TSA Assignment3

Debangshu Bhattacharya (MDS201910)

04/11/2020

Importing Libraries

```
library(tseries)
library(forecast)
library(rugarch)
```

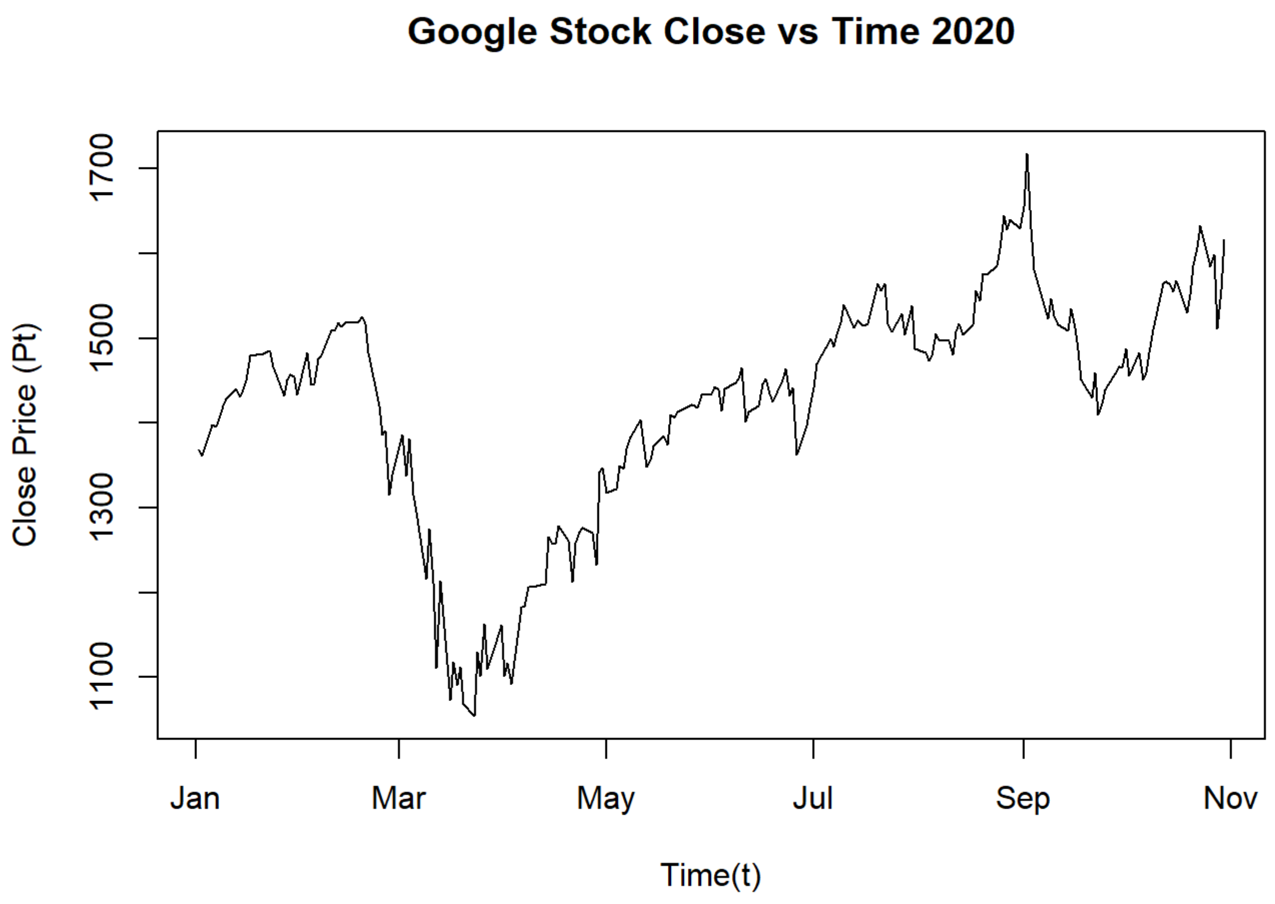
1. Reading data, making it stationary and splitting into train and test set:

Reading Google stock price data from 1st January 2020 to 31st October 2020.

```
start_date <- as.Date("2020-01-01")
end_date <- as.Date("2020-10-31")
google_stock<-get.hist.quote(instrument = "GOOGL"
                                      ,start=start_date,end=end_date
                                      ,quote="Close",provider = "yahoo")
```

```
## time series starts 2020-01-02
## time series ends   2020-10-30
```

```
plot(google_stock, xlab = "Time(t)", ylab ="Close Price (Pt)", main = "Google Stock Close vs
 Time 2020
     ")
```



```
data = ts(google_stock)
```

Stationarity Test of the stock price series :

```
adf.test(data)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -2.1001, Lag order = 5, p-value = 0.5341
## alternative hypothesis: stationary
```

```
kpss.test(data)
```

```
## Warning in kpss.test(data): p-value smaller than printed p-value
```

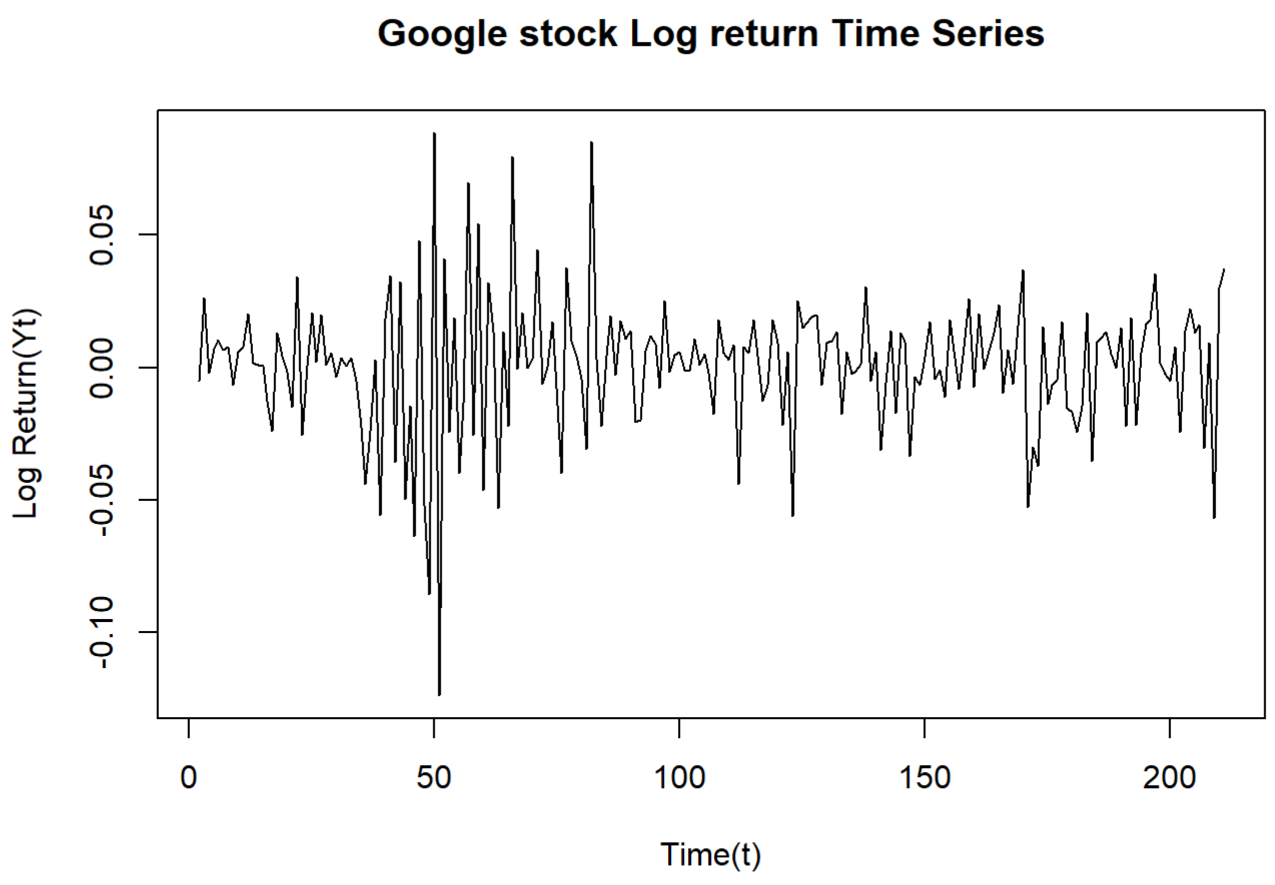
```
##
##  KPSS Test for Level Stationarity
##
## data:  data
## KPSS Level = 1.8117, Truncation lag parameter = 4, p-value = 0.01
```

```
pp.test(data)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  data
## Dickey-Fuller Z(alpha) = -7.2673, Truncation lag parameter = 4, p-value
## = 0.7006
## alternative hypothesis: stationary
```

We can see that for ADF and PP test we cannot reject the null hypothesis of non-stationarity while for KPSS test the null hypothesis of stationarity is rejected for 5% level of significance. Thus, clearly the stock price series is not stationary. So, now we will check for the stationarity of the log return series.

```
lrt = diff(log(data))
plot(lrt, xlab = "Time(t)", ylab = "Log Return(Yt)",
     main = "Google stock Log return Time Series")
```



```
adf.test(lrt)
```

```
## Warning in adf.test(lrt): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  lrt
## Dickey-Fuller = -5.7366, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(lrt)
```

```
## Warning in kpss.test(lrt): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  lrt
## KPSS Level = 0.10914, Truncation lag parameter = 4, p-value = 0.1
```

```
pp.test(lrt)
```

```
## Warning in pp.test(lrt): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  lrt
## Dickey-Fuller Z(alpha) = -291.64, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary
```

From the stationarity tests, we can see that the log return series is stationary. So, we are going to build our model on the log return series instead of the original closing stock price series.

Splitting data into train and test set where test set is the last 6 observations:

```
period = 6
n<- length(lrt)
train_lim = n - period
train_data = lrt[c(1:train_lim)]
test_data = lrt[c((train_lim+1):n)]

length(train_data)
```

```
## [1] 204
```

```
length(test_data)
```

```
## [1] 6
```

```
d<-ndiffs(train_data)
```

We can see that there are 204 observations in the train set for model building and 6 observations in the test set.

2. Fitting Appropriate ARIMA model

AIC model

```
aic_model = auto.arima(train_data,max.p = 5, max.q = 5,d = d, ic = "aic")
aic_model
```

```
## Series: train_data
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##           ar1
##       -0.3356
## s.e.   0.0658
##
## sigma^2 estimated as 0.0005843:  log likelihood=470.38
## AIC=-936.76   AICc=-936.7   BIC=-930.12
```

BIC model

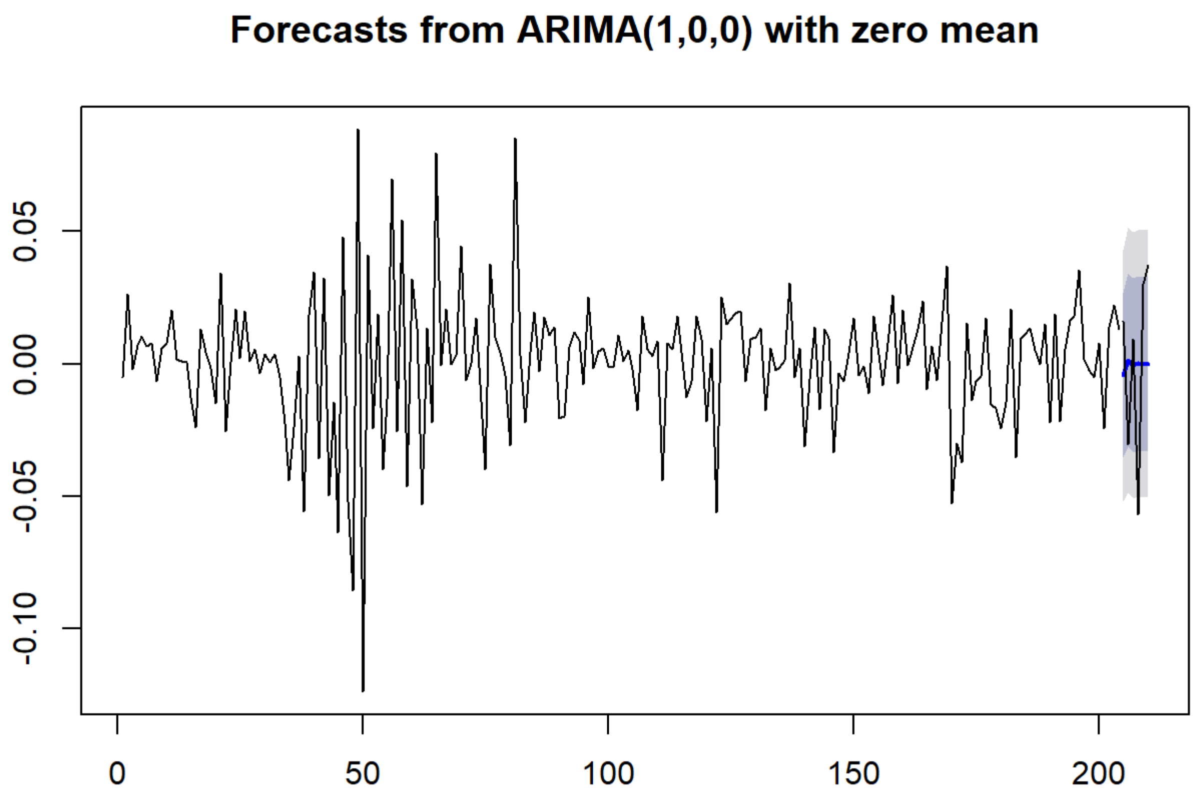
```
bic_model = auto.arima(train_data,max.p = 5, max.q = 5,d =d, ic = "bic")
bic_model
```

```
## Series: train_data
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##           ar1
##       -0.3356
## s.e.   0.0658
##
## sigma^2 estimated as 0.0005843:  log likelihood=470.38
## AIC=-936.76   AICc=-936.7   BIC=-930.12
```

We can see that for AIC and BIC selection criteria an ARMA(1,0) model is selected as the best model.

```
pred_arima<- forecast(aic_model, h = period)
arima_forecast<- pred_arima$mean

plot(forecast(aic_model,h=period),type='l')
lines(c((train_lim+1):n),test_data, col = 'black')
```



Getting the residuals of ARMA(1,0) model.

```
residuals = aic_model$residuals
residuals2 = residuals ^ 2
```

Mean squared error for ARIMA forecast:

```
mse_arima = sum((test_data - arima_forecast)^2)
mse_arima
```

```
## [1] 0.007052334
```

3. Performing Ljung-Box test on residuals and residuals squared:

```
Box.test(residuals, lag = log(length(train_data)), type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  residuals
## X-squared = 6.3032, df = 5.3181, p-value = 0.3127
```

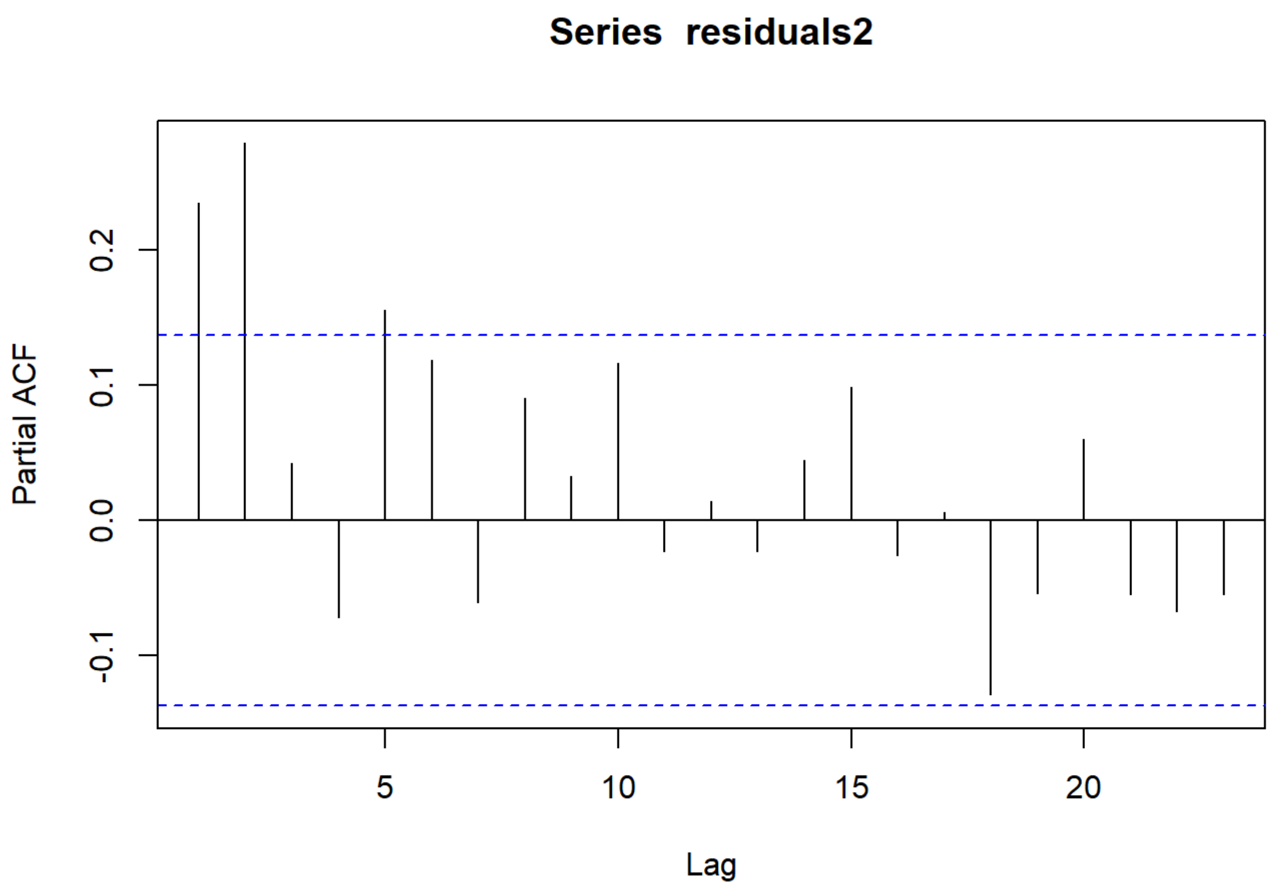
```
Box.test(residuals2,lag = log(length(train_data)), type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  residuals2
## X-squared = 46.224, df = 5.3181, p-value = 1.206e-08
```

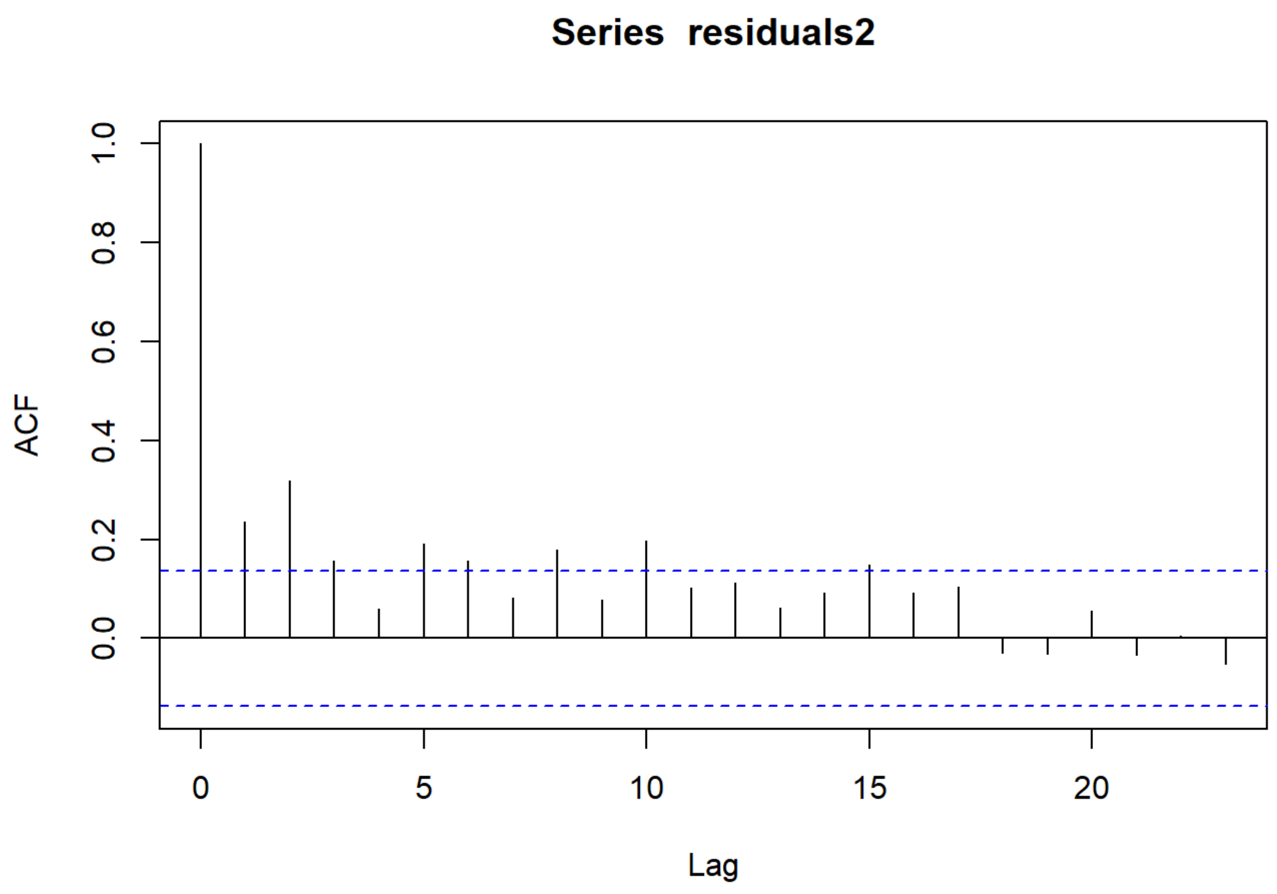
The number of lags considered for Ljung-Box test is log of the length of train data as that is considered empirically to be a good enough value. From the Ljung Box test on the residuals we see that the null hypothesis of “no auto-correlation” cannot be rejected for a 5% level of significance whereas for the Ljung-Box test on the squared residuals we see that the null hypothesis of “no auto-correlation” is rejected for a 5% level of significance.

Thus, we look to fit an appropriate GARCH model:

```
pacf(residuals2)
```



```
acf(residuals2)
```



From the ACF and PACF plots of the squared residuals, GARCH(2,2) seems to be an appropriate model. However, empirically GARCH(1,1) is a decent model for financial data. So, we compare the AIC and BIC values of these two models and make our model selection.

```
spec = ugarchspec(variance.model = list(model = "sGARCH",
                                        garchOrder = c(1,1)),
                  mean.model     = list(armaOrder = c(1, 0)))

garch_lrt <- ugarchfit(spec = spec, data = train_data)
infocriteria(garch_lrt)
```

```
##
## Akaike       -4.818235
## Bayes        -4.736909
## Shibata      -4.819399
## Hannan-Quinn -4.785337
```

```
spec = ugarchspec(variance.model = list(model = "sGARCH",
                                        garchOrder = c(2,2)),
                  mean.model     = list(armaOrder = c(1, 0)))

garch_lrt <- ugarchfit(spec = spec, data = train_data)
infocriteria(garch_lrt)
```

```
##
## Akaike       -4.810990
## Bayes        -4.697133
## Shibata      -4.813242
## Hannan-Quinn -4.764932
```

However, from the AIC values, it is clear that the GARCH(1,1) model is better as it has a lower AIC and BIC value. So, we will use this model.

```
spec = ugarchspec(variance.model = list(model = "sGARCH",
                                        garchOrder = c(1,1)),
                  mean.model     = list(armaOrder = c(1, 0)))

garch_lrt <- ugarchfit(spec = spec, data = train_data)

pred_garch = ugarchforecast(garch_lrt, data = train_data, n.ahead = period)

forecast_garch<- pred_garch@forecast$seriesFor

sigma_forecast<- pred_garch@forecast$sigmaFor
```

Computing the mean squared error for our GARCH model forecast on the test set.

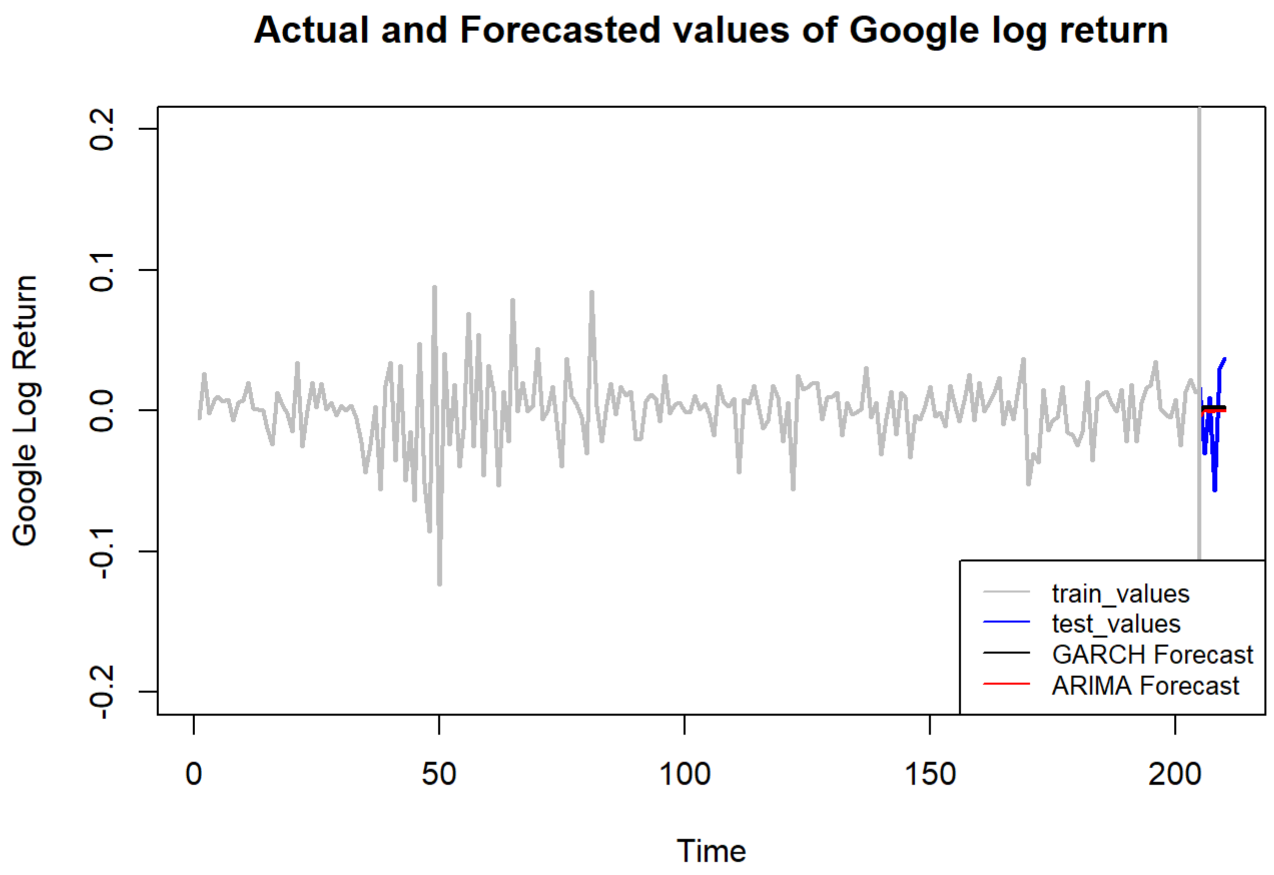
```
mse_garch = sum((test_data - forecast_garch)^2)
mse_garch
```

```
## [1] 0.006875429
```

4. Putting it all together:

Forecast Plot:

```
plot(NULL,xlim = c(1,n),ylim = c(-0.2,0.2),ylab = 'Google Log Return',xlab='Time',
     main = 'Actual and Forecasted values of Google log return')
points(c(1:train_lim), train_data,type='l',lwd=2,col='grey')
points(c(1:n), lrt, type = 'l', col = 'grey')
points(c((train_lim+1):n), test_data,type='l',lwd=2,col='blue')
points(c((train_lim+1):n), arima_forecast, type='l', lwd = 2, col = 'red')
points(c((train_lim+1):n), forecast_garch,type='l',lwd=2,col='black')
abline(v = (train_lim+1), col = 'grey', lwd = 2)
legend("bottomright",
       legend =  c("train_values","test_values","GARCH Forecast", "ARIMA Forecast"),
       col = c("grey","blue","black", "red"),lty = 1,
       cex = 0.8)
```



Comparison of MSE for ARIMA and GARCH:

```
cbind(mse_arima, mse_garch)
```

```
##        mse_arima   mse_garch
## [1,] 0.007052334 0.006875429
```

Thus, we can see that GARCH model has a lower MSE on the forecast values than ARIMA for log returns of GOOGLE Stock Close Price data.