# STA457 Homework 3

*Depeng Ye 1002079500*

*25/11/2019*

## Q1

Data read, crop, and overview are shown below.

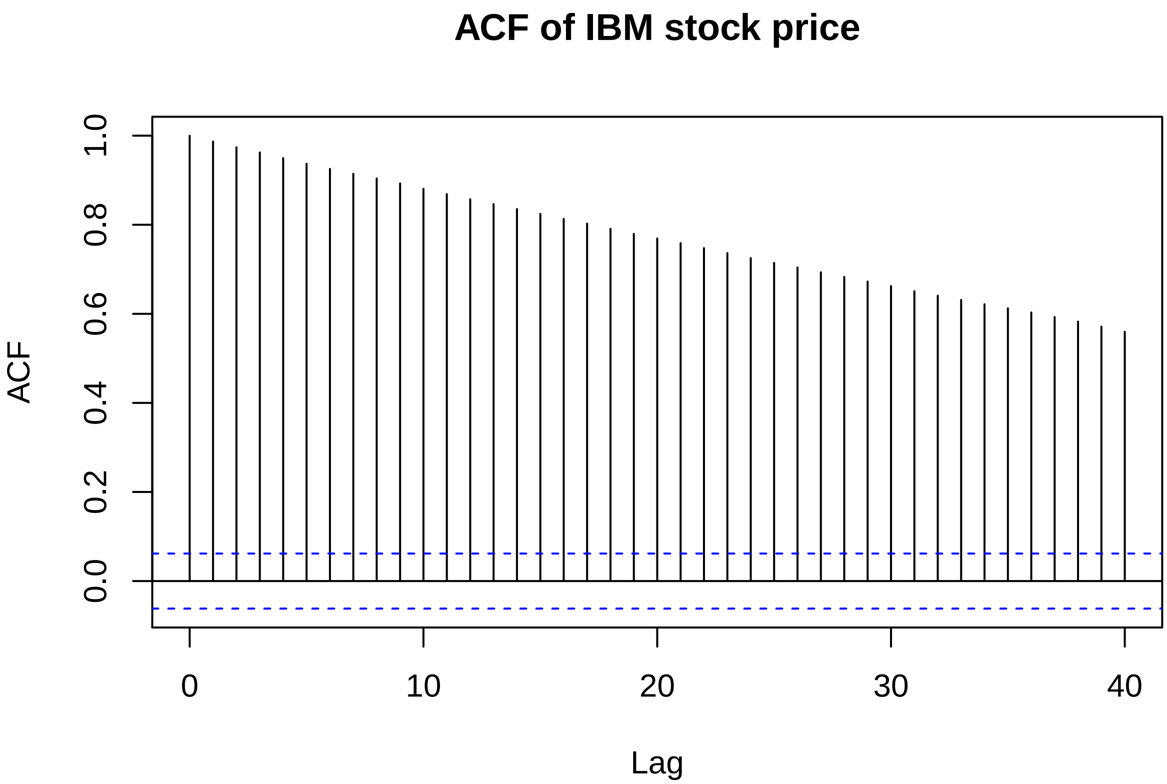
```
# read the csv file.
IBM = read.csv("IBM.csv", header = T)
# crop the csv file into our desired time period.
IBM_crop = IBM[203:1208,]
# show first month (in date) of our data.
Samp = data.frame(tail(IBM_crop, n = 20))
# sample data table shown below
```

| | Date | Adj.Close |
|---|---|---|
| 1189 | 2015-01-30 | 124.9993 |
| 1190 | 2015-01-29 | 126.7686 |
| 1191 | 2015-01-28 | 123.5643 |
| 1192 | 2015-01-27 | 125.2928 |
| 1193 | 2015-01-26 | 127.4861 |
| 1194 | 2015-01-23 | 127.0865 |
| 1195 | 2015-01-22 | 126.6952 |
| 1196 | 2015-01-21 | 124.0046 |
| 1197 | 2015-01-20 | 127.9671 |
| 1198 | 2015-01-16 | 128.1220 |

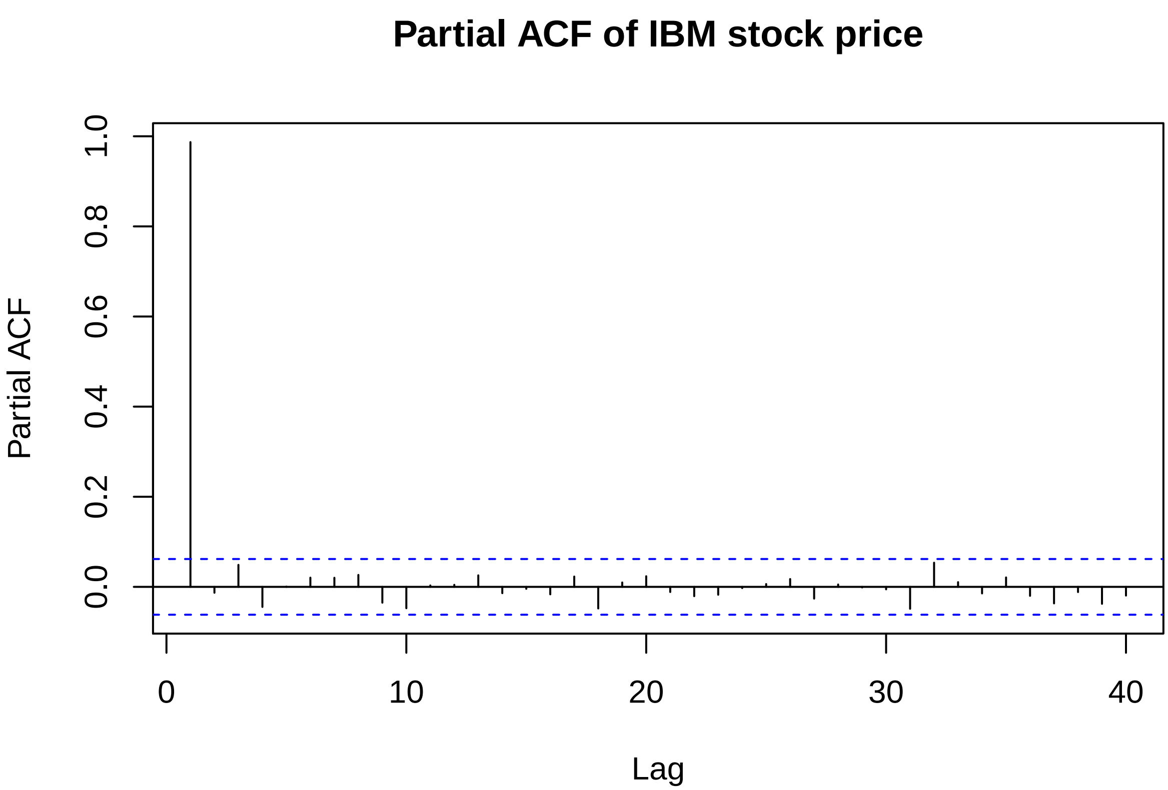
| | Date | Adj.Close |
|---|---|---|
| 1199 | 2015-01-15 | 126.0266 |
| 1200 | 2015-01-14 | 127.0295 |
| 1201 | 2015-01-13 | 127.8530 |
| 1202 | 2015-01-12 | 127.5513 |
| 1203 | 2015-01-09 | 129.7283 |
| 1204 | 2015-01-08 | 129.1657 |
| 1205 | 2015-01-07 | 126.4180 |
| 1206 | 2015-01-06 | 127.2496 |
| 1207 | 2015-01-05 | 130.0544 |
| 1208 | 2015-01-02 | 132.1335 |

## Q2

```
acf(IBM_crop$Adj.Close, lag = 40, main = "ACF of IBM stock price")
```




```r
pacf(IBM_crop$Adj.Close, lag = 40, main = "Partial ACF of IBM stock price")
```




We notice that the ACF follows a geometric pattern. Hence, our PACF should determine our value of $p$ and a AR model should be fitted.
Note that there is only the first lag is significant in the PACF plot. Therefore, we should use AR(1).

# Q3

We use the Ljung-Box test for such a simultaneous test for $\rho(1) = \rho(2) = \cdots = \rho(K)$ where $K = 5$ for IBM price.
$H_0$: $\rho(1) = \rho(2) = \ldots \rho(K)$ for $K = 5$.
$H_1$: one or more of $\rho(1)$, $\rho(2)$, ..., $\rho(K)$ is nonzero.

```
Box.test(IBM_crop$Adj.Close, lag = 5, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  IBM_crop$Adj.Close
## X-squared = 4680.7, df = 5, p-value < 2.2e-16
```

According to the result of Ljung-Box test, we have a $p$-value less than $2.2e$-16. We can strongly reject $H_0$ because the $p$-value is too small. Namely, it means that at least one of the 5 autocorrelations is nonzero.

# Q4

```
AR1Fit = arima(IBM_crop$Adj.Close, order = c(1, 0, 0))
AR1Fit
```

```
##
## Call:
## arima(x = IBM_crop$Adj.Close, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9905   132.6845
## s.e.  0.0043     5.1763
##
## sigma^2 estimated as 2.915:  log likelihood = -1967.57,  aic = 3941.14
```

```
#test the adquataness with K = 5, 10, 15, and 20
Box.test(residuals(AR1Fit), lag = 5, type = "Ljung-Box", fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  residuals(AR1Fit)
## X-squared = 5.5574, df = 4, p-value = 0.2347
```

```
Box.test(residuals(AR1Fit), lag = 10, type = "Ljung-Box", fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  residuals(AR1Fit)
## X-squared = 10.122, df = 9, p-value = 0.3407
```

```
Box.test(residuals(AR1Fit), lag = 15, type = "Ljung-Box", fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  residuals(AR1Fit)
```

```
## X-squared = 11.127, df = 14, p-value = 0.676
```

```r
Box.test(residuals(AR1Fit), lag = 20, type = "Ljung-Box", fitdf = 1)
```

```
##
##  Box-Ljung test
##
## data:  residuals(AR1Fit)
## X-squared = 15.92, df = 19, p-value = 0.6626
```

According to the test result of our fited AR1 model, we can conclude that the significance of rejecting the null hypothesis is not high, meaning that the residuals are uncorrelated for whatever value of lag. This is a sign that the AR(1) model is an adquate fit for our IBM stock price data.
AR(1) with estimated parameters:

$$\hat{Y}_t = 132.68 + 0.99Y_{t-1} + \epsilon_i$$

# Q5

This data set is stationary. Use KPSS test, ADF test, and Plillip-Peron test to test the stationarity. THe results are shown as follows:

```r
kpss.test(IBM_crop$Adj.Close)
```

```
## Warning in kpss.test(IBM_crop$Adj.Close): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  IBM_crop$Adj.Close
## KPSS Level = 1.9266, Truncation lag parameter = 7, p-value = 0.01
```

```r
adf.test(IBM_crop$Adj.Close)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  IBM_crop$Adj.Close
## Dickey-Fuller = -2.9457, Lag order = 10, p-value = 0.178
## alternative hypothesis: stationary
```

```r
pp.test(IBM_crop$Adj.Close)
```

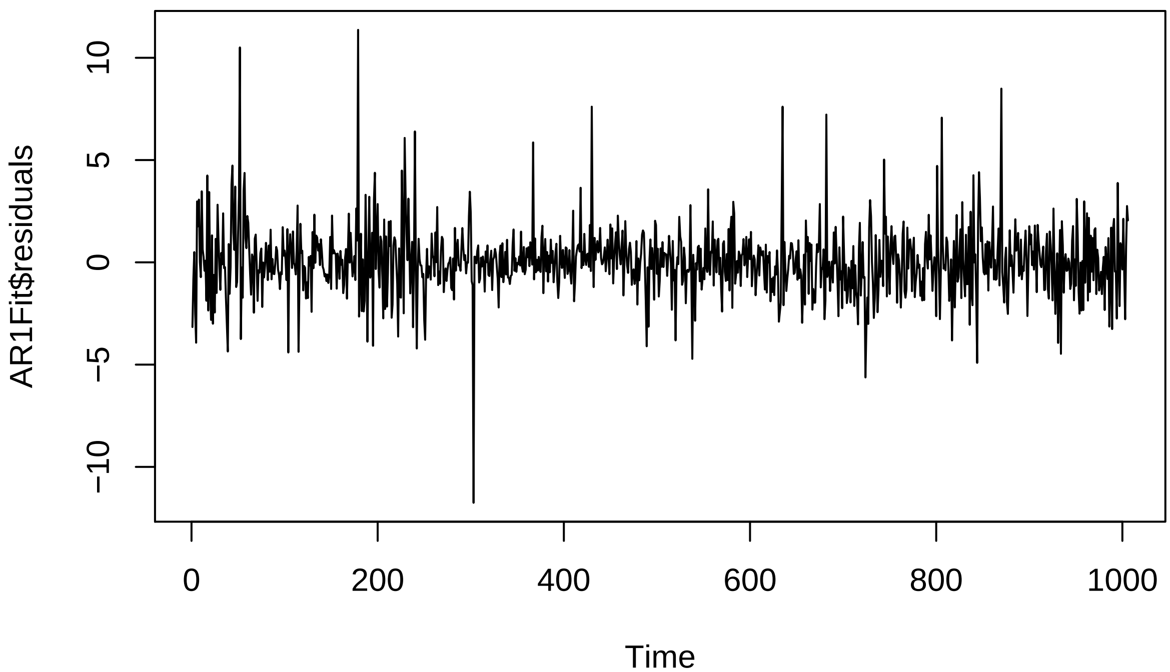
```
##
##  Phillips-Perron Unit Root Test
##
## data:  IBM_crop$Adj.Close
## Dickey-Fuller Z(alpha) = -13.637, Truncation lag parameter = 7, p-value
## = 0.349
## alternative hypothesis: stationary
```

Notice that in both P-P test and ADf test, stationary can not be rejected because of the large $p$-value, while in the KPSS test, non-stationary can be rejected with a sufficiently small $p$-value.
Hence, the data set is stationary.

## Q6

The noise $\epsilon_i$ of our AR(1) model is Gaussian because the estimated mean of the noise $\epsilon$ is 0.0043 and variance is 2.92 according to the outcome of our fitted AR(1) model. Hence, it seems to have a constant mean, and variance. The following plots can show constant mean and variance too.

```
plot(AR1Fit$residuals)
```




## Q7

Use auto.arima() function to find the best ARIMA model based on AIC.

```
AutoFit = auto.arima(IBM_crop$Adj.Close, max.p = 10, max.q = 10, d = 0, ic = "aic")
AutoFit
```

```
## Series: IBM_crop$Adj.Close
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2      mean
##       0.9963  -0.0062  133.9117
## s.e.  0.0315   0.0316    5.0050
##
## sigma^2 estimated as 2.924:  log likelihood=-1967.6
## AIC=3943.19   AICc=3943.23   BIC=3962.85
```

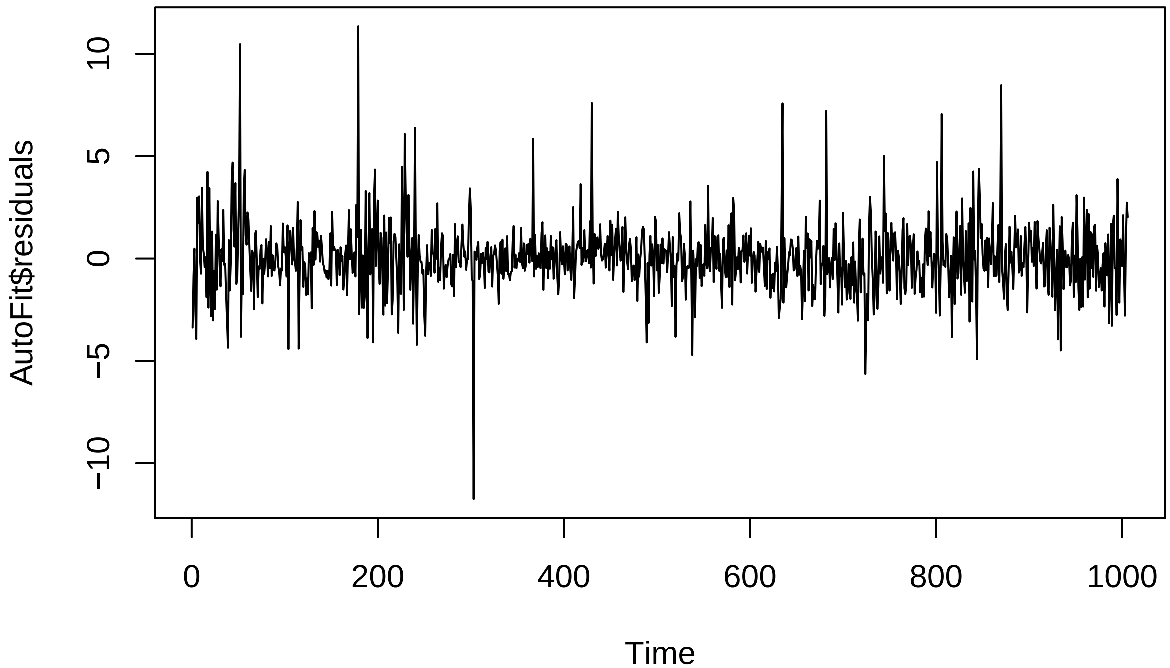
Notice that the result of ARIMA model fitted based on AIC is ARIMA(2, 0, 0). The model mathematically looks like:

$$\hat{Y}_t = 133.912 + 0.996Y_{t-1} - 0.006Y_{t-2} + \epsilon_t$$

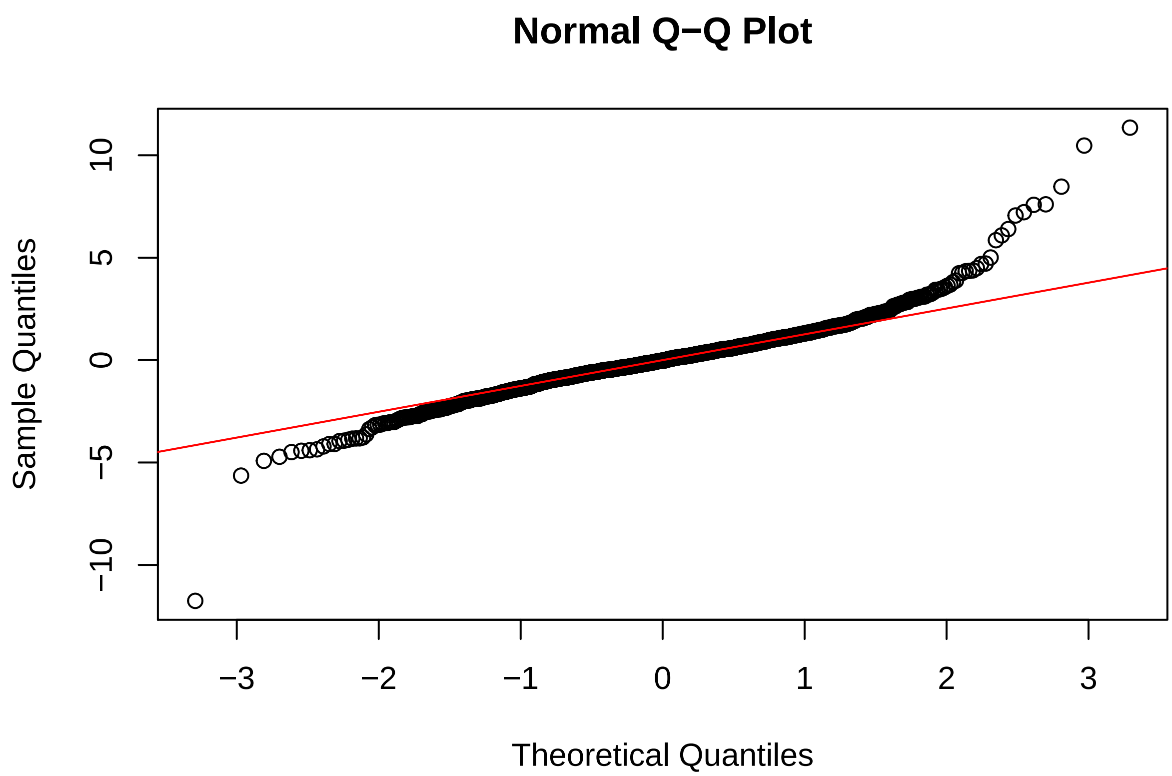
## Q8

Draw some plots of the residuls of the AIC fitted ARIMA model first, we can notice that the distribution shall be heavy tailed. Hence, we are trying to fit a *t*-distribution to the residuls.

```
plot(AutoFit$residuals)
```




```
qqnorm(AutoFit$residuals)
qqline(AutoFit$residuals, col = "red")
```



```r
#fit the t-distributino
tFit = fitdistr(AutoFit$residuals, "t")
tFit$estimate
```

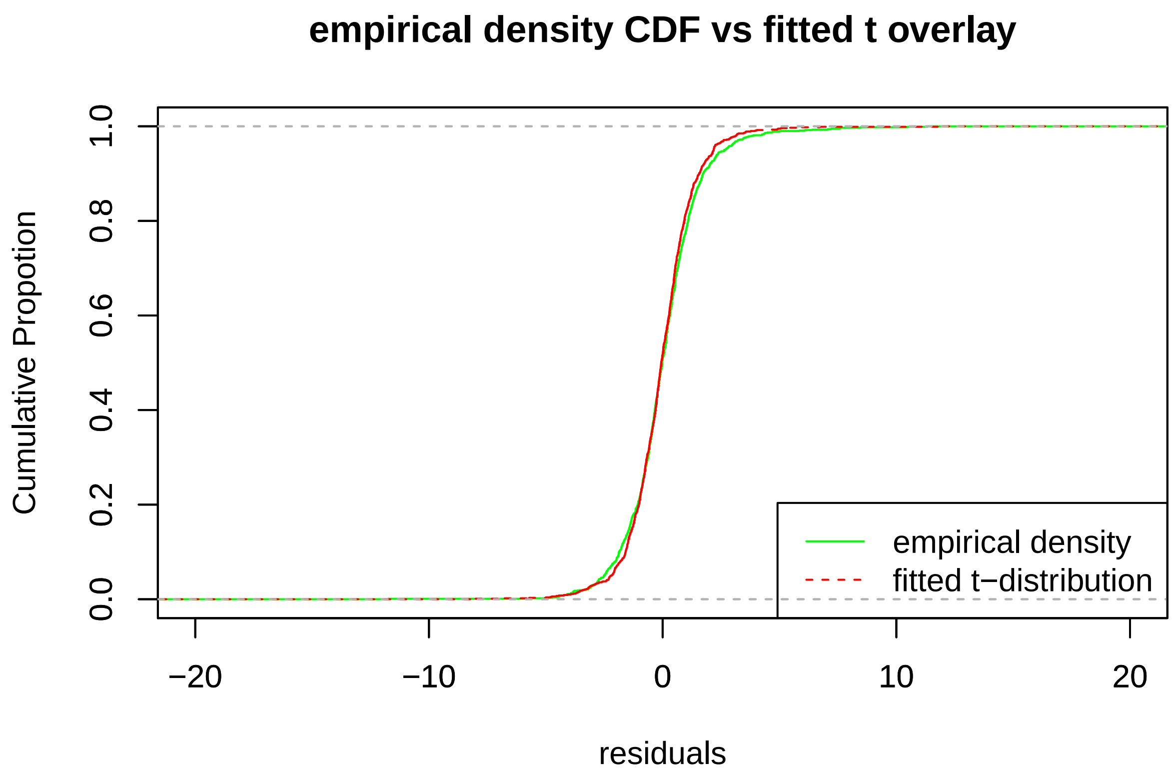
```
##           m           s          df
## -0.01819281  1.13658871  3.47205469
```

As we can see from the result of the fit, $\epsilon_t \sim t(3.47)$.

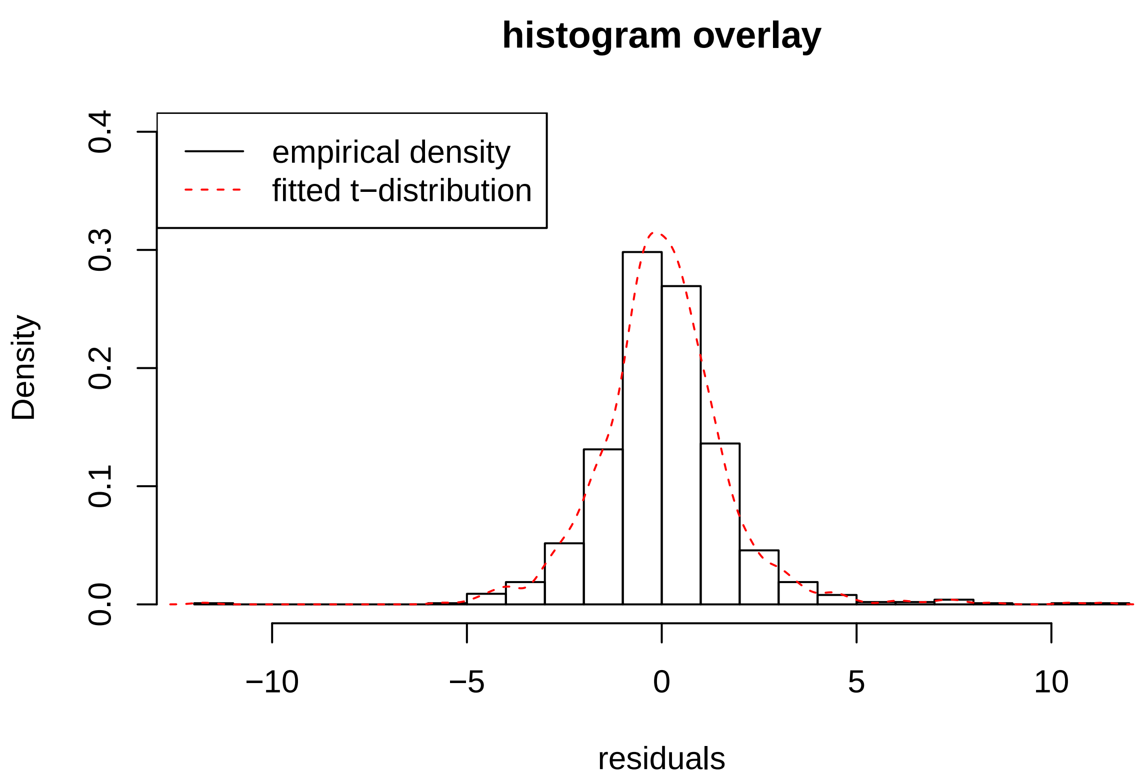
## Q9

As the quantile plot has ben shown in Q8, I will only do the overlay plot in this part.

```r
plot(ecdf(AutoFit$residuals), col = "green", xlim = c(-20, 20),
     main = "empirical density CDF vs fitted t overlay",
     ylab = "Cumulative Propotion", xlab = "residuals")
par(new = T)
plot(ecdf(rt(1006, tFit$estimate["df"])), col = "red",
     xlim = c(-20, 20), main = "", xlab = "", ylab = "", lty = 2)
legend("bottomright", legend = c("empirical density", "fitted t-distribution"),
       col = c("green", "red"), lty = 1 : 2)
```




```r
res = data.frame(AutoFit$residuals)
colnames(res) = "resid"
hist(res$resid, prob = T, ylim = c(0, 0.4), breaks = 20, main = "histogram overlay",
     xlab = "residuals")
lines(density(res$resid), col = "red", lty = 2)
legend("topleft", legend = c("empirical density", "fitted t-distribution"),
       col = c("black", "red"), lty = 1 : 2)
```




It is not hard to spot from the overlaied plot that the t-distribution is well fitted when compared to the empirical density.

## Q10

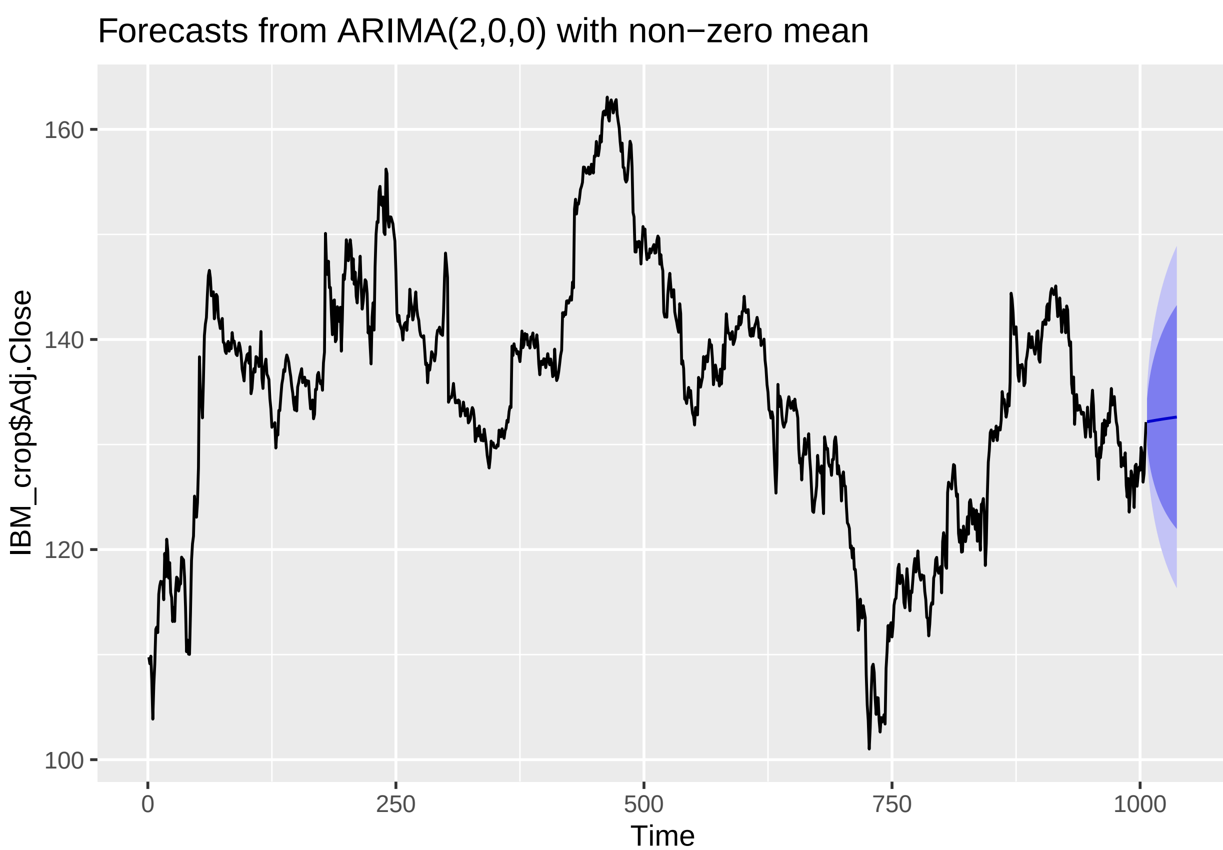
We can use the forecast() function to do a modelbased forecast of the month of January in 2019. The 95% confidence interval is (116.3, 148.9). plot of confidence band are shown below. The lighter blue represents 95% confident interval, while the darker blue represents 85%.

```
# forecasting the following 31 days using AR(2) model.
f = forecast(AutoFit, 31)
f
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1007       132.1639 129.9725 134.3552 128.8125 135.5153
## 1008       132.1813 129.0878 135.2747 127.4503 136.9122
## 1009       132.1984 128.4245 135.9724 126.4266 137.9702
## 1010       132.2154 127.8767 136.5540 125.5799 138.8508
## 1011       132.2322 127.4036 137.0608 124.8475 139.6168
## 1012       132.2488 126.9841 137.5136 124.1971 140.3006
## 1013       132.2653 126.6055 137.9251 123.6094 140.9212
## 1014       132.2816 126.2596 138.3036 123.0718 141.4914
## 1015       132.2977 125.9408 138.6547 122.5756 142.0199
## 1016       132.3137 125.6446 138.9828 122.1142 142.5132
## 1017       132.3296 125.3681 139.2910 121.6829 142.9762
## 1018       132.3452 125.1086 139.5819 121.2777 143.4127
## 1019       132.3607 124.8642 139.8573 120.8957 143.8258
## 1020       132.3761 124.6332 140.1191 120.5343 144.2179
## 1021       132.3913 124.4143 140.3684 120.1915 144.5912
## 1022       132.4064 124.2063 140.6065 119.8654 144.9473
```

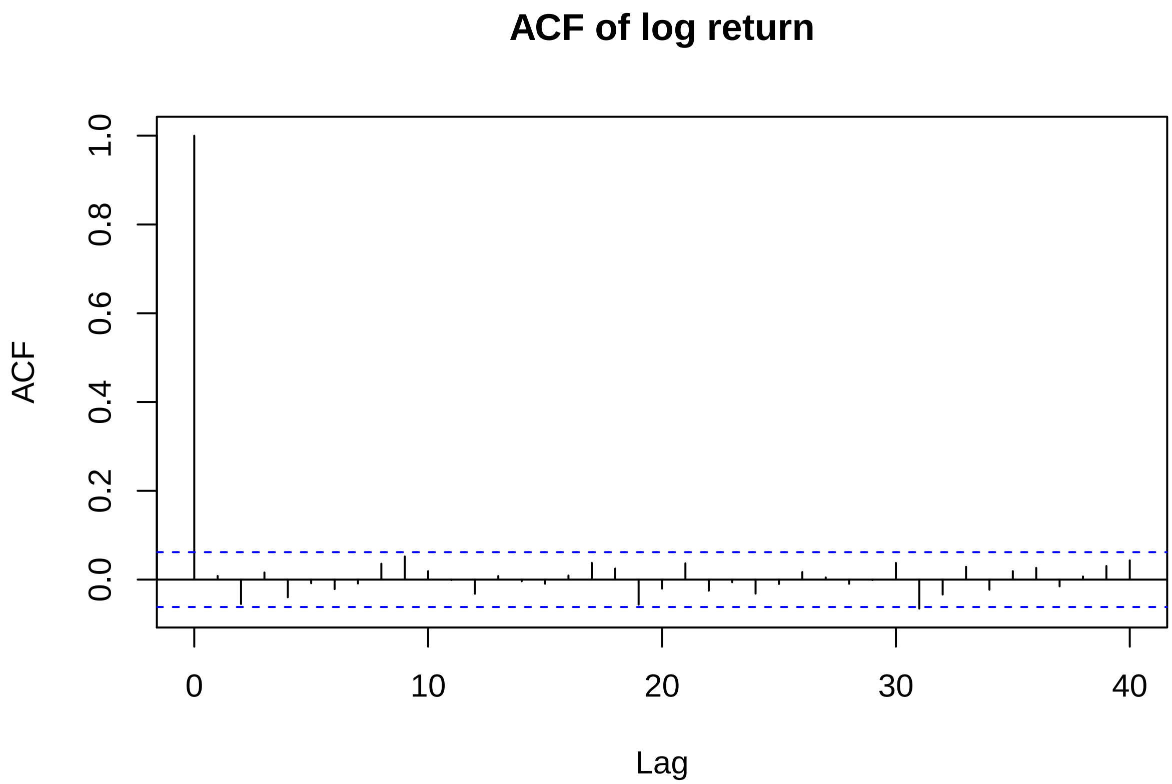
```
## 1023       132.4213 124.0083 140.8342 119.5548 145.2878
## 1024       132.4361 123.8195 141.0526 119.2582 145.6139
## 1025       132.4507 123.6391 141.2622 118.9746 145.9268
## 1026       132.4651 123.4665 141.4638 118.7030 146.2273
## 1027       132.4795 123.3012 141.6578 118.4425 146.5164
## 1028       132.4937 123.1426 141.8447 118.1924 146.7949
## 1029       132.5077 122.9903 142.0251 117.9521 147.0633
## 1030       132.5216 122.8439 142.1993 117.7209 147.3223
## 1031       132.5354 122.7031 142.3676 117.4982 147.5725
## 1032       132.5490 122.5675 142.5305 117.2837 147.8144
## 1033       132.5625 122.4369 142.6881 117.0767 148.0483
## 1034       132.5759 122.3109 142.8408 116.8770 148.2748
## 1035       132.5891 122.1894 142.9888 116.6841 148.4941
## 1036       132.6022 122.0721 143.1324 116.4977 148.7067
## 1037       132.6152 121.9587 143.2717 116.3175 148.9129
```

```r
autoplot(f)
```

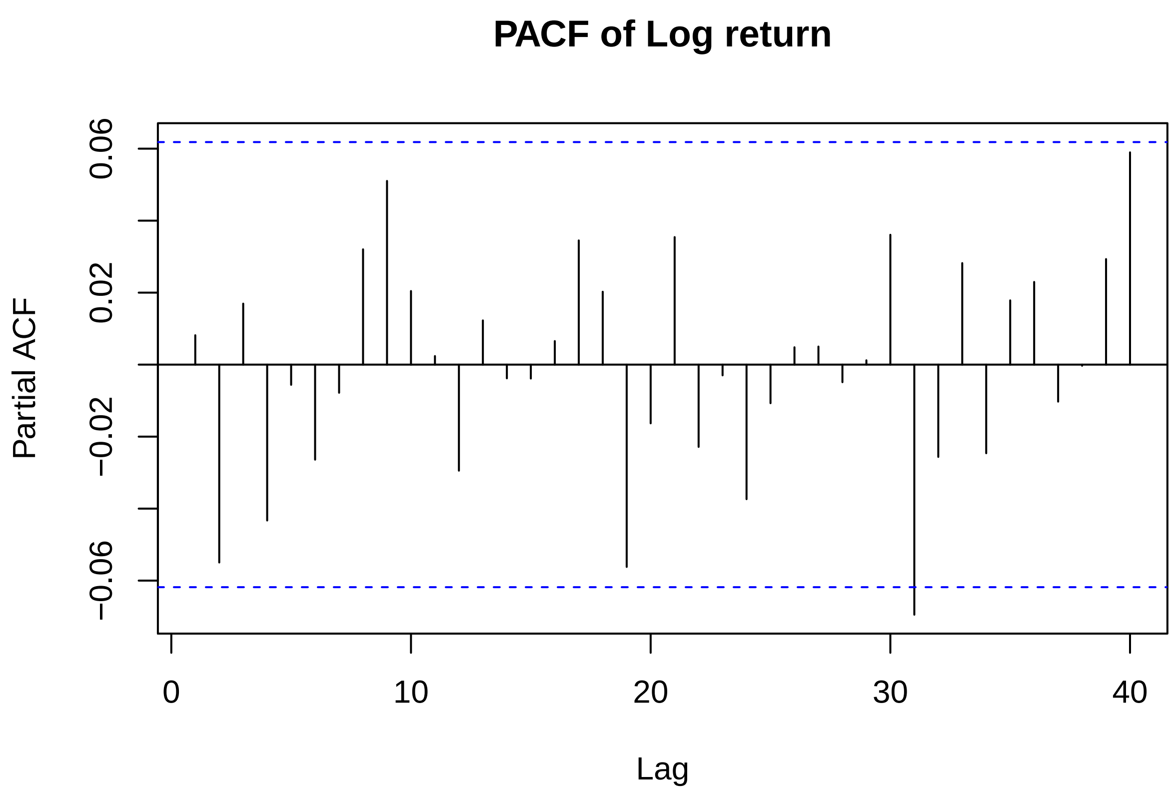



## Q11

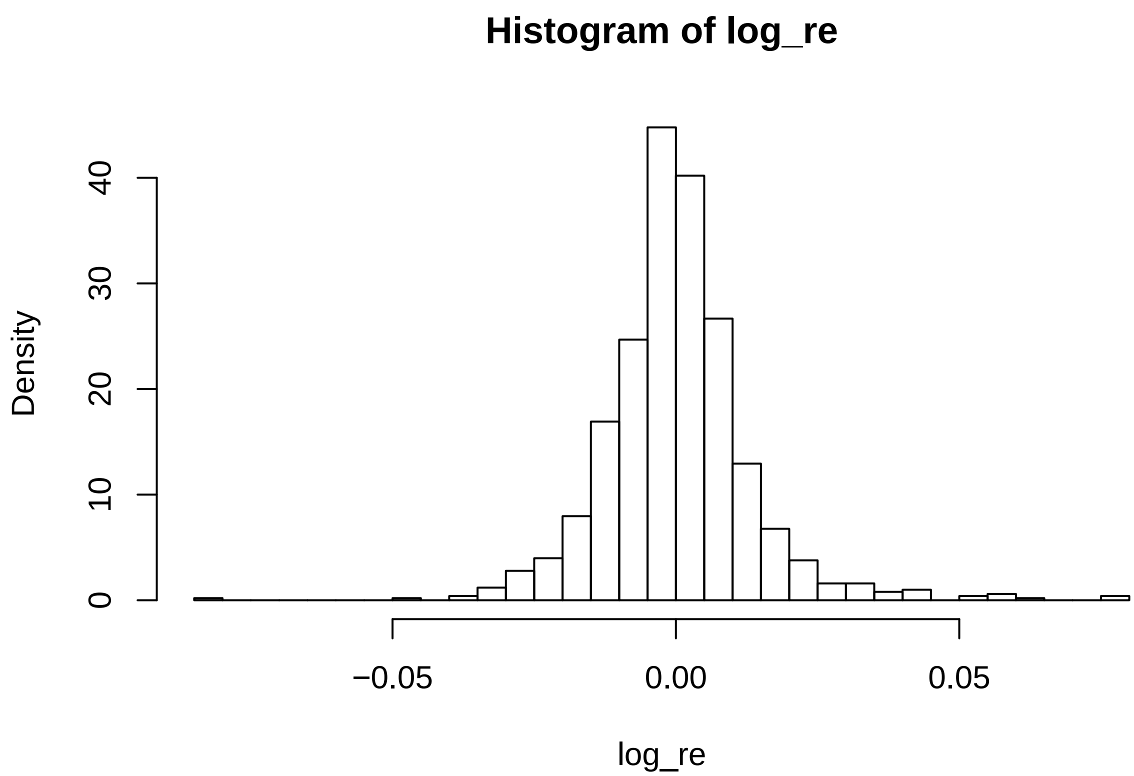
```r
log_re = diff(c(NA, log(IBM_crop$Adj.Close)))
log_re = na.omit(log_re)
acf(log_re, lag = 40, main = "ACF of log return")
```



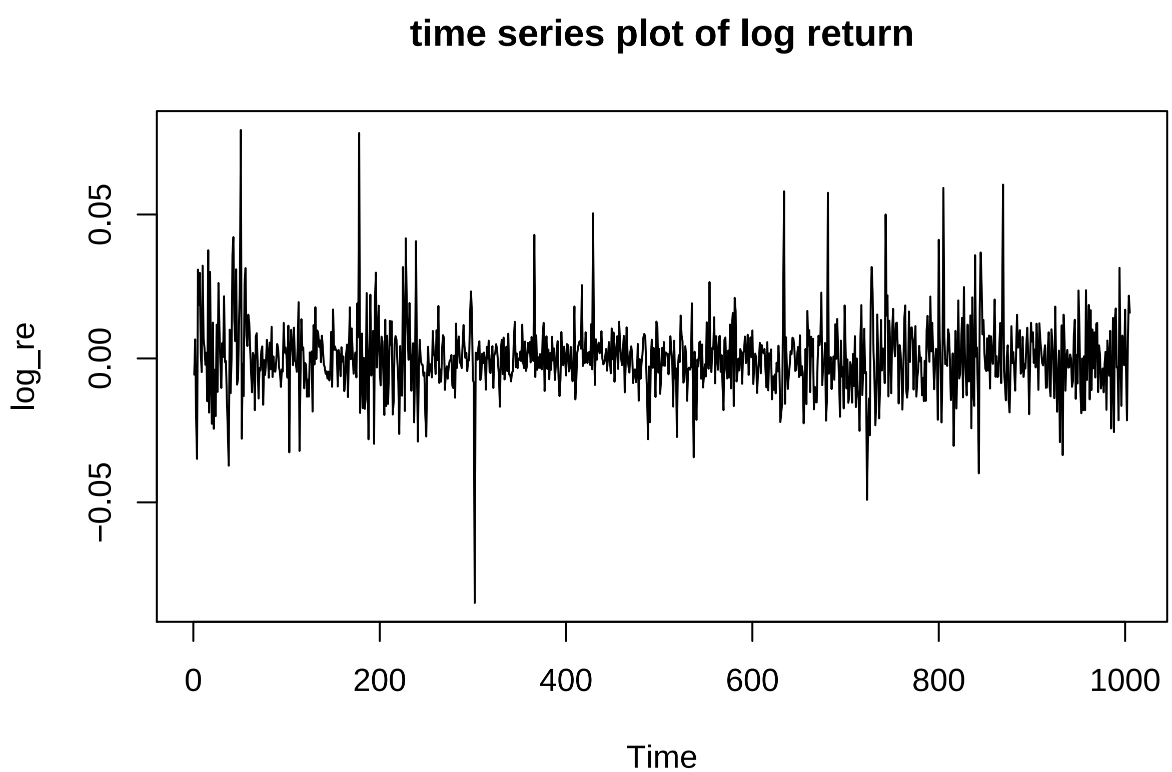

```r
pacf(log_re, lag = 40, main = "PACF of Log return")
```



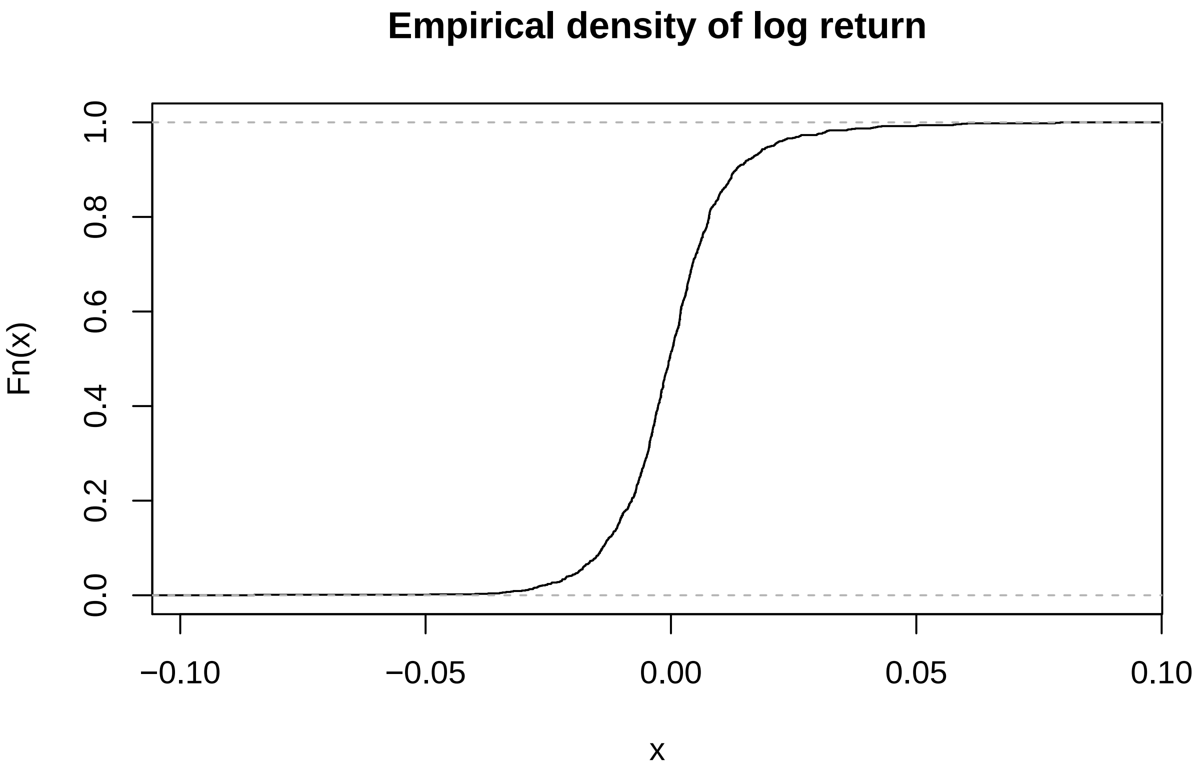

```r
hist(log_re, breaks = 25, prob = T)
```




```r
ts.plot(log_re,main = "time series plot of log return")
```




```r
plot(ecdf(log_re), main = "Empirical density of log return")
```




With all those plots shown above, it is not hard to notice that log return of the closing price of IBM is a stable time series data set, log return is a more unbiased set of data when compared to closing price. Moreover, log return is a time additive return instead of a portfolio additive return, meaning it can be added to a sum when it comes to a multi time period analysis. Adding multiple log returns will not change the distribution of a single return, which is a good and important point when it comes to data analysis that we don't need to fit a new model to the log return when changing the length of time period.