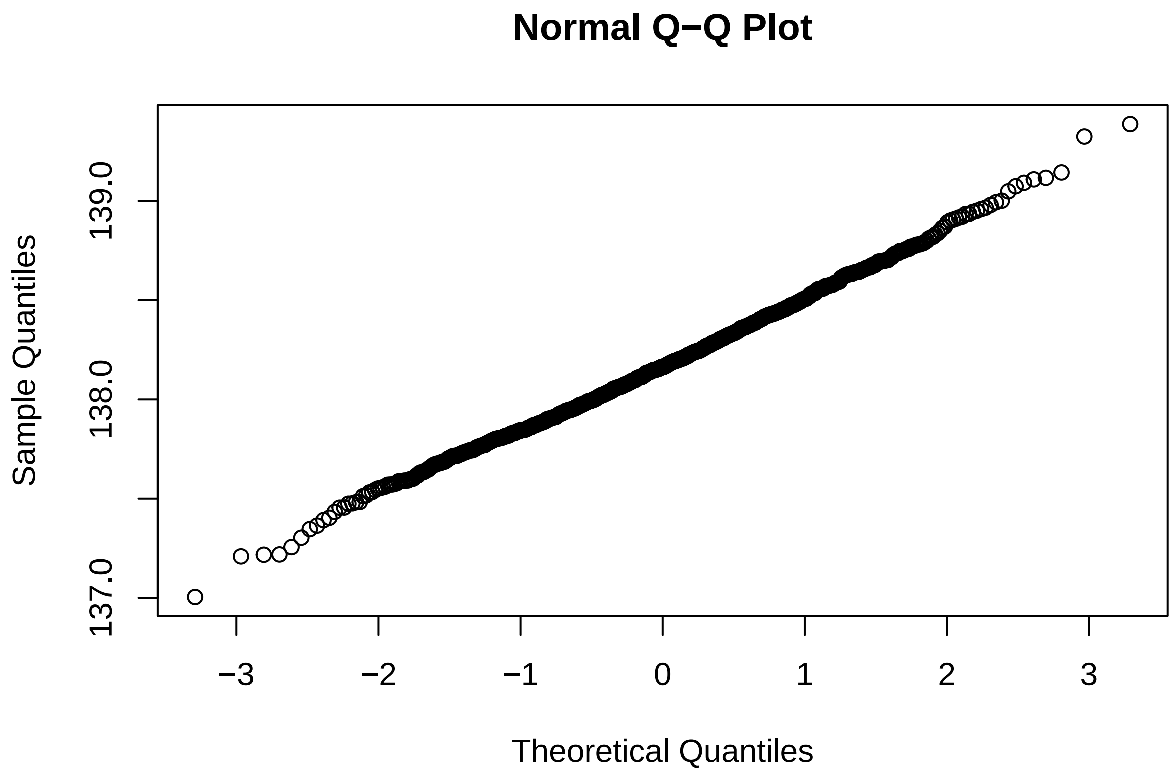
```r
#3
options(digits=5)
VSample=vector()
MSample=matrix(, nrow = 1000, ncol = 1000)
for (i in 1:1000){
  s=sample(Price,1000,replace=TRUE)
  MSample[i,]=s
  VSample <- c(VSample,mean(s))
}
mean_sample=mean(VSample)
mean_sample
```

```
## [1] 138.17
```

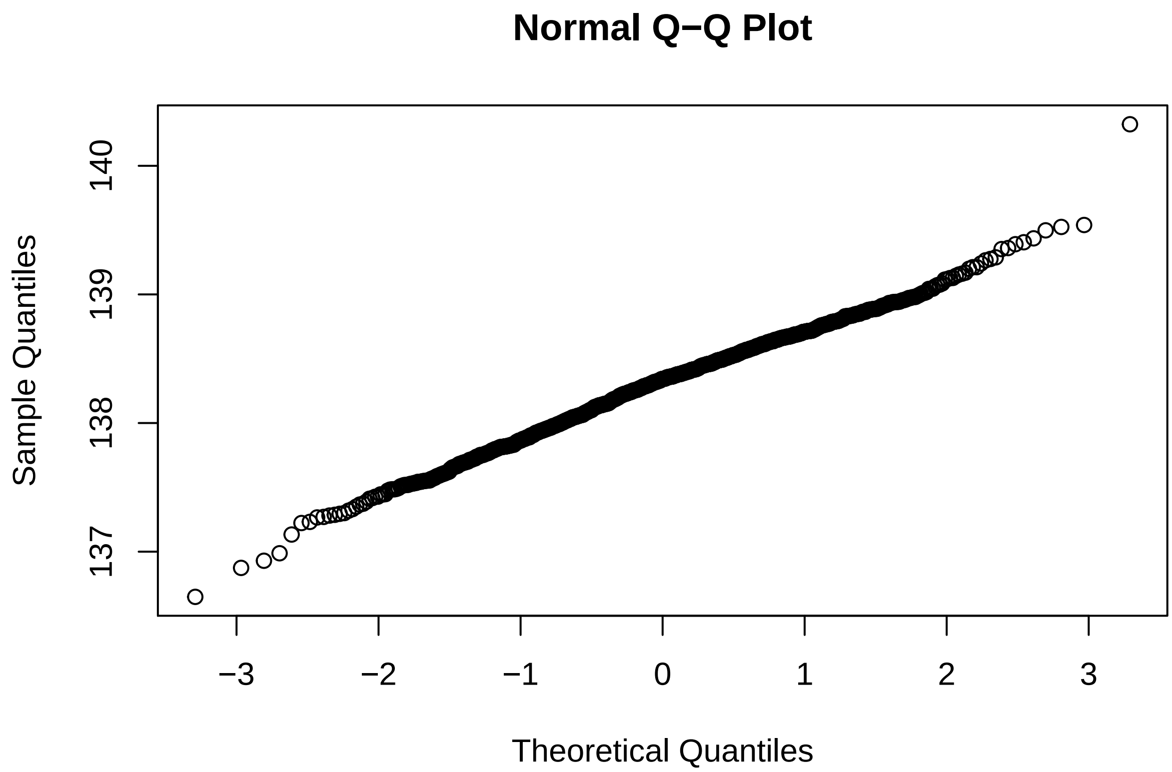
```r
VModel=vector()
MModel=matrix(, nrow = 1000, ncol = 1000)
for (i in 1:1000){
  Vector=rstd(1000,138.311938,13.542733,2.631856)
  MModel[i,]=Vector
  VModel <- c(VModel,mean(Vector))
}
mean_model=mean(VModel)
mean_model
```

```
## [1] 138.3
```

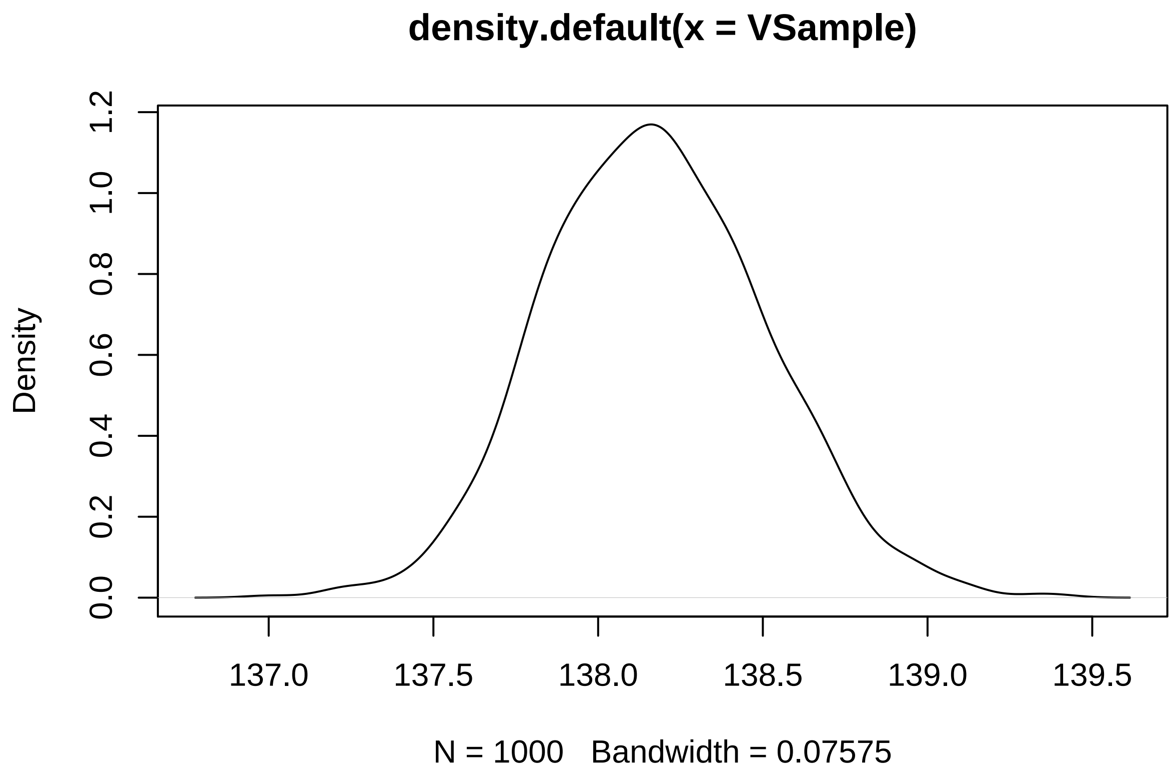
```r
#4
qqnorm(VSample)
```



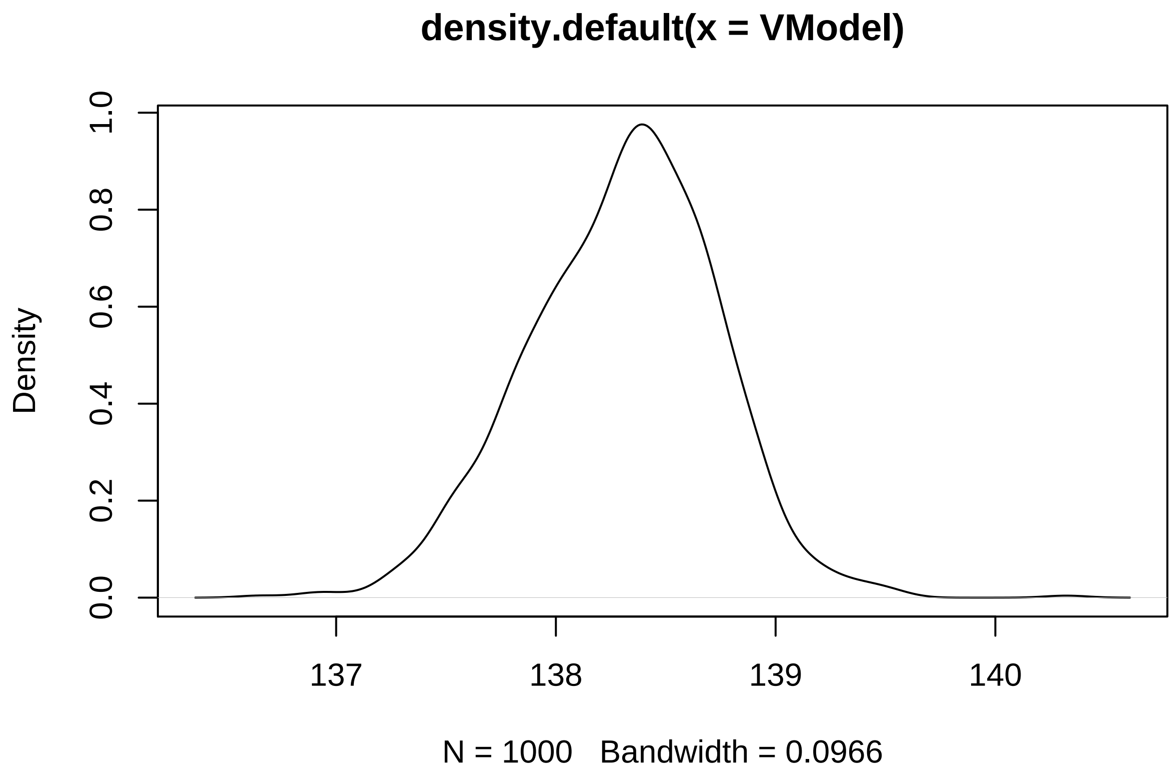

```
qqnorm(VModel)
```



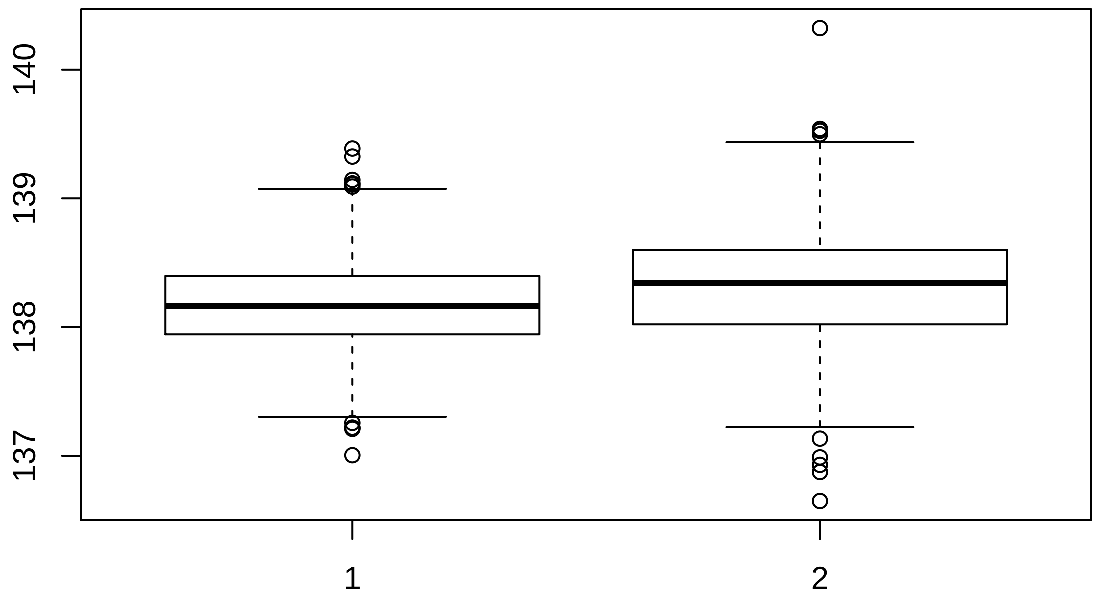

```
plot(density(VSample))
```




```r
plot(density(VModel))
```




```r
boxplot(VSample,VModel)
```




```
#As expected the plots show that the model-based distribution
#is skewed to the right, while the sample-based distribution
#is more symmetric.
```

```
#5
mean_diff=vector()
for (i in 1:1000){
  mean_diff=cbind(mean_diff,mean(MSample[1,i]-mean_sample))
}
q0.975=quantile(mean_diff,0.975)
q0.025=quantile(mean_diff,0.025)

#The lower boundary of the 95% model free confidence interval is:
Conf_lower=mean_sample-q0.975
#The upper boundary of the 95% model free confidence interval is:
Conf_upper=mean_sample-q0.025

mean_diff=vector()
for (i in 1:1000){
  mean_diff=cbind(mean_diff,mean(MModel[1,i]-mean_model))
}
q0.975=quantile(mean_diff,0.975)
q0.025=quantile(mean_diff,0.025)

#The lower boundary of the 95% model based confidence interval is:
Conf_lower=mean_model-q0.975
```

```r
#The upper boundary of the 95% model based confidence interval is:
Conf_upper=mean_model-q0.025
```

```r
BiasModelFree=mean_sample-mean(Price)
#The bias of the sample mean of IBM based on model-free bootstraps is:
BiasModelFree
```

```
## [1] -0.0087108
```

```r
BiasModel=mean_model-mean(Price)
#The bias of the sample mean of IBM based on model-based bootstraps is:
BiasModel
```

```
## [1] 0.12175
```

```r
#7
V=var(VSample)
MSE=V+BiasModelFree^2
#The mean squared error (MSE) of the sample mean of data IBM is:
MSE
```

```
## [1] 0.11234
```