Time Series Forecasting of Reliance Industry Limited Stock Price

Group No. 6

1. Praveen Kumar P (2022H1540826P)

2. Prasad A (2022H1540814P)

3. Sanjeev S (2022H1540837P)

4. R Naven (2022H1540835P)

5. Parul Shrivastava (2022H1540845P)

04-05-2023

Table of Contents

1. Statement of purpose: -

- To find the best fit model for predicting the stock price of Reliance Industries Limited and to predict the future prices of the stock.
- **Key issues raised**: To check for the influence of trend,seasonality on the stock price time series and to also ensure the series was stationary.

2. Background: -

- For the project requirements,the background reading involved analysing stock prices of various companies which generally could be converted as an effective time series data and Reliance stock prices were the one with which we decided to move ahead.
- We obtained the reliance stock price dataset using the getSymbols function from the quantmod library in R Programming Language to conceptualize our project.
- We then used the library rugarch comprises of functions that are required for specifying and fitting the model.
- We then used the tseries model to convert the data into a time series data.
- We then used the tidyverse model for data manipulation and initialising the time series as a data frame.

3,4. Methods and Results: -

Steps: - 1). The project's objective was to forecast the stock prices of Reliance Industries Limited at a future time period by creating an effective time series model and to check the accuracy of the model chosen by comparing the test data and the predicted data. - 2). We then tested the GJRGARCH model with the dataset and used the functions in R language like ugarchfit and ugarchforecast to obtain the results. - 3). Using the code below,we had determined the good fit model and the forecasted stock prices from the forecasted returns.

```
library(quantmod) # Scrapping

## Warning: package 'quantmod' was built under R version 4.2.3

## Loading required package: xts

## Warning: package 'xts' was built under R version 4.2.2

## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.2.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Warning: package 'TTR' was built under R version 4.2.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

library(rugarch) # Specification and Estimation

## Warning: package 'rugarch' was built under R version 4.2.3

## Loading required package: parallel

##
## Attaching package: 'rugarch'

## The following object is masked from 'package:stats':
##
##     sigma

library(xts) # Manipulation
library(PerformanceAnalytics) # Analyze the performance

## Warning: package 'PerformanceAnalytics' was built under R version 4.2.3

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.2.3

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last
```

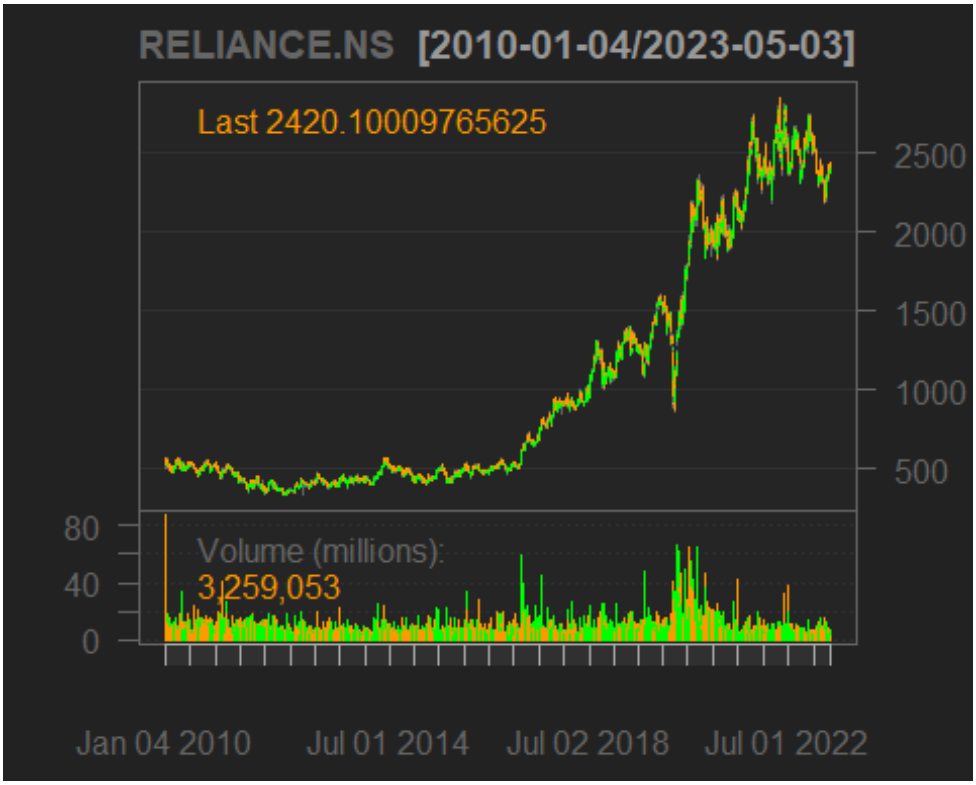
```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyverse)

## ── Attaching packages
## ─────────────────────────────────────
## tidyverse 1.3.2 ──

## ✓ ggplot2 3.3.6     ✓ purrr   0.3.4
## ✓ tibble  3.2.1     ✓ stringr 1.4.1
## ✓ tidyr   1.2.0     ✓ forcats 0.5.2
## ✓ readr   2.1.2

## Warning: package 'tibble' was built under R version 4.2.3

## ── Conflicts ──────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::first()  masks xts::first()
## ✕ dplyr::lag()    masks stats::lag()
## ✕ dplyr::last()   masks xts::last()
## ✕ purrr::reduce() masks rugarch::reduce()

library(tseries)

## Warning: package 'tseries' was built under R version 4.2.2

library(forecast)

## Warning: package 'forecast' was built under R version 4.2.3
```

Inference: - - The library quantmod comprises of functions that are required for scrapping. - The library rugarch comprises of functions that are required for estimation, filtering, forecasting, simulation, inference tests and plots. - The library xts comprises of functions that provide extensible time series class, enabling uniform handling of many R time series classes. - The library PerformanceAnalytics provides an R package of econometric functions for performance and risk analysis of financial instruments or portfolios.

For our project,we are considering the Reliance Stock price dataset - "RELIANCE.NS" which we would convert into a time series and then use specific models like ARMA-GJRGARCH to predict its future stock price.

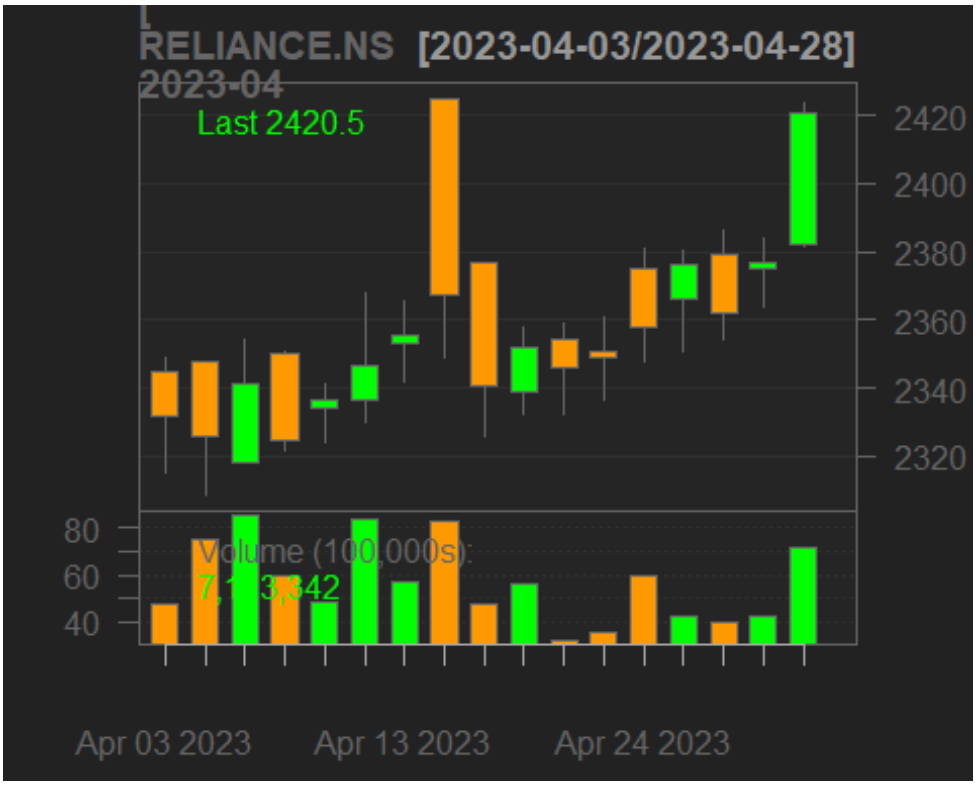
```
getSymbols("RELIANCE.NS", from = "2010-01-01", to = Sys.Date())

## Warning: RELIANCE.NS contains missing values. Some functions will not work
## if objects contain missing values in the middle of the series. Consider
```

```
using
## na.omit(), na.approx(), na.fill(), etc to remove or replace them.

## [1] "RELIANCE.NS"

chartSeries(RELIANCE.NS)
```

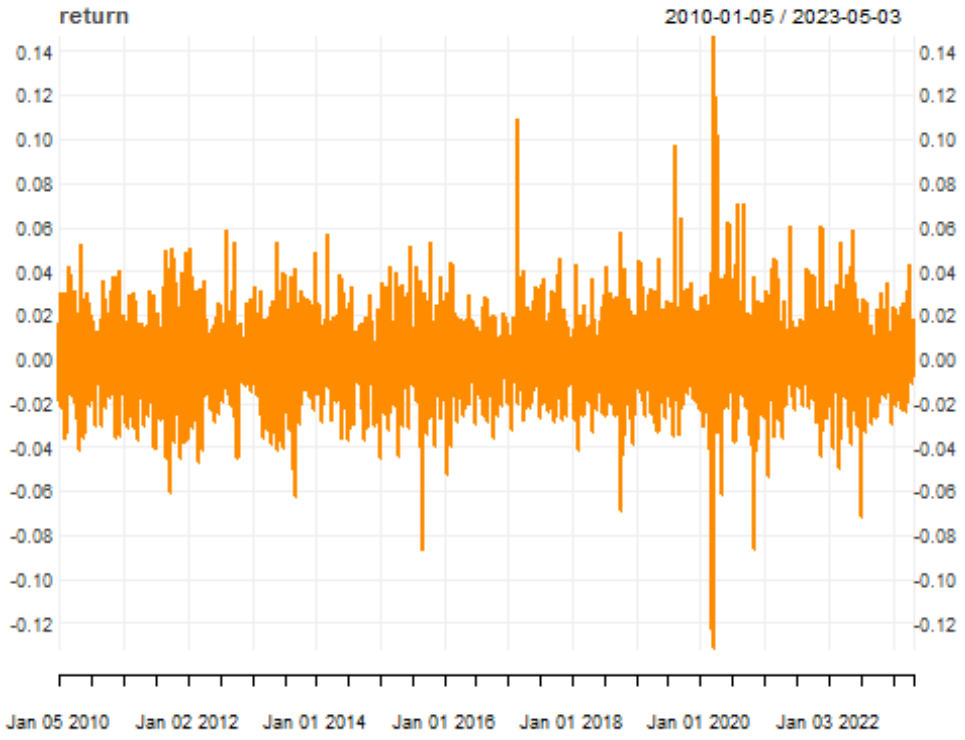



```
head(RELIANCE.NS)

##            RELIANCE.NS.Open RELIANCE.NS.High RELIANCE.NS.Low
RELIANCE.NS.Close
## 2010-01-04         540.4273         540.4273        506.1274
532.7005
## 2010-01-05         569.5512         569.5512        527.6979
530.3231
## 2010-01-06         534.8799         542.1113        530.2983
538.8918
## 2010-01-07         538.8918         552.2651        533.9388
547.8321
## 2010-01-08         548.7980         551.6707        542.8543
546.3957
## 2010-01-11         562.0226         569.4521        520.0703
535.6724
##            RELIANCE.NS.Volume RELIANCE.NS.Adjusted
## 2010-01-04           35372156              482.8124
## 2010-01-05            9872785              480.6577
## 2010-01-06           10933743              488.4240
```

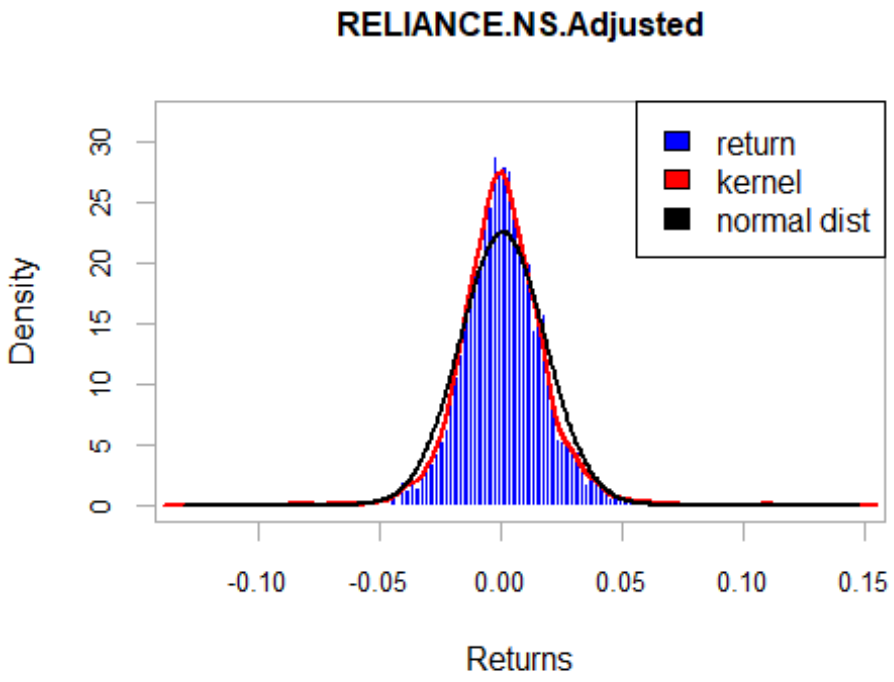
```
## 2010-01-07           12090336              496.5270
## 2010-01-08            6973331              495.2250
## 2010-01-11           87061779              485.5059

tail(RELIANCE.NS)

##            RELIANCE.NS.Open RELIANCE.NS.High RELIANCE.NS.Low
RELIANCE.NS.Close
## 2023-04-25           2366.0           2380.6         2350.50
2376.05
## 2023-04-26           2379.0           2386.1         2354.05
2362.10
## 2023-04-27           2375.0           2384.0         2364.00
2377.05
## 2023-04-28           2382.0           2423.9         2381.75
2420.50
## 2023-05-02           2436.2           2445.8         2428.10
2441.05
## 2023-05-03           2445.0           2445.0         2413.05
2420.10
##            RELIANCE.NS.Volume RELIANCE.NS.Adjusted
## 2023-04-25            4262471              2376.05
## 2023-04-26            3977129              2362.10
## 2023-04-27            4230627              2377.05
## 2023-04-28            7183342              2420.50
## 2023-05-02            5991101              2441.05
## 2023-05-03            3259053              2420.10

chartSeries(RELIANCE.NS["2023-04"])
```




```
#These functions are used to obtain the Reliance Stock Price dataset using
getSymbols and print a chart of the dataset in the form of a time series
plot.

return = CalculateReturns(RELIANCE.NS$RELIANCE.NS.Adjusted)
return = return[-c(1),]
head(return)

##            RELIANCE.NS.Adjusted
## 2010-01-05         -0.004462860
## 2010-01-06          0.016157582
## 2010-01-07          0.016590152
## 2010-01-08         -0.002622218
## 2010-01-11         -0.019625535
## 2010-01-12          0.002635291

#This Calculate function is used to calculate the returns from stock price
which can be further used to check volatality and fit a GARCH model.

chart_Series(return)
```




```
chart.Histogram(return,methods = c("add.density","add.normal"),
                colorset = c("blue","red","black"))
legend("topright",legend=c("return","kernel","normal dist"),
       fill = c('blue','red','black'))
```



Inference: - - Here,we first create a chart showing the returns of the stock price time series and then create a histogram showcasing kernel,normality and other specifications using the appropriate colours.

```
ndiffs(return)
```

```
## [1] 0
```

```
#nsdiffs(return)
```

Inference: - - ndiffs is used to check if any difference is required in the data to avoid trend.
- As the data is non seasonal data,we need not use the nsdiffs function.

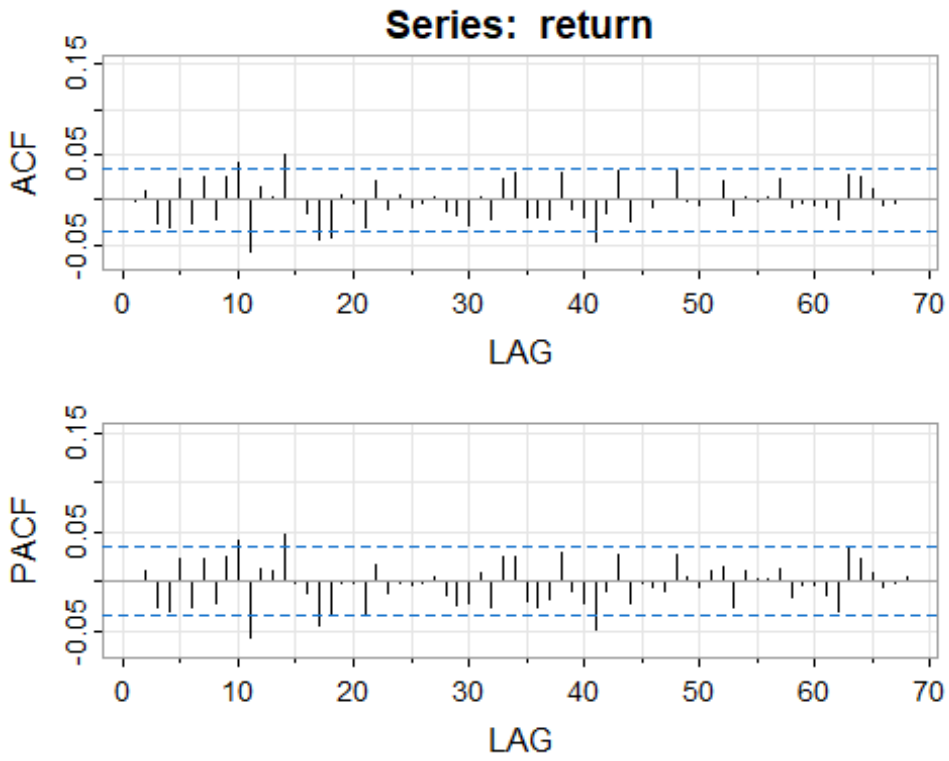
```
library(astsa)
```

```
## Warning: package 'astsa' was built under R version 4.2.2
```

```
##
## Attaching package: 'astsa'
```

```
## The following object is masked from 'package:forecast':
##
##     gas
```

```
acf2(return)
```

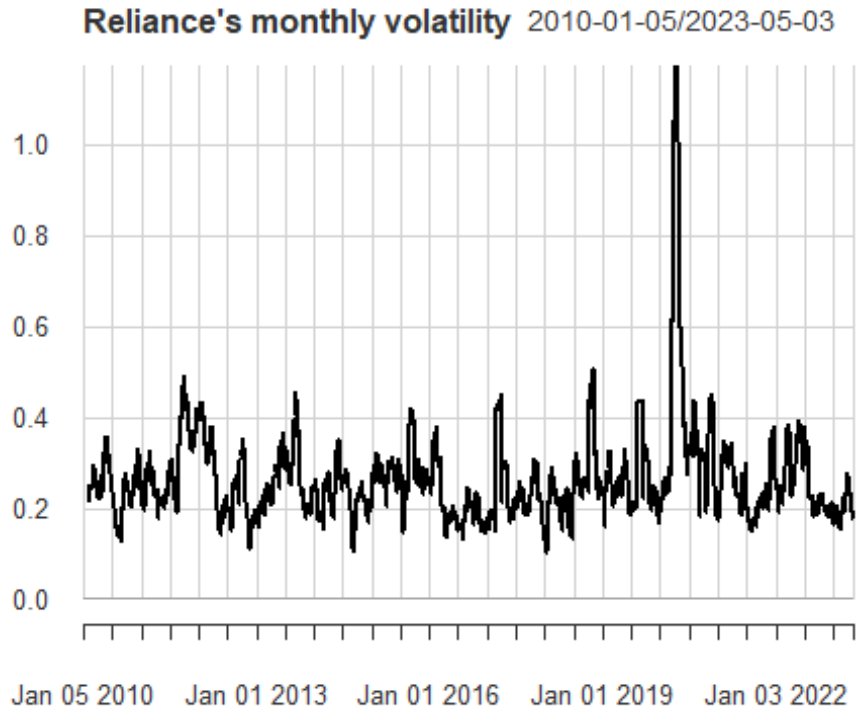



```
##      [,1] [,2]  [,3]  [,4] [,5]  [,6] [,7]  [,8] [,9] [,10] [,11] [,12]
[,13]
## ACF     0 0.01 -0.03 -0.03 0.02 -0.03 0.03 -0.02 0.03  0.04 -0.06  0.02
0.00
## PACF    0 0.01 -0.03 -0.03 0.02 -0.03 0.02 -0.02 0.03  0.04 -0.06  0.01
0.01
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[,25]
## ACF   0.05     0 -0.01 -0.04 -0.04     0     0 -0.03  0.02 -0.01     0 -
0.01
## PACF  0.05     0 -0.01 -0.05 -0.03     0     0 -0.03  0.02 -0.01     0
0.00
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
[,37]
## ACF      0     0 -0.01 -0.02 -0.03  0.00 -0.02  0.02  0.03 -0.02 -0.02 -
0.02
## PACF     0     0 -0.01 -0.02 -0.02  0.01 -0.03  0.02  0.03 -0.02 -0.03 -
0.02
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
[,49]
## ACF   0.03 -0.01 -0.02 -0.05 -0.01  0.03 -0.02     0 -0.01  0.00  0.03
0
## PACF  0.03 -0.01 -0.02 -0.05 -0.01  0.03 -0.02     0 -0.01 -0.01  0.03
0
##      [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60]
[,61]
## ACF  -0.01  0.00  0.02 -0.02  0.00     0     0  0.02 -0.01     0 -0.01 -
```

```
0.01
## PACF -0.01  0.01  0.01 -0.03  0.01     0     0  0.01 -0.02     0  0.00 -
0.01
##      [,62] [,63] [,64] [,65] [,66] [,67] [,68]
## ACF  -0.02  0.03  0.03  0.01 -0.01     0     0
## PACF -0.03  0.04  0.02  0.01 -0.01     0     0

library(forecast)
auto.arima(return)

## Series: return
## ARIMA(0,0,3) with non-zero mean
##
## Coefficients:
##           ma1     ma2      ma3   mean
##       -0.0016  0.0123  -0.0275  6e-04
## s.e.   0.0175  0.0180   0.0179  3e-04
##
## sigma^2 = 0.0003147:  log likelihood = 8577.68
## AIC=-17145.35   AICc=-17145.33   BIC=-17114.85

return <- na.omit(return)
adf.test(return)

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

##
##  Augmented Dickey-Fuller Test
##
## data:  return
## Dickey-Fuller = -14.248, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(return)

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

##
##  KPSS Test for Level Stationarity
##
## data:  return
## KPSS Level = 0.24379, Truncation lag parameter = 9, p-value = 0.1
```

Inference: - - Acf2 returns ACF and PACF plot which we used to find if any AR,MA components exist in the data. Here we weren't able to exactly find order of AR,MA components. - na.omit() function is used to omit NA records in the return data which might be due to market holidays. - The adf.test performs the Augmented Dickey-Fuller test for the null hypothesis of a unit root of a univarate time series and suggests that the data is stationary here. - The kpss.test computes the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for the null hypothesis that x is level or trend stationary. Kpss test also suggests that the data is stationary.

```
chart.RollingPerformance(R = return["2010::2023"], width = 22,
                         FUN = "sd.annualized",scale = 252,
                         main = "Reliance's monthly volatility")
```



A wrapper to create a chart of rolling performance metrics in a line chart and which shows Reliance's monthly volatility using returns data.

We are now using the GJRGARCH(1,0) model to specify,fit the model and predict the future stock prices.

#We go for GJRGARCH model to capture the news impact better in the model

```
armaOrder <- c(0,0)

# We are defining the ARMA order to be 0,0 as we can't find evident AR, MA
components from the ACF, PACF plot

garchOrder <- c(1,0)

# We are defining the GARCH order to be 1,0

varModel <- list(model = "gjrGARCH", garchOrder = garchOrder)

mod_specify <- ugarchspec(varModel, mean.model = list(armaOrder =
armaOrder),distribution.model = "sstd")
```

```r
#Here we are specifying the model distribution as sstd which is used to
compute density, distribution function, quantile function and to generate
random variates for the skew Student-t distribution.

mod_fitting = ugarchfit(data = return,spec = mod_specify,out.sample = 20)
mod_fitting
```

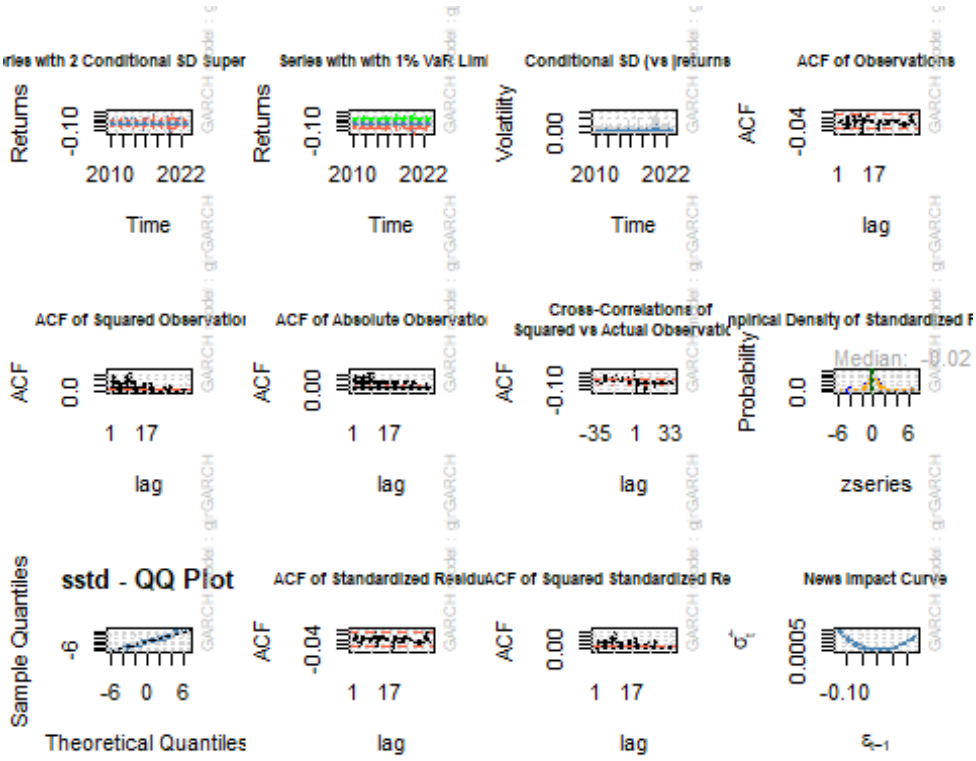
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,0)
## Mean Model   : ARFIMA(0,0,0)
## Distribution : sstd
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000524    0.000296  1.76938 0.076831
## omega   0.000263    0.000012 21.66398 0.000000
## alpha1  0.111192    0.034383  3.23390 0.001221
## gamma1  0.049957    0.051165  0.97639 0.328869
## skew    1.052111    0.026008 40.45319 0.000000
## shape   5.599285    0.521314 10.74070 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000524    0.000295  1.77469 0.075950
## omega   0.000263    0.000015 17.05078 0.000000
## alpha1  0.111192    0.052272  2.12717 0.033406
## gamma1  0.049957    0.054432  0.91779 0.358729
## skew    1.052111    0.028251 37.24155 0.000000
## shape   5.599285    0.727717  7.69432 0.000000
##
## LogLikelihood : 8736.615
##
## Information Criteria
## ------------------------------------
##
## Akaike       -5.3529
## Bayes        -5.3417
## Shibata      -5.3529
## Hannan-Quinn -5.3489
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
```

```
## Lag[1]                      2.742 0.09773
## Lag[2*(p+q)+(p+q)-1][2]     2.937 0.14592
## Lag[4*(p+q)+(p+q)-1][5]     6.706 0.06109
## d.o.f=0
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic   p-value
## Lag[1]                     0.2993 5.843e-01
## Lag[2*(p+q)+(p+q)-1][2]   19.5065 5.998e-06
## Lag[4*(p+q)+(p+q)-1][5]   61.5077 0.000e+00
## d.o.f=1
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale   P-Value
## ARCH Lag[2]     38.37 0.500 2.000 5.861e-10
## ARCH Lag[4]     75.66 1.397 1.611 0.000e+00
## ARCH Lag[6]     93.17 2.222 1.500 0.000e+00
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.2932
## Individual Statistics:
## mu     0.35265
## omega  0.45486
## alpha1 0.27377
## gamma1 0.39682
## skew   0.07952
## shape  0.96366
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.49 1.68 2.12
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           0.5799 0.56204
## Negative Sign Bias  0.3956 0.69243
## Positive Sign Bias  1.6482 0.09941   *
## Joint Effect        2.8796 0.41056
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     9.723       0.9595
## 2    30    23.543       0.7511
```

```
## 3     40    27.491       0.9165
## 4     50    33.892       0.9505
##
##
## Elapsed time : 2.680126
```

```
plot(mod_fitting, which = 'all')
```

```
##
## please wait...calculating quantiles...
```




```
?ugarchfit
```

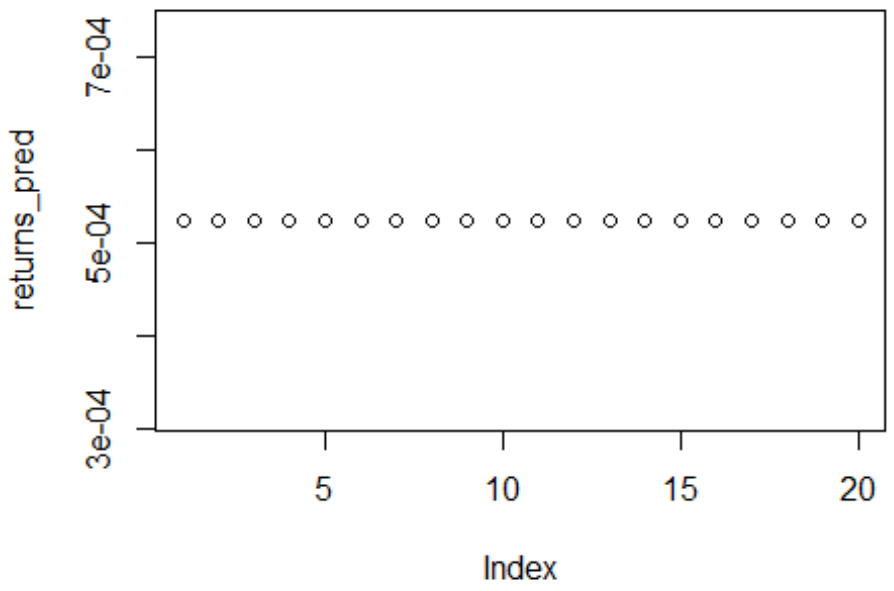
```
## starting httpd help server ... done
```

```
#Ugarchfit is a method for fitting a variety of univariate GARCH models.
```

Predict future returns using ugarchforecast which provides methods for forecasting from a variety of univariate GARCH models.n.ahead represents the forecast horizon.

We also plot the forecasts to obtain a visual of the predicted forecasts.

```
returns_pred <- ugarchforecast(mod_fitting, n.ahead = 20)
returns_pred <- as.numeric(returns_pred@forecast$seriesFor[1:20])
plot(returns_pred)
```



Calculate the predicted stock price based on the forecasted returns using the tail function.

```
last_price <- tail(RELIANCE.NS$RELIANCE.NS.Adjusted, 1)
last_price
```

```
##            RELIANCE.NS.Adjusted
## 2023-05-03               2420.1
```

With the help of predicted returns we are calculating predicted prices for next 20 days

```
predicted_prices <- c()
for (i in seq_along(returns_pred)) {
  predicted_price <- last_price * (1 + sum(returns_pred[1:i]))
  predicted_prices <- c(predicted_prices, predicted_price)
}

predicted_prices
```

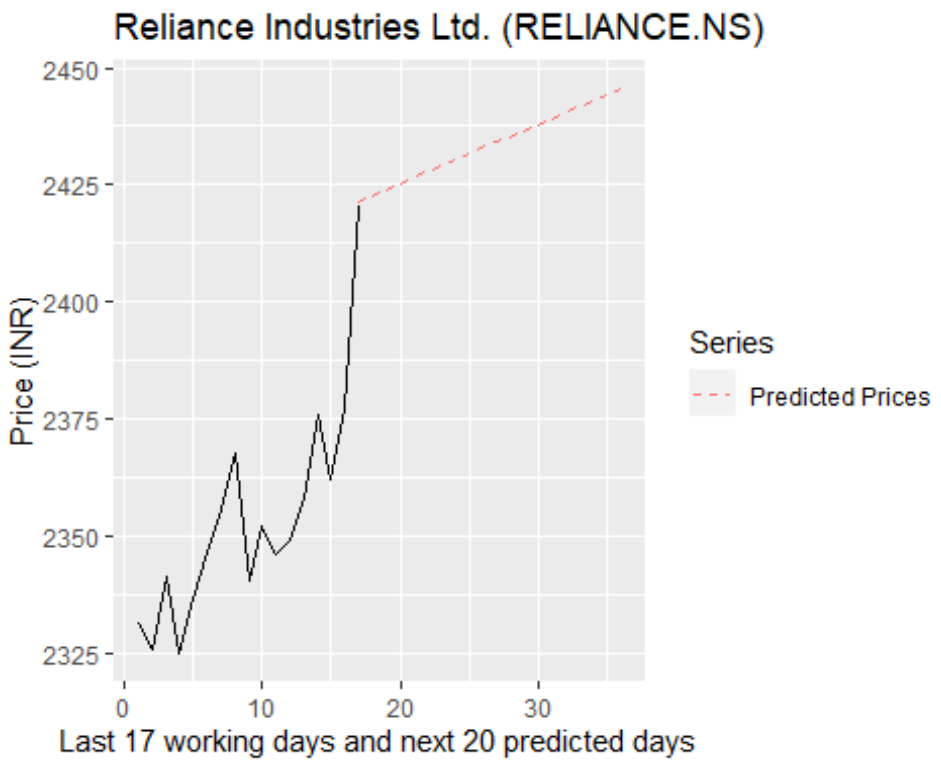
```
##  [1] 2421.369 2422.637 2423.906 2425.175 2426.443 2427.712 2428.980
2430.249
##  [9] 2431.518 2432.786 2434.055 2435.323 2436.592 2437.861 2439.129
2440.398
## [17] 2441.667 2442.935 2444.204 2445.472
```

Converting the predicted prices data to a time series data in order to plot with the past data

```
ts_pred <- ts(predicted_prices, start = 17, frequency = 1)
```

Plot the original and predicted prices

```
autoplot(ts(RELIANCE.NS$RELIANCE.NS.Adjusted["2023-04"])) +
  autolayer((ts_pred), series = "Predicted Prices", linetype = "dashed") +
  ggtitle("Reliance Industries Ltd. (RELIANCE.NS)") +
  ylab("Price (INR)") +
  xlab("Last 17 working days and next 20 predicted days") +
  guides(colour = guide_legend(title = "Series"))
```



Now we are going to measure the accuracy of the model, by splitting into train and test data.

```
n = 20
armaOrder2 <- c(0,0) # ARMA order
garchOrder2 <- c(1,0) # GARCH order
varModel2 <- list(model = "gjrGARCH", garchOrder = garchOrder)
mod_specify2 <- ugarchspec(varModel, mean.model = list(armaOrder =
armaOrder), distribution.model = "sstd")
l2 <- length((return$RELIANCE.NS.Adjusted))- n
train<-head((return$RELIANCE.NS.Adjusted), l2)
test<-tail((RELIANCE.NS$RELIANCE.NS.Adjusted),n)
mod_fitting2 = ugarchfit(data = train,spec = mod_specify,out.sample = n)
returns_pred2 <- ugarchforecast(mod_fitting2, n.ahead = nrow(test))
returns_pred2

##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: gjrGARCH
```

```
## Horizon: 20
## Roll Steps: 0
## Out of Sample: 20
##
## 0-roll forecast [T0=2023-02-28]:
##         Series   Sigma
## T+1  0.0005377 0.01805
## T+2  0.0005377 0.01756
## T+3  0.0005377 0.01749
## T+4  0.0005377 0.01748
## T+5  0.0005377 0.01748
## T+6  0.0005377 0.01748
## T+7  0.0005377 0.01748
## T+8  0.0005377 0.01748
## T+9  0.0005377 0.01748
## T+10 0.0005377 0.01748
## T+11 0.0005377 0.01748
## T+12 0.0005377 0.01748
## T+13 0.0005377 0.01748
## T+14 0.0005377 0.01748
## T+15 0.0005377 0.01748
## T+16 0.0005377 0.01748
## T+17 0.0005377 0.01748
## T+18 0.0005377 0.01748
## T+19 0.0005377 0.01748
## T+20 0.0005377 0.01748
```

```
returns_pred2 <- as.numeric(returns_pred2@forecast$seriesFor[1:20])
train_price<-head((RELIANCE.NS$RELIANCE.NS.Adjusted), l2)
last<-tail(train_price,1)
predicted_prices2 <- c()
predicted_price2<-c()
for (i in seq_along(returns_pred2)) {
  predicted_price2 <- last_price * (1 + sum(returns_pred2[1:i]))
  predicted_prices2 <- c(predicted_prices2, predicted_price2)
}
predicted_prices2
```

```
##  [1] 2421.401 2422.703 2424.004 2425.305 2426.607 2427.908 2429.209
2430.510
##  [9] 2431.812 2433.113 2434.414 2435.716 2437.017 2438.318 2439.619
2440.921
## [17] 2442.222 2443.523 2444.825 2446.126
```

Finding the accuracy of the model by calculating MAPE value using test and predicted price data

```
mape <- function(actual, predicted) {
  mean(abs((actual - predicted)/actual))*100
}
mape(test, predicted_prices2)
```

```
## [1] 3.134325
```

5. Conclusion: -

- Thus,from the above methods and results,we have tried obtaining the best model which can be used to analyse and forecast the stock prices,and we have also checked the accuracy of our model.
- Through this project,we learnt about the various Time series models which can be used to fit and forecast and also about various R packages used for creating a time series model and performing statistical and analytical functions and also effectively deciphering the information from the plots to obtain the required results.
- This project has given us an idea on how to deal with complex time series data and models and thereby fit better models.

6. Critique: -

- The presence of temporal dependencies, which might influence the chosen statistical models and methodologies, is one of the major issues in time series analysis.
- Furthermore, if time series data are not handled properly, external influences like seasonality, trends, and outliers may skew or produce inaccurate results.
- The necessity to analyse the model's performance using relevant metrics, such as mean absolute percetnage error and the choice of an acceptable evaluation window to judge the model's propensity to anticipate future values are both crucial factors in time series projects.
- The effectiveness of the model and the precision of the predictions must also be taken into account when considering the effects of data preparation, such as missing data imputation, normalisation, and smoothing.
- The choice of models and procedures, data pretreatment, and proper assessment metrics are only a few of the many complicated and difficult issues that must be carefully taken into account while performing time series analysis.
- Though we have explored different orders and models to come with this model, we understand that MAPE value can be further decreased by performing further deep analysis.