Shri Vaishnav Vidyapeeth Vishwavidyalaya, Shri Vaishnav Institute of Information Technology Department of Computer Science & Engineering

Course Name: **- “Data Visualization + Machine Learning with R”**

Course Code**: - BTIBMA401**

II nd – Year / IV th – Semester

Class – CS-AI / Section:-‘D’

PROJECT_FILE

Submitted By: -

Kamakshi Sharma

Tanay shroff

Gautum Jaiswal

Sahil Vaidya

Submitted to:

Ms. Anmol choudhary

STOCK MARKET PRICE PREDICTION.

Libraries used in the project –

library(quantmod): Quantitative Financial Modelling Framework (use to load a variety of data from different sources).

library(tseries): tseries stands for Time Series Analysis and Computational Finance. Imports: graphics, stats, utils, quadprog, zoo, quantmod, and other statistical operations like ADF test, p value test.

library(timeSeries): Basic functions such as scaling and sorting, sub-setting, mathematical operations, and statistical functions.

library(forecast): Provides Methods and tools for displaying and analyzing univariate time series forecasts including exponential smoothing via state space models and automatic ARIMA modelling.

Arima Model: (auto-regressive integrated moving average) ARIMA models provide another approach to time series forecasting. Exponential smoothing and ARIMA models are the two most widely used approaches to time series forecasting and provide complementary approaches to the problem. While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

data come from financialyahoo.com

addbands ()- it gives 3 line 2 red and 1 white

red line gives highest price at particular instance

white gives average value

lower red line gives lowest price at any instant

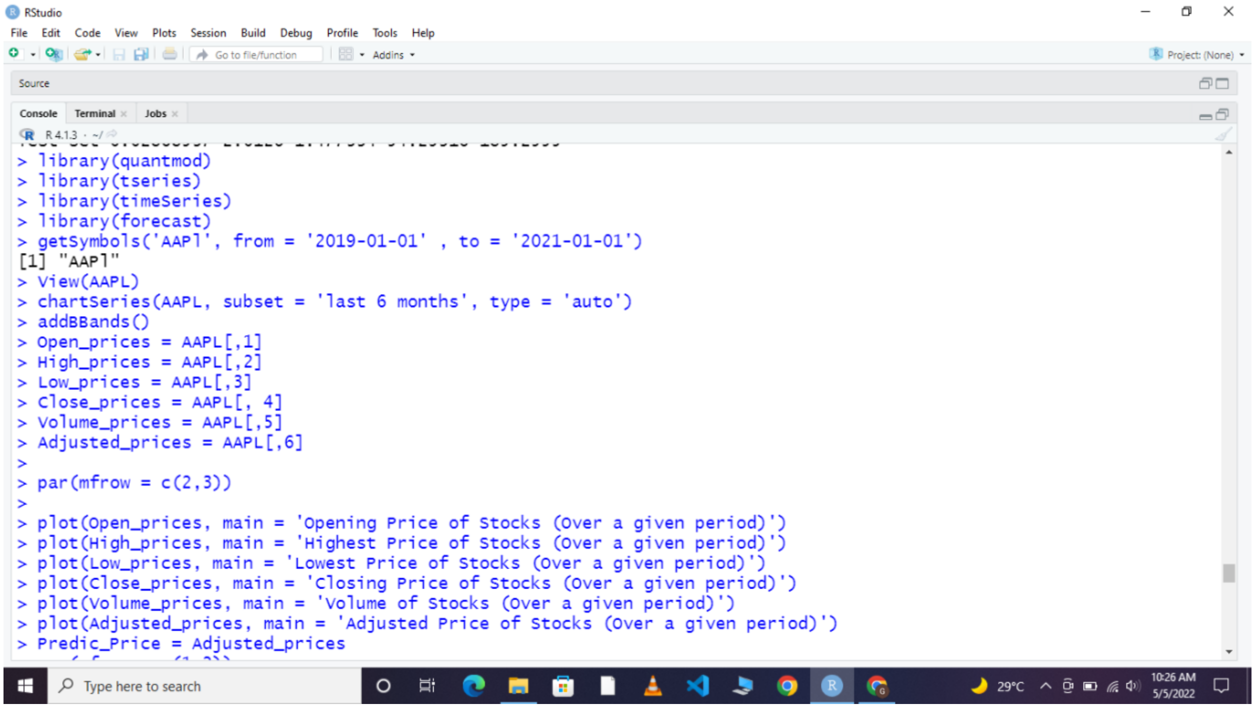
parameter(): - It used to give grid to the graph

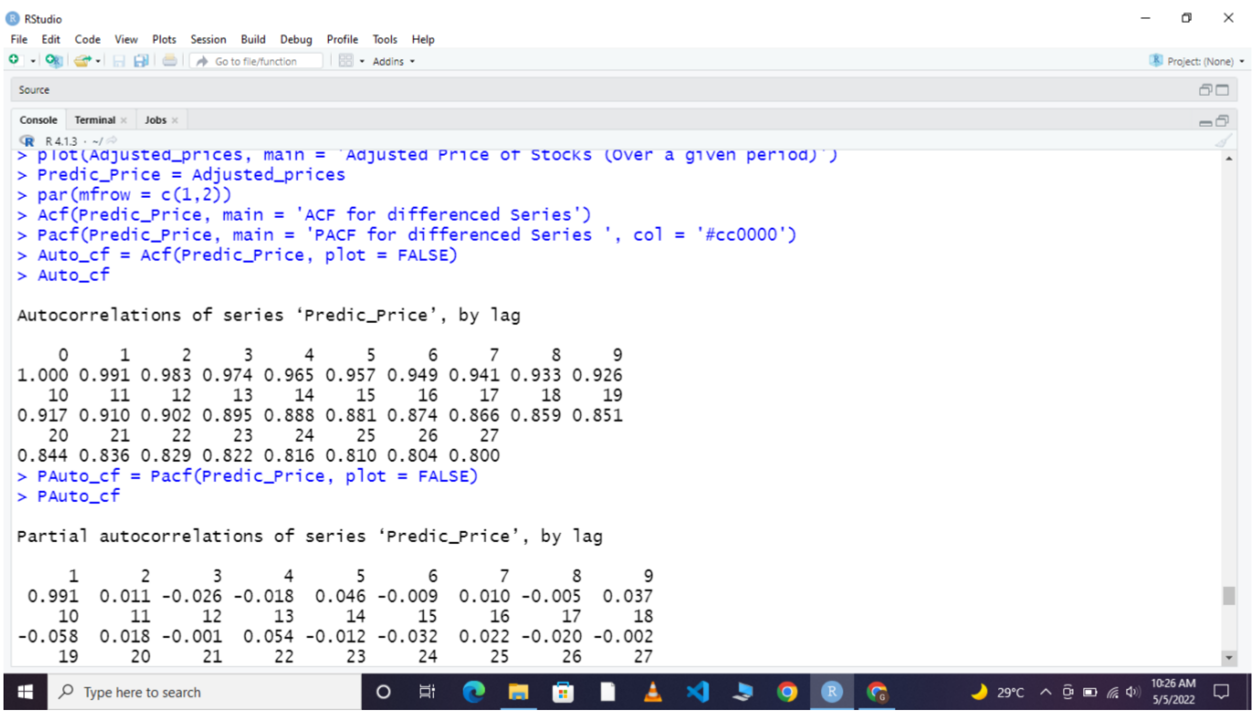
CODE[(c(row,column)].

```
library(quantmod)
library(tseries)
library(timeSeries)
library(forecast)
getSymbols('AAPl', from = '2019-01-01' , to = '2021-01-01')
View(AAPL)
chartSeries(AAPL, subset = 'last 6 months', type = 'auto')
addBBands()
Open_prices = AAPL[,1]
High_prices = AAPL[,2]
Low_prices = AAPL[,3]
Close_prices = AAPL[, 4]
Volume_prices = AAPL[,5]
Adjusted_prices = AAPL[,6]

par(mfrow = c(2,3))

plot(Open_prices, main = 'Opening Price of Stocks (Over a given period)')
plot(High_prices, main = 'Highest Price of Stocks (Over a given period)')
plot(Low_prices, main = 'Lowest Price of Stocks (Over a given period)')
plot(Close_prices, main = 'Closing Price of Stocks (Over a given period)')
plot(Volume_prices, main = 'Volume of Stocks (Over a given period)')
plot(Adjusted_prices, main = 'Adjusted Price of Stocks (Over a given period)')

Predic_Price = Adjusted_prices
#class(Predic_Price)


######## Finding the Linear Relation between observations ########

par(mfrow = c(1,2))
Acf(Predic_Price, main = 'ACF for differenced Series')
Pacf(Predic_Price, main = 'PACF for differenced Series ', col = '#cc0000')
Auto_cf = Acf(Predic_Price, plot = FALSE)
Auto_cf
PAuto_cf = Pacf(Predic_Price, plot = FALSE)
PAuto_cf

print(adf.test(Predic_Price))
```

```
################### Prediction of Return ##########################

return_AAPL<- 100*diff(log(Predic_Price))

AAPL_return_train<- return_AAPL[1:(0.9*length(return_AAPL))]

AAPL_return_test<- return_AAPL[(0.9*length(return_AAPL)+1):length(return_AAPL)]

auto.arima(AAPL_return_train, seasonal = FALSE)

fit<- Arima(AAPL_return_train, order = c(1,0,0))

preds<- predict(fit, n.ahead = (length(return_AAPL) -
(0.9*length(return_AAPL))))$pred
preds


################## Forecasting Predicted Result ##################

test_forecast<- forecast(fit,h = 15)
test_forecast

par(mfrow = c(1,1))
plot(test_forecast, main = "Arima forecast for Apple Stock")

accuracy(preds, AAPL_return_test)
```

RESULT


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
R 4.1.3 · ~/
> library(quantmod)
> library(tseries)
> library(timeSeries)
> library(forecast)
> getSymbols('AAPl', from = '2019-01-01' , to = '2021-01-01')
[1] "AAPl"
> View(AAPL)
> chartSeries(AAPL, subset = 'last 6 months', type = 'auto')
> addBBands()
> Open_prices = AAPL[,1]
> High_prices = AAPL[,2]
> Low_prices = AAPL[,3]
> Close_prices = AAPL[, 4]
> Volume_prices = AAPL[,5]
> Adjusted_prices = AAPL[,6]
>
> par(mfrow = c(2,3))
>
> plot(Open_prices, main = 'Opening Price of Stocks (Over a given period)')
> plot(High_prices, main = 'Highest Price of Stocks (Over a given period)')
> plot(Low_prices, main = 'Lowest Price of Stocks (Over a given period)')
> plot(Close_prices, main = 'Closing Price of Stocks (Over a given period)')
> plot(Volume_prices, main = 'Volume of Stocks (Over a given period)')
> plot(Adjusted_prices, main = 'Adjusted Price of Stocks (Over a given period)')
> Predic_Price = Adjusted_prices
Type here to search
29°C
10:26 AM
5/5/2022

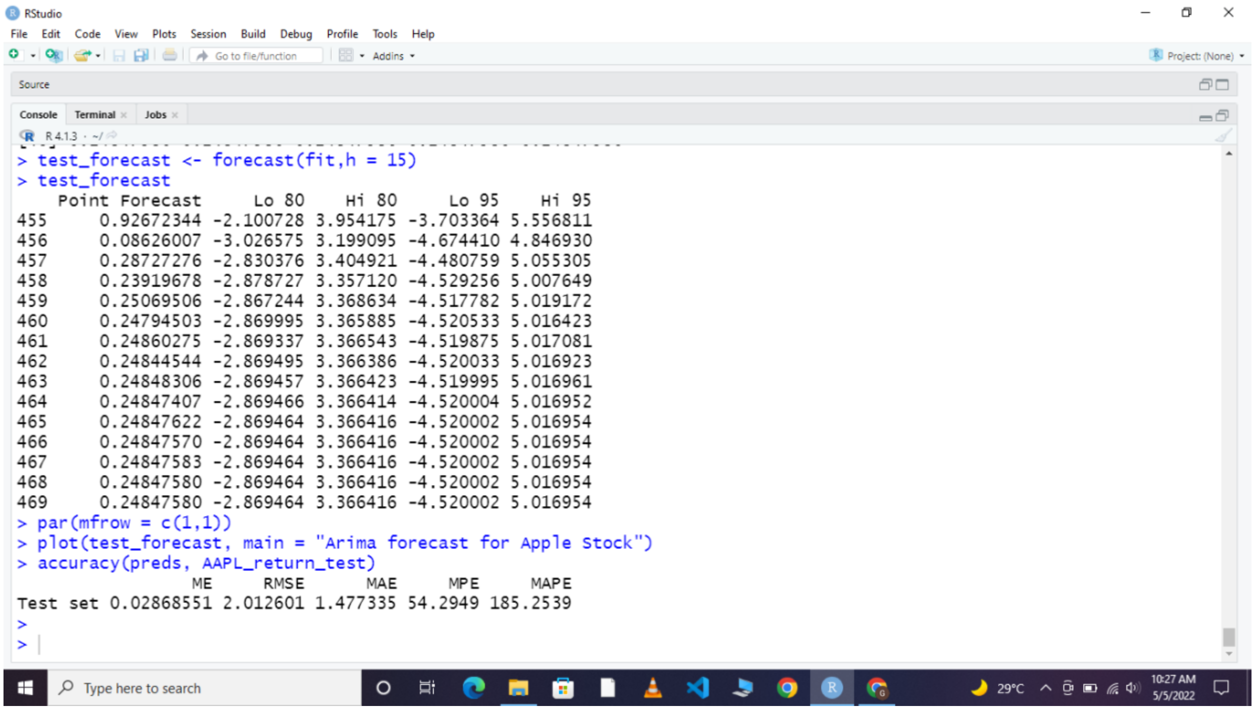

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Source
Console Terminal Jobs
R 4.1.3 · ~/
> plot(Adjusted_prices, main = 'Adjusted Price of Stocks (Over a given period)')
> Predic_Price = Adjusted_prices
> par(mfrow = c(1,2))
> Acf(Predic_Price, main = 'ACF for differenced Series')
> Pacf(Predic_Price, main = 'PACF for differenced Series ', col = '#cc0000')
> Auto_cf = Acf(Predic_Price, plot = FALSE)
> Auto_cf
Autocorrelations of series 'Predic_Price', by lag
0 1 2 3 4 5 6 7 8 9
1.000 0.991 0.983 0.974 0.965 0.957 0.949 0.941 0.933 0.926
10 11 12 13 14 15 16 17 18 19
0.917 0.910 0.902 0.895 0.888 0.881 0.874 0.866 0.859 0.851
20 21 22 23 24 25 26 27
0.844 0.836 0.829 0.822 0.816 0.810 0.804 0.800
> PAuto_cf = Pacf(Predic_Price, plot = FALSE)
> PAuto_cf
Partial autocorrelations of series 'Predic_Price', by lag
1 2 3 4 5 6 7 8 9
0.991 0.011 -0.026 -0.018 0.046 -0.009 0.010 -0.005 0.037
10 11 12 13 14 15 16 17 18
-0.058 0.018 -0.001 0.054 -0.012 -0.032 0.022 -0.020 -0.002
19 20 21 22 23 24 25 26 27
Type here to search
29°C
10:26 AM
5/5/2022

```
0.844 0.836 0.829 0.822 0.816 0.810 0.804 0.800
> PAuto_cf = Pacf(Predic_Price, plot = FALSE)
> PAuto_cf

Partial autocorrelations of series ‘Predic_Price’, by lag

     1      2      3      4      5      6      7      8      9 
 0.991  0.011 -0.026 -0.018  0.046 -0.009  0.010 -0.005  0.037 
    10     11     12     13     14     15     16     17     18 
-0.058  0.018 -0.001  0.054 -0.012 -0.032  0.022 -0.020 -0.002 
    19     20     21     22     23     24     25     26     27 
-0.031  0.006  0.011 -0.022  0.054  0.021  0.035 -0.016  0.051 
> print(adf.test(Predic_Price))

	Augmented Dickey-Fuller Test

data:  Predic_Price
Dickey-Fuller = -1.8237, Lag order = 7, p-value = 0.6529
alternative hypothesis: stationary

> return_AAPL <- 100*diff(log(Predic_Price))
> AAPL_return_train <- return_AAPL[1:(0.9*length(return_AAPL))]
> AAPL_return_test <- return_AAPL[(0.9*length(return_AAPL)+1):length(return_AAPL)]
> auto.arima(AAPL_return_train, seasonal = FALSE)
Series: AAPL_return_train 
ARIMA(1,0,0) with non-zero mean
```

```
> return_AAPL <- 100*diff(log(Predic_Price))
> AAPL_return_train <- return_AAPL[1:(0.9*length(return_AAPL))]
> AAPL_return_test <- return_AAPL[(0.9*length(return_AAPL)+1):length(return_AAPL)]
> auto.arima(AAPL_return_train, seasonal = FALSE)
Series: AAPL_return_train 
ARIMA(1,0,0) with non-zero mean 

Coefficients:
          ar1    mean
      -0.2392  0.2485
s.e.   0.0467  0.0894

sigma^2 = 5.581:  log likelihood = -1031.23
AIC=2068.46   AICc=2068.51   BIC=2080.8
> fit <- Arima(AAPL_return_train, order = c(1,0,0))
> preds <- predict(fit, n.ahead = (length(return_AAPL) - (0.9*length(return_AAPL))))$pred
> preds
Time Series:
Start = 455 
End = 504 
Frequency = 1 
 [1] 0.92672344 0.08626007 0.28727276 0.23919678 0.25069506
 [6] 0.24794503 0.24860275 0.24844544 0.24848306 0.24847407
[11] 0.24847622 0.24847570 0.24847583 0.24847580 0.24847580
[16] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[21] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
```

```
AIC=2068.46   AICc=2068.51   BIC=2080.8
> fit <- Arima(AAPL_return_train, order = c(1,0,0))
> preds <- predict(fit, n.ahead = (length(return_AAPL) - (0.9*length(return_AAPL))))$pred
> preds
Time Series:
Start = 455
End = 504
Frequency = 1
 [1] 0.92672344 0.08626007 0.28727276 0.23919678 0.25069506
 [6] 0.24794503 0.24860275 0.24844544 0.24848306 0.24847407
[11] 0.24847622 0.24847570 0.24847583 0.24847580 0.24847580
[16] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[21] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[26] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[31] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[36] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[41] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
[46] 0.24847580 0.24847580 0.24847580 0.24847580 0.24847580
> test_forecast <- forecast(fit,h = 15)
> test_forecast
    Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
455     0.92672344 -2.100728 3.954175 -3.703364 5.556811
456     0.08626007 -3.026575 3.199095 -4.674410 4.846930
457     0.28727276 -2.830376 3.404921 -4.480759 5.055305
458     0.23919678 -2.878727 3.357120 -4.529256 5.007649
```

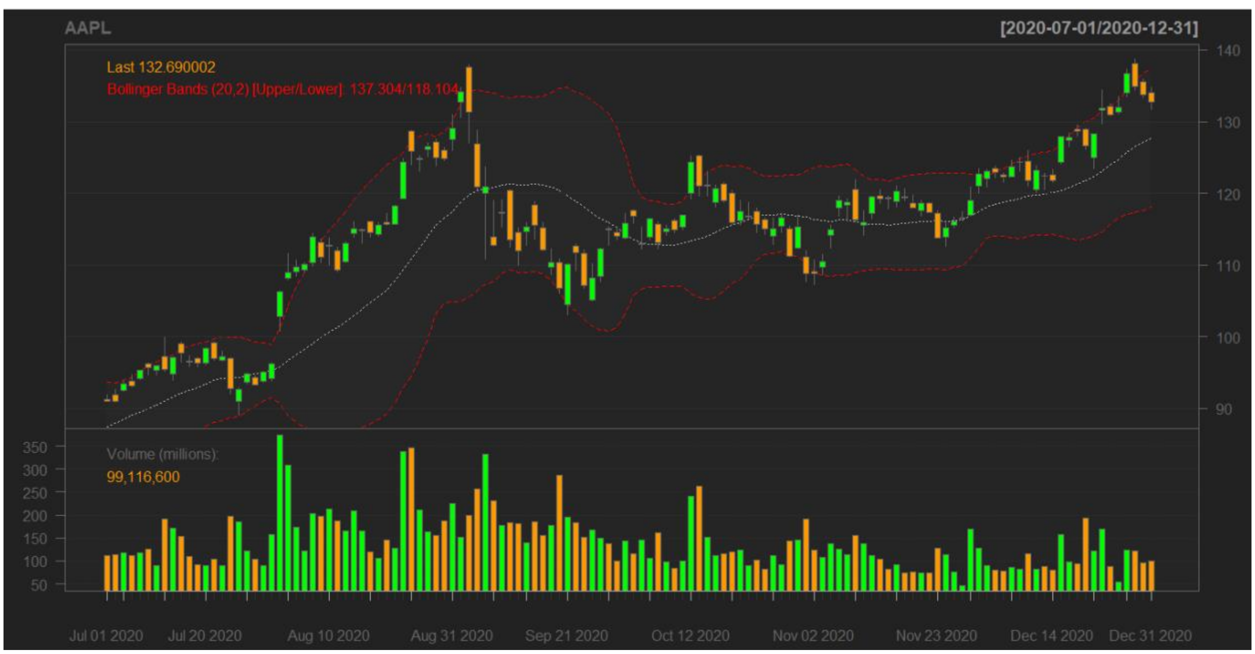
```
> test_forecast <- forecast(fit,h = 15)
> test_forecast
    Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
455     0.92672344 -2.100728 3.954175 -3.703364 5.556811
456     0.08626007 -3.026575 3.199095 -4.674410 4.846930
457     0.28727276 -2.830376 3.404921 -4.480759 5.055305
458     0.23919678 -2.878727 3.357120 -4.529256 5.007649
459     0.25069506 -2.867244 3.368634 -4.517782 5.019172
460     0.24794503 -2.869995 3.365885 -4.520533 5.016423
461     0.24860275 -2.869337 3.366543 -4.519875 5.017081
462     0.24844544 -2.869495 3.366386 -4.520033 5.016923
463     0.24848306 -2.869457 3.366423 -4.519995 5.016961
464     0.24847407 -2.869466 3.366414 -4.520004 5.016952
465     0.24847622 -2.869464 3.366416 -4.520002 5.016954
466     0.24847570 -2.869464 3.366416 -4.520002 5.016954
467     0.24847583 -2.869464 3.366416 -4.520002 5.016954
468     0.24847580 -2.869464 3.366416 -4.520002 5.016954
469     0.24847580 -2.869464 3.366416 -4.520002 5.016954
> par(mfrow = c(1,1))
> plot(test_forecast, main = "Arima forecast for Apple Stock")
> accuracy(preds, AAPL_return_test)
                 ME     RMSE      MAE     MPE     MAPE
Test set 0.02868551 2.012601 1.477335 54.2949 185.2539
>
>
```



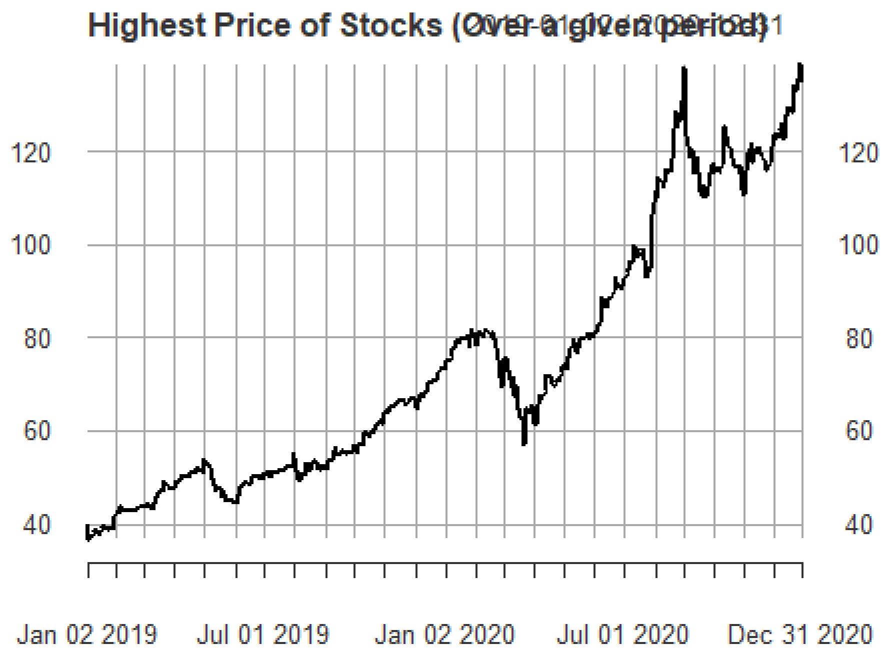
OUTPUT

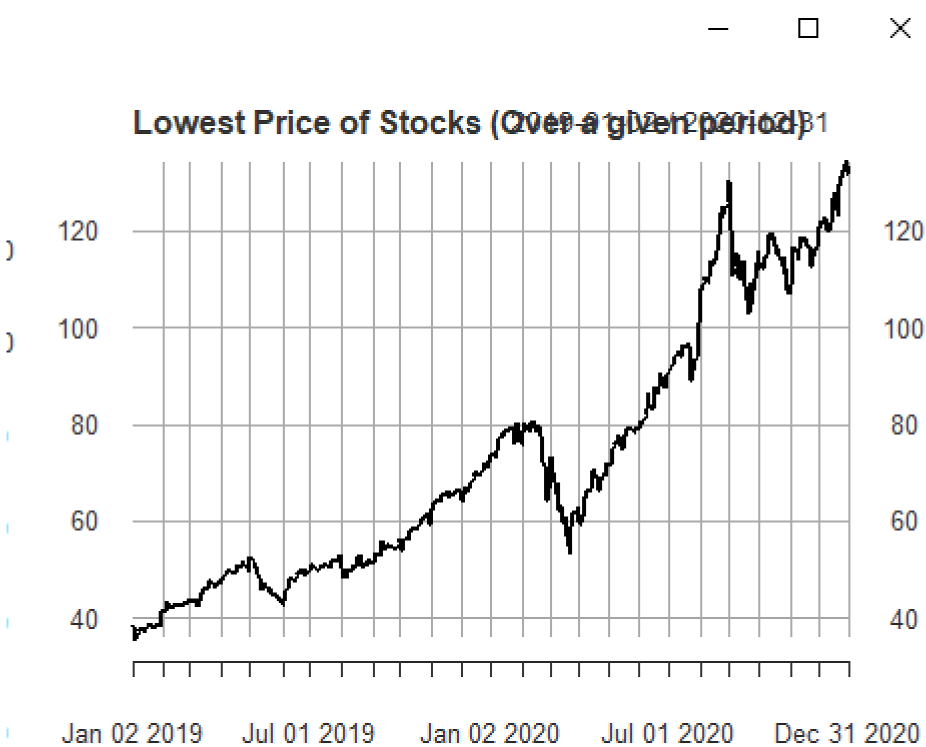
```r
library(quantmod)
library(tseries)
library(timeSeries)
library(forecast)
getSymbols('AAPl', from = '2019-01-01' , to = '2021-01-01')
View(AAPL)
chartSeries(AAPL, subset = 'last 6 months', type = 'auto')
addBBands()
Open_prices = AAPL[,1]
High_prices = AAPL[,2]
Low_prices = AAPL[,3]
Close_prices = AAPL[, 4]
Volume_prices = AAPL[,5]
Adjusted_prices = AAPL[,6]
```

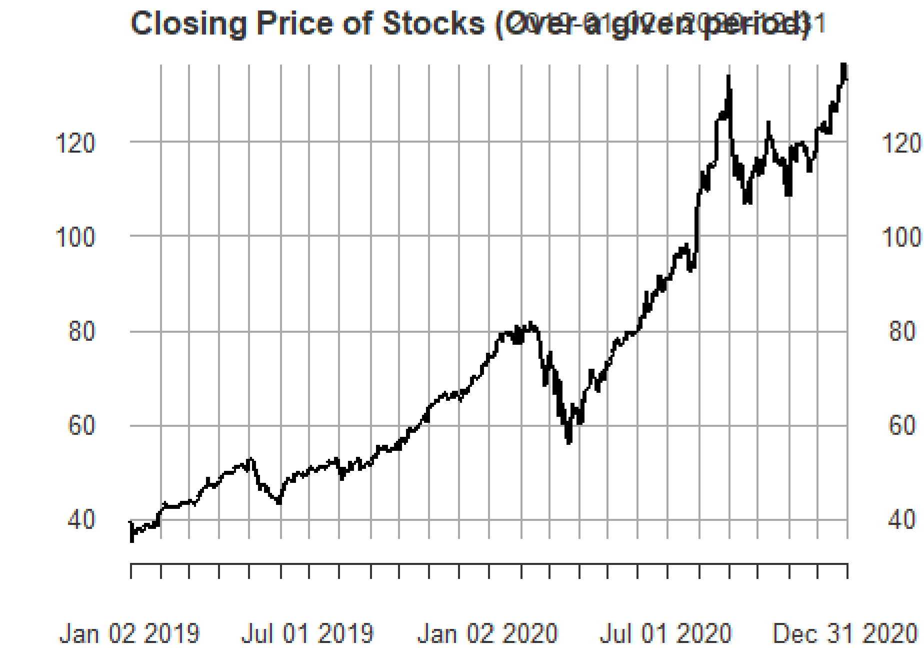
```
par(mfrow = c(2,3))
```

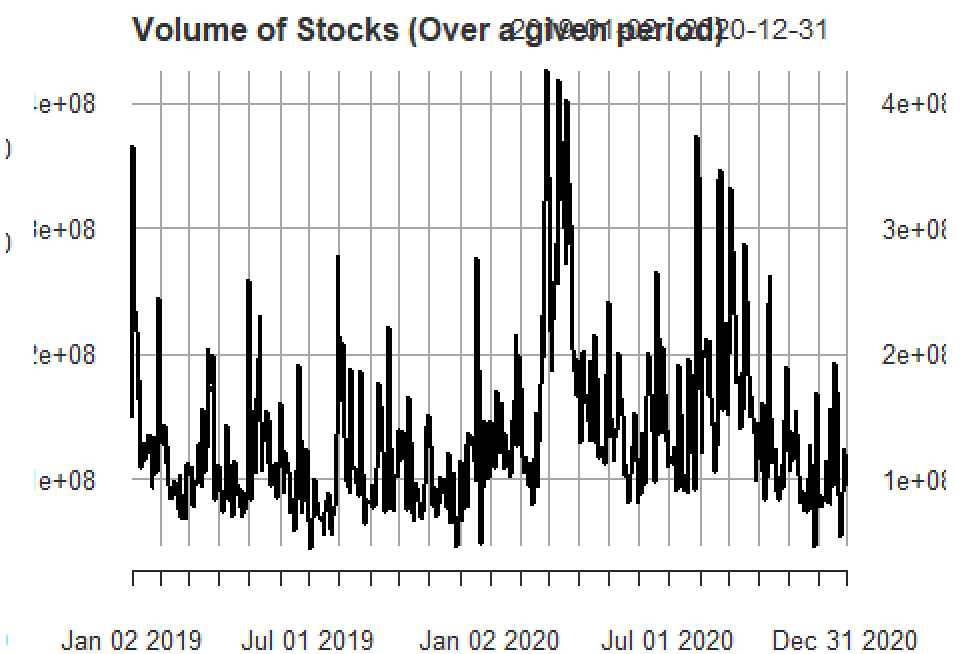


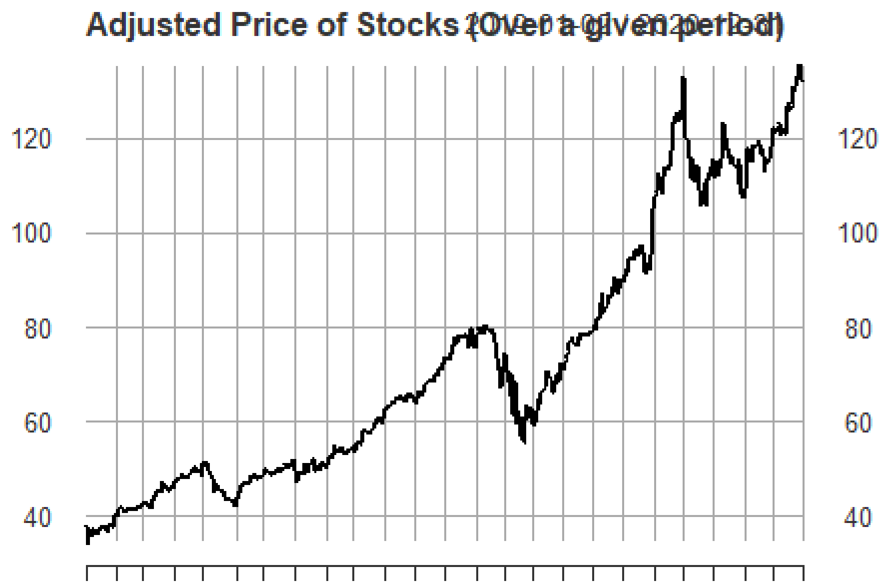
```
plot(Open_prices, main = 'Opening Price of Stocks (Over a given period)')
plot(High_prices, main = 'Highest Price of Stocks (Over a given period)')
plot(Low_prices, main = 'Lowest Price of Stocks (Over a given period)')
plot(Close_prices, main = 'Closing Price of Stocks (Over a given period)')
plot(Volume_prices, main = 'Volume of Stocks (Over a given period)')
plot(Adjusted_prices, main = 'Adjusted Price of Stocks (Over a given period)')
```


Highest Price of Stocks (Over a given period)
2019-01-02 / 2020-12-31
120
100
80
60
40
Jan 02 2019
Jul 01 2019
Jan 02 2020
Jul 01 2020
Dec 31 2020

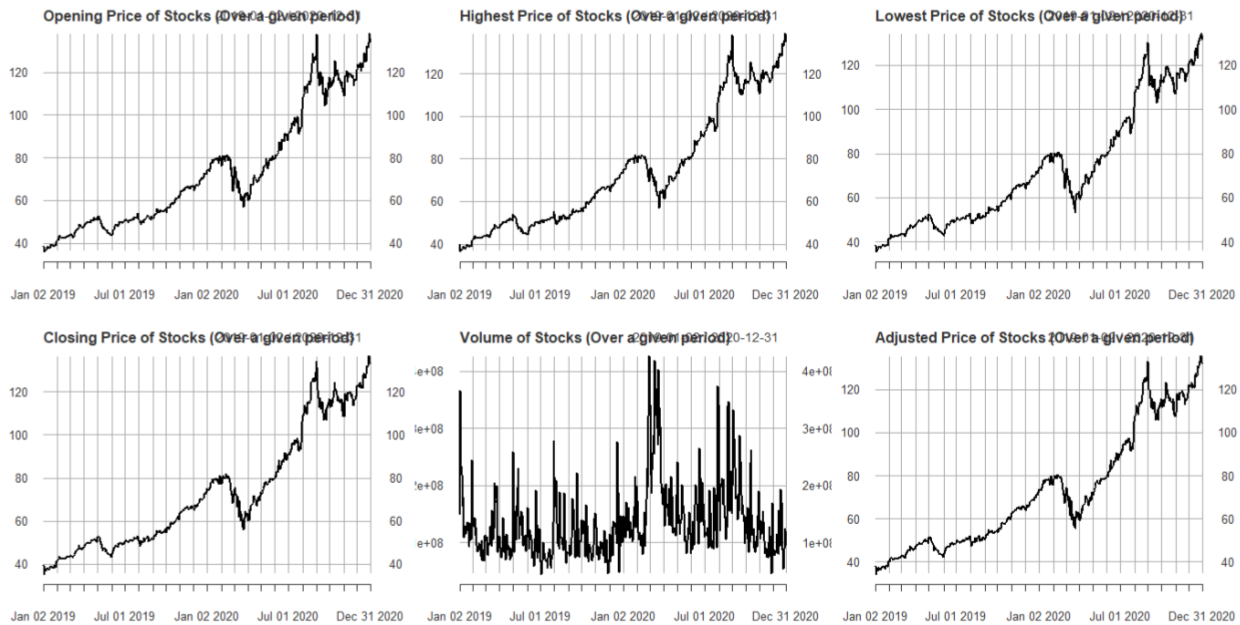

Lowest Price of Stocks (Over a given period)
2019-01-02 / 2020-12-31
120
100
80
60
40
Jan 02 2019
Jul 01 2019
Jan 02 2020
Jul 01 2020
Dec 31 2020


Closing Price of Stocks (Over a given period)
2019-01-02 / 2020-12-31
120
100
80
60
40
Jan 02 2019
Jul 01 2019
Jan 02 2020
Jul 01 2020
Dec 31 2020

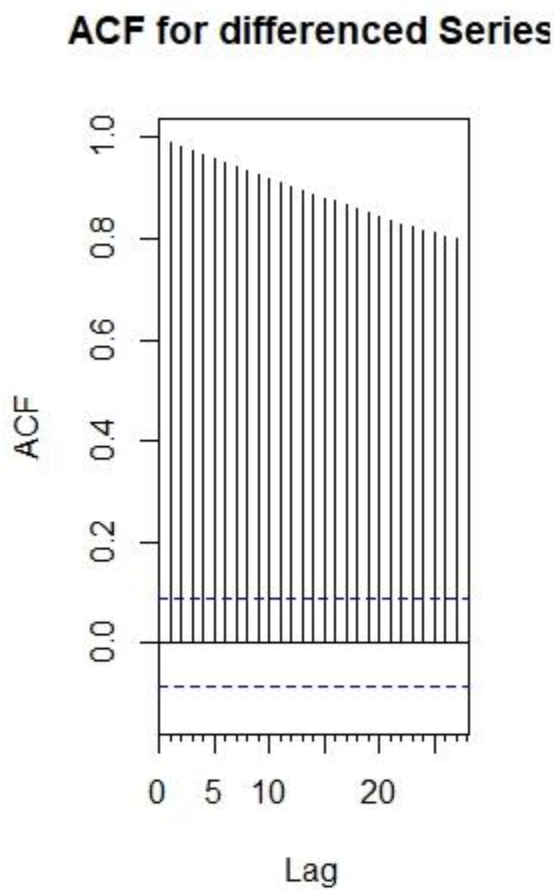

Volume of Stocks (Over a given period)
2019-01-02 / 2020-12-31
4e+08
3e+08
2e+08
1e+08
Jan 02 2019
Jul 01 2019
Jan 02 2020
Jul 01 2020
Dec 31 2020



Adjusted Price of Stocks (Over a given period)
120
100
80
60
40

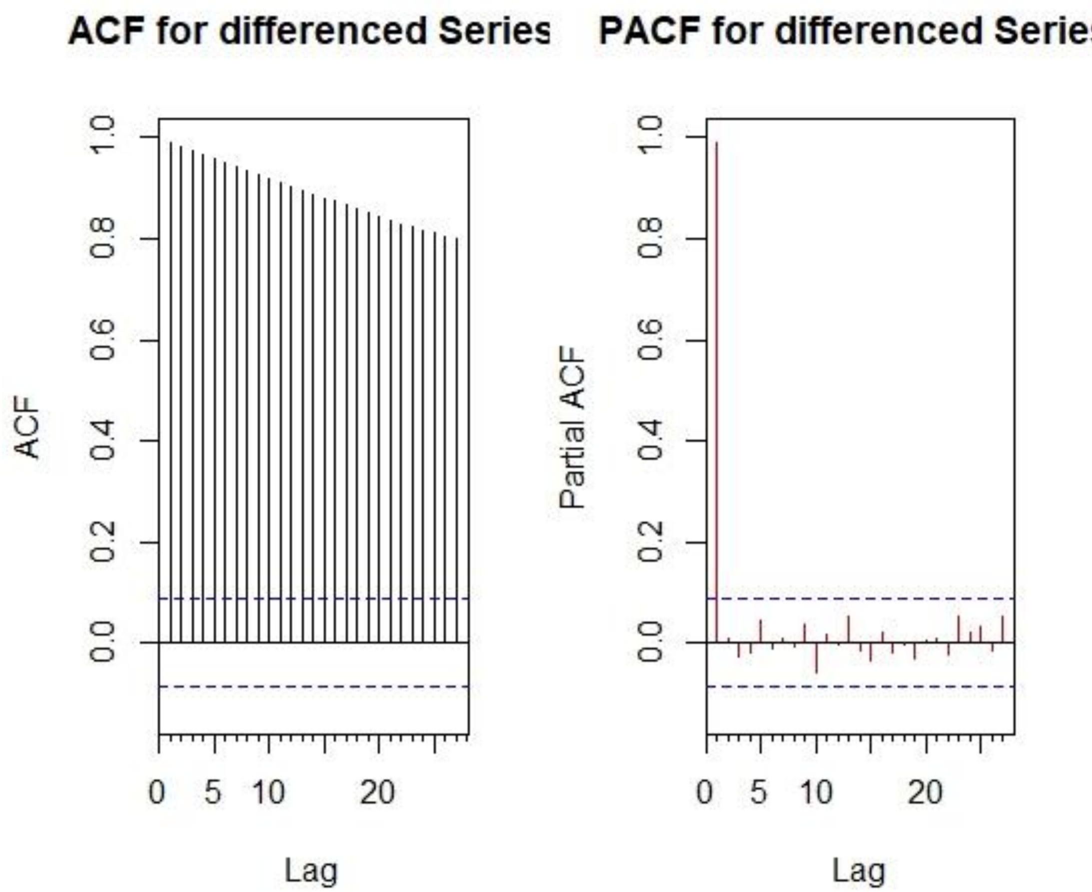



```
Predic_Price = Adjusted_prices

#class(Predic_Price)


######## Finding the Linear Relation between observations ########

par(mfrow = c(1,2))
Acf(Predic_Price, main = 'ACF for differenced Series')
Pacf(Predic_Price, main = 'PACF for differenced Series ', col = '#cc0000')
```

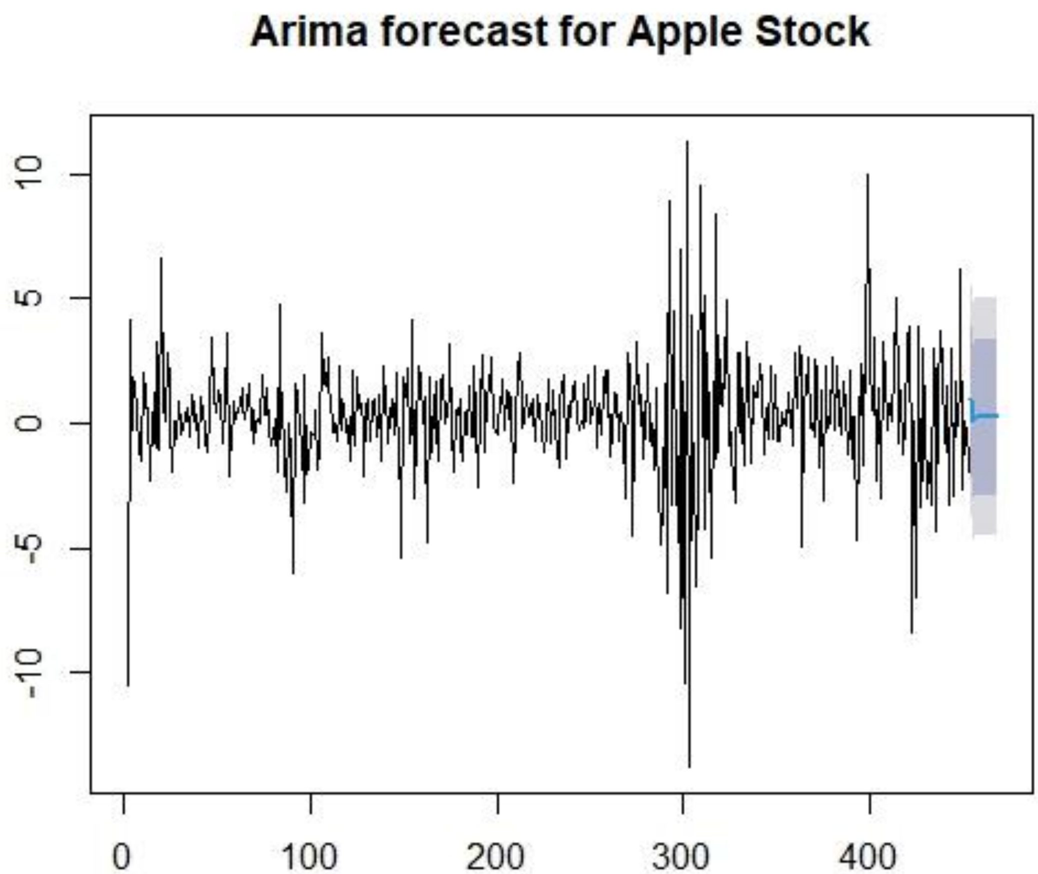



```
Auto_cf = Acf(Predic_Price, plot = FALSE)

Auto_cf
PAuto_cf = Pacf(Predic_Price, plot = FALSE)
PAuto_cf
```

```
print(adf.test(Predic_Price))



################### Prediction of Return ##########################

return_AAPL<- 100*diff(log(Predic_Price))

AAPL_return_train<- return_AAPL[1:(0.9*length(return_AAPL))]

AAPL_return_test<- return_AAPL[(0.9*length(return_AAPL)+1):length(return_AAPL)]

auto.arima(AAPL_return_train, seasonal = FALSE)

fit<- Arima(AAPL_return_train, order = c(1,0,0))

preds<- predict(fit, n.ahead = (length(return_AAPL) -
(0.9*length(return_AAPL))))$pred
preds
```




```
################## Forecasting Predicted Result ##################

test_forecast<- forecast(fit,h = 15)
test_forecast

par(mfrow = c(1,1))
plot(test_forecast, main = "Arima forecast for Apple Stock")

accuracy(preds, AAPL_return_test)
```


Arima forecast for Apple Stock
10
5
0
-5
-10
0
100
200
300
400