APPLE STOCK PRICE FORECASTING

Prepared by (Group 8):

ARUN JAYAKUMAR
MEENAKSHI MOCHERLA
PRIYANKA THACKER
SUREN SUGHAND
UTSAV SEN

INDEX

Introduction:

A little bit about the company's stock, its value and information from a stakeholder's point of view:

About Apple:

- Founded by Steve Jobs, Steve Wozniak, and Ronald Wayne in April 1976.
- Initial purpose was to sell Wozniak's Apple I personal computer.
- Went public in 1980 to instant financial success.
- Launched the first iPhone in 2007.

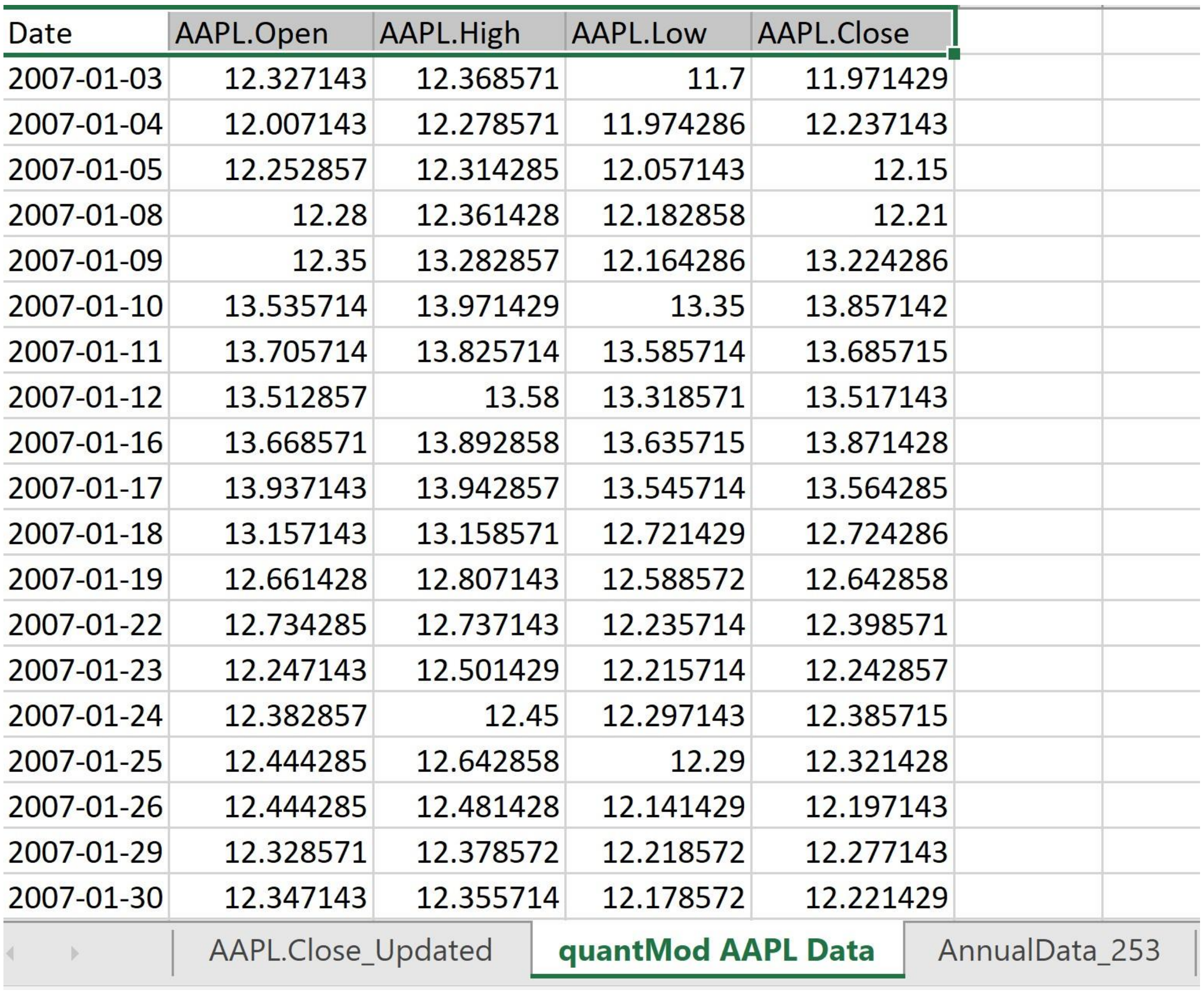
Our project is on Apple INC. stocks from the years- 2007 to 2018. We chose this dataset for our project to understand underlying trends, seasonality and forecasts of this trillion-dollar worth, American multinational technology firm that designs, develops, and sells consumer electronics, computer software, and online services.

Trivia:

- If you invested in Apple a decade ago, you'd probably be feeling pretty good about it today. According to CNBC calculations, a $1,000 investment made in early August 2008 would be worth more than $9,222.50 as of August 2, 2018, or over nine times as much, including price appreciation and excluding dividends.
- There was a stock split of 7:1, that happened on June 9th, 2014. The Kaggle dataset that we started with had values that were not adjusted for this split and hence there was a break in the graph. To avoid this we switched to the Quantmod dataset which had already divided historic values by 7 and hence had no breaks.
- The challenge was to download the Quantmod data into excel for data massaging which we did by the following commands:

```
library(quantmod)
getSymbols("AAPL")
library(WriteXLS)
dataFrame <- data.frame(AAPL)
WriteXLS::testPerl()#This needed Perl installation as well!
WriteXLS(dataFrame, ExcelFileName = "aapl.xlsx",row.names = TRUE )
```

About our dataset:



Date	AAPL.Open	AAPL.High	AAPL.Low	AAPL.Close
2007-01-03	12.327143	12.368571	11.7	11.971429
2007-01-04	12.007143	12.278571	11.974286	12.237143
2007-01-05	12.252857	12.314285	12.057143	12.15
2007-01-08	12.28	12.361428	12.182858	12.21
2007-01-09	12.35	13.282857	12.164286	13.224286
2007-01-10	13.535714	13.971429	13.35	13.857142
2007-01-11	13.705714	13.825714	13.585714	13.685715
2007-01-12	13.512857	13.58	13.318571	13.517143
2007-01-16	13.668571	13.892858	13.635715	13.871428
2007-01-17	13.937143	13.942857	13.545714	13.564285
2007-01-18	13.157143	13.158571	12.721429	12.724286
2007-01-19	12.661428	12.807143	12.588572	12.642858
2007-01-22	12.734285	12.737143	12.235714	12.398571
2007-01-23	12.247143	12.501429	12.215714	12.242857
2007-01-24	12.382857	12.45	12.297143	12.385715
2007-01-25	12.444285	12.642858	12.29	12.321428
2007-01-26	12.444285	12.481428	12.141429	12.197143
2007-01-29	12.328571	12.378572	12.218572	12.277143
2007-01-30	12.347143	12.355714	12.178572	12.221429

- Our dataset consists of stocks from January 2007 to November, 2018. It had 3001 rows originally.
- To ensure a consist frequency across years we found the year with maximum number of trading days i.e. 253 (2008 which was a leap year) and then strategically added the closing value of previous trading day to the next non-trading day for one or more dates as needed to make all years have 253 records. Original data had mix of 250, 251, 252 and 253. The modified data to match frequency of 253 finally had 3015 records.
- For analysis purpose, we divided the data into two parts:
  Training data which consists of 80% of our data and **Test Data** which is 20% of the dataset.

  ### Code:
  80% Train - Fit model
  20% Test; Forecast values and use accuracy function to compute errors, model which will give least error is best model

```
library(fpp2)
library(readxl)
```

Here we are Importing Apple dataset in projectData variable:

```
projectData <- read_excel("C:/Users/user/Desktop/Business Forecasting/BF
Project/aapl.xlsx",sheet = "AAPL.Close_Updated")
```

Extracting adjusted close price in appl_close variable:

```
appl_close <- projectData$AAPL.Close
```

Making close price of Apple(appl_close) into time-series data with frequency 253:

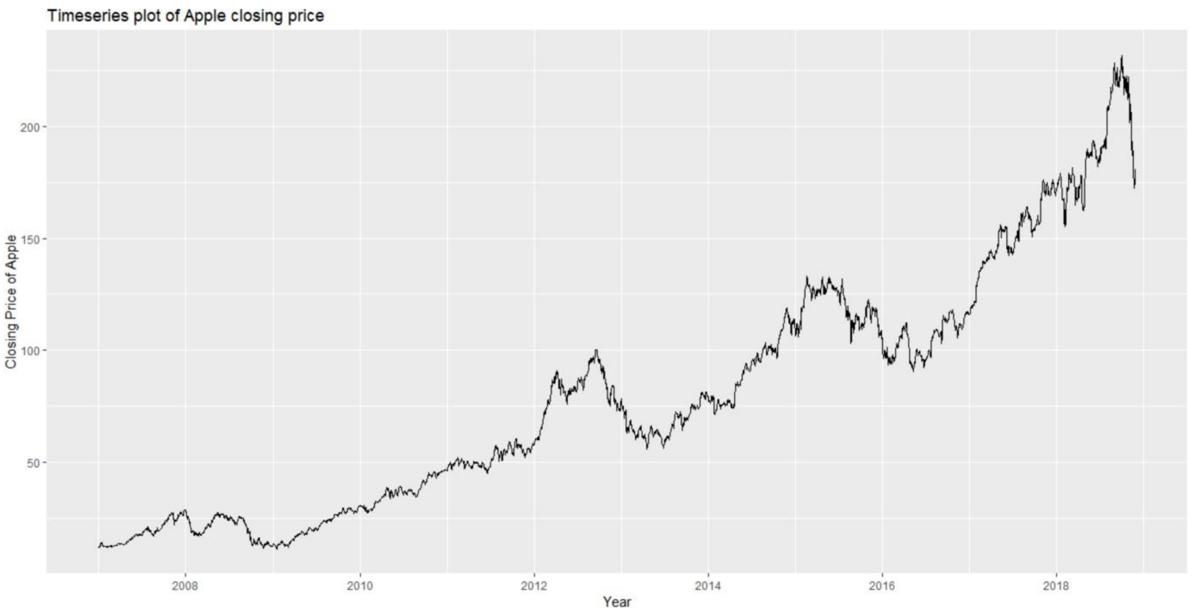
Frequency is 253 as in one year we have only 253 data entries(records)

```
appl_close_ts <- ts(data=appl_close, frequency=253, start=c(2007,1))
```

Plotting close price to observe the behaviour of the data:

```
autoplot(appl_close_ts) + xlab("Year") + ylab("Closing Price of Apple")+
ggtitle("Timeseries plot of Apple closing price")
```

OUTPUT:



Data analysis:

To deep dive into the data, we have used the following techniques and methodologies:

1. Autocorrelation
2. STL Decomposition

   <u>Forecasting Models:</u>
3. Mean Model
4. Naïve Model
5. Seasonal Naive Model
6. Drift Model
7. Regression Model
8. Holts Linear Model
9. ARIMA Model

1. Autocorrelation

Covariance and Correlation measures the extent of a linear relationship between two variables.
Autocorrelation measures the linear relationship between lagged values of a time series.
There are several autocorrelation coefficients, depending on the lag length.
For example, we can measure the relationship between $\square_{\square}$ and $\square_{\square-1}$. Or relationship between $\square_{\square}$ and $\square_{\square-2}$, and so on.
Auto correlation is calculated as follows:

$$r_k = \frac{\sum_{t=k+1}^{T} (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T} (y_t - \bar{y})^2},$$

Where $\square_{\square}$ is the relationship between $\square_{\square}$ and $\square_{\square-\square}$ and T is length of the series.
Assumption: The observations are equi-spaced.

Autocorrelation Function:

The bar in the ACF indicates the autocorrelation, with values between -1 and 1. The first bar indicates how successive values of y relate to each other. The second bar indicates how y values two periods apart relate to each other. The kth bar is almost the same as the sample correlation between $\square\square$ and $\square\square-\square$.
Autocorrelation is defined over different lags, we can plot out the ACF. The plot shows that significant autocorrelation exists in this data.

Trend:

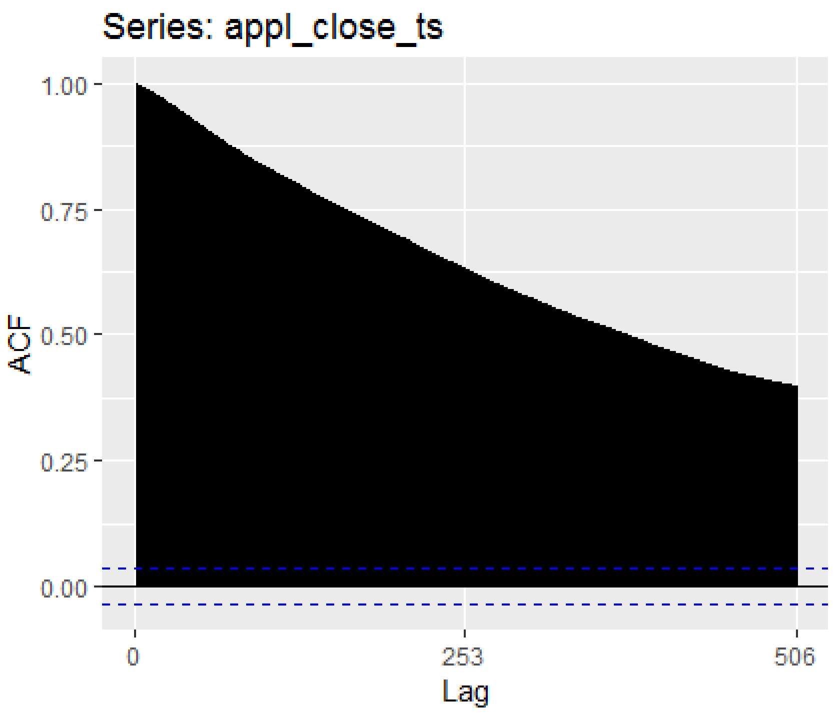
A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend "changing direction" when it might go from an increasing trend to a decreasing trend.

Seasonal:
A seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). Seasonality is always of a fixed and known period. Hence, seasonal time series are sometimes called periodic time series.

Cyclic:
A cyclic pattern exists when data exhibit rises and falls that are not of fixed period. The duration of these fluctuations is usually of at least 2 years.

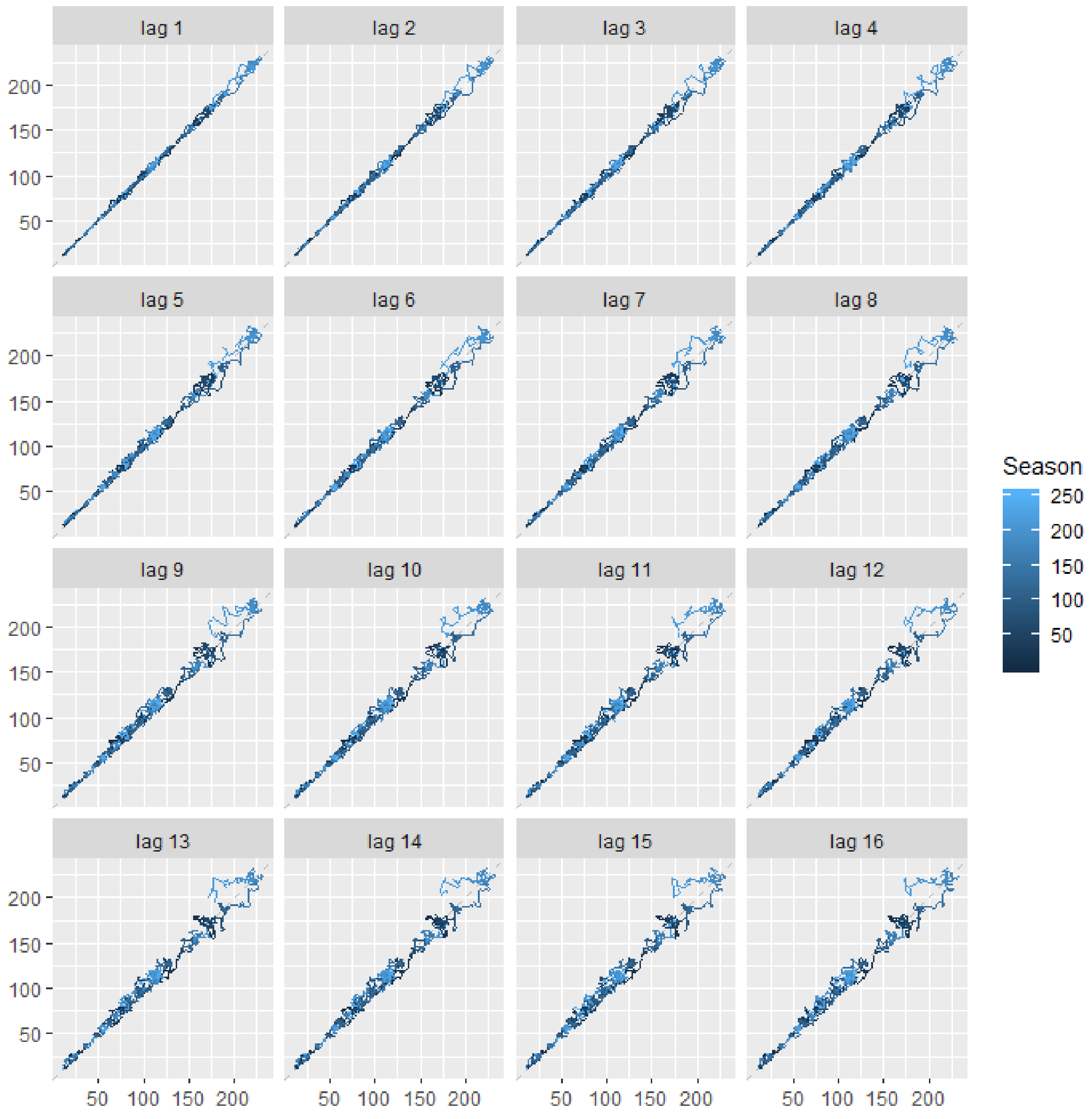
Code for Autocorrelation:
Autocorrelation to find the trend and seasonality:
ggAcf(appl_close_ts)#Constant decrease in ACF is due to trend.

OUTPUT:



Lag plots:
gglagplot(appl_close_ts)

OUTPUT:



NOTE: As seen above there is no seasonality in evident from the first several lags.

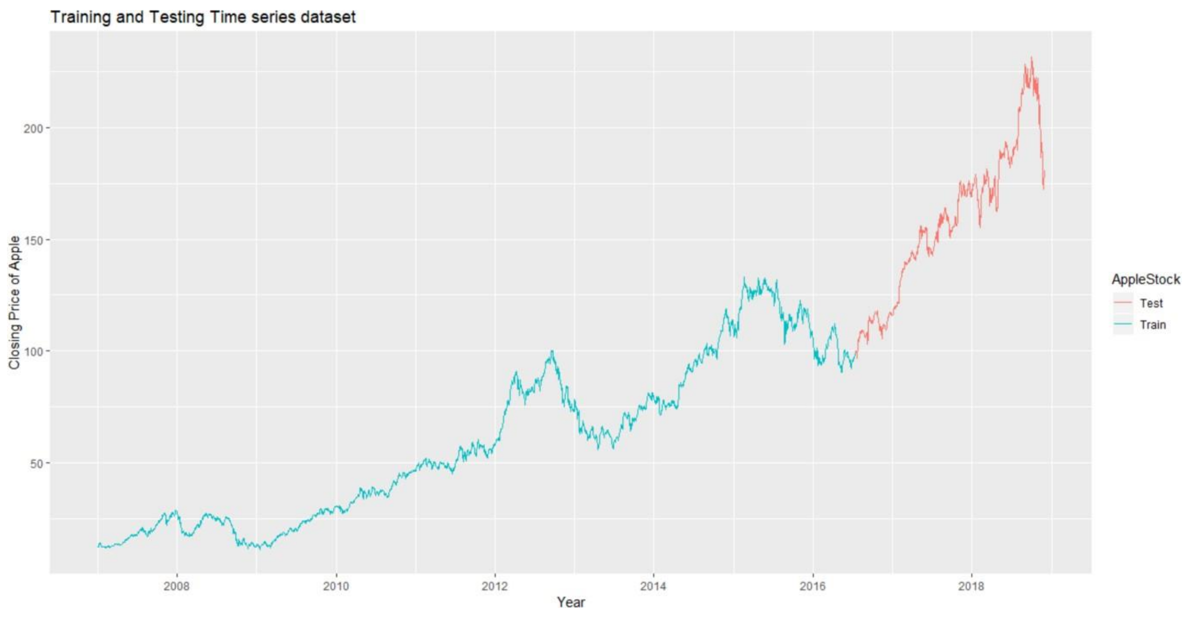
Splitting data in Training and Testing (80:20) set for forecasting:
3015 total records, 2412 training and 603 is test set so we will use h=603

```
appl_training <- window(appl_close_ts, start = c(2007,1), end=c(2016,135))
appl_testing <- window(appl_close_ts, start= c(2016,136), end=c(2018,232))
```

Plotting Traning and testing set:

```
autoplot(appl_training,series="Train") +
autolayer(appl_testing,series="Test") +
xlab("Year") + ylab("Closing Price of Apple") +
  ggtitle("Training and Testing Time series dataset") +
  guides(colour=guide_legend(title="AppleStock"))
```

OUTPUT:



2. STL Decomposition

- We are decomposing apple close price of the stocks to analyze Trend- Cycle and Seasonality.
- We have used the additive model in this case as the additive model is useful when the seasonal variation is relatively constant over time.
- Additive model

Additive: y_t = Seasonal + Trend + Random

□□ = □□ + □□ + □□

Decomposition procedures are used in time series to describe the trend and seasonal factors in a time series. More extensive decompositions might also include long-run cycles, holiday effects, day of week effects and so on. Here, we'll only consider trend and seasonal decompositions.

Code for STL Model:

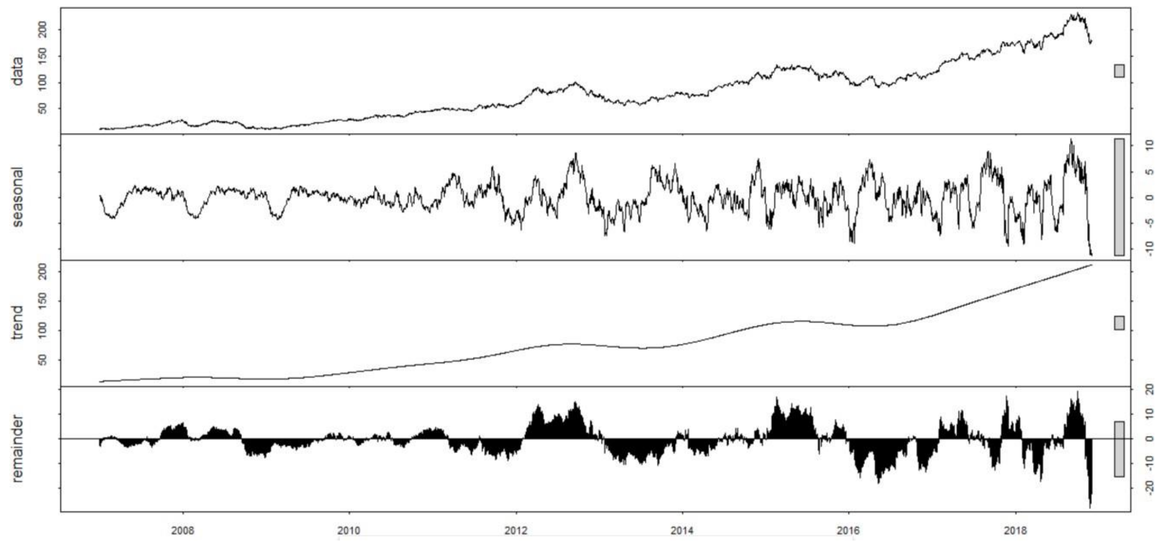
Decomposing apple close price of stock to analyze Trend- Cycle and Seasonality using STL:
s.window is the number of consecutive years to be used in estimating each value in the seasonal component. The user must specify s.window as there is no default. Setting it to be infinite is equivalent to forcing the seasonal component to be periodic (i.e., identical across years).

```
appl_stl<-stl(appl_close_ts, s.window = 5)
```

STL decomposed components plot:

```
plot(appl_stl)
```

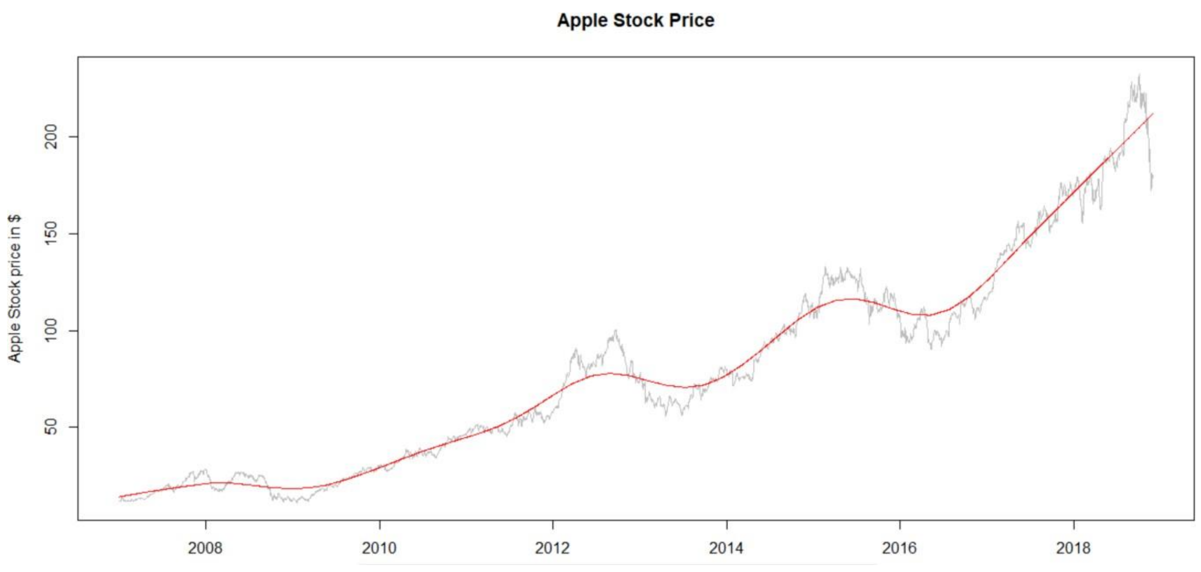
OUTPUT:



Data plotted with Trend component:

```
plot(appl_close_ts, col="gray",
     main="Apple Stock Price",
     ylab="Apple Stock Price in $", xlab="Time")
lines(appl_stl$time.series[,"trend"],col="red",ylab="Trend")
```

OUTPUT:



Seasonal adjusted plot:

```
seasadj(appl_stl)
```

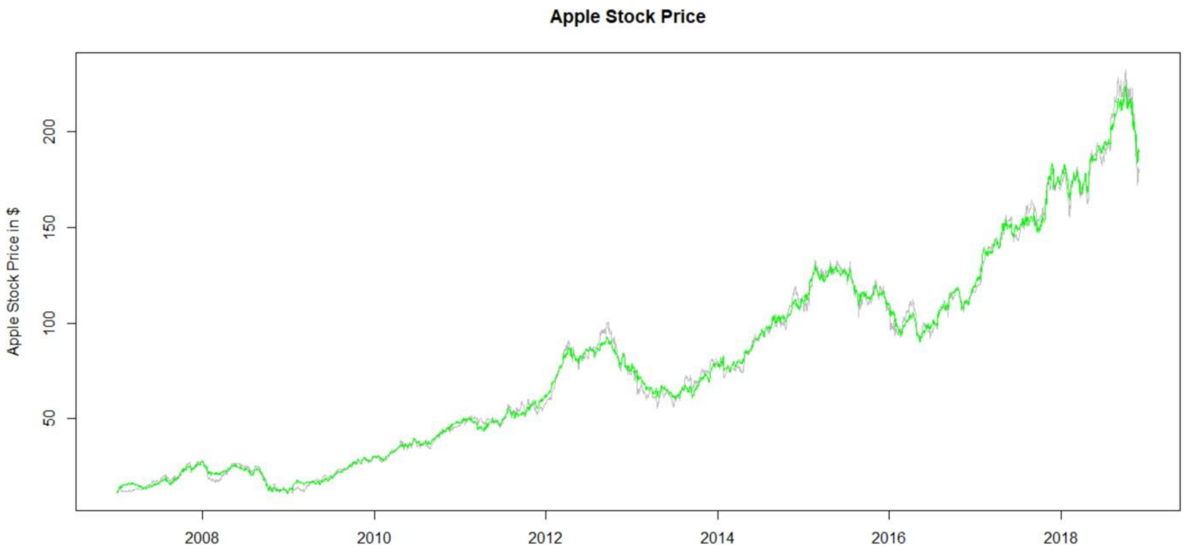
OUTPUT:

```
Time Series:
Start = c(2007, 1)
End = c(2018, 232)
Frequency = 253
   [1] 13.46080 14.00527 14.66148 14.70105 14.35892 15.27825 15.92082 16.04933
15.68166 16.73136 16.25500
  [12] 15.48673 15.44751 14.86352 15.05120 16.06401 16.93740 16.62235 16.55765
17.05807 16.37420 16.07586
  [23] 16.65233 16.83053 16.20228 16.36999 16.30102 15.32146 14.98928 14.18091
13.79171 13.70429 13.86069
  [34] 14.13232 14.69109 14.78618 14.77876 14.23611 13.67845 13.04997 13.41594
13.63577 13.31843 13.70172
  [45] 14.50298 14.21896 14.07125 13.82866 14.01929 13.67230 13.55709 13.54164
13.22786 13.65861 13.85959
  [56] 14.19117 14.55917 14.07408 14.13888 14.25045 13.77179 14.30643 14.05315
13.52458 13.33182 14.21723
  [67] 14.28527 13.78272 13.81064 13.63415 13.78945 13.80178 14.33279 14.43393
14.52101 15.16500 15.14578
  [78] 15.39406 15.00174 15.04787 15.80222 15.59214 15.69464 14.61712 14.77858
14.23301 14.27756 14.76878
  [89] 14.42155 14.68063 14.86220 15.24509 15.37488 15.86979 15.47636 16.02139
15.86903 15.79537 15.78338
```

```
 [100] 15.46777 14.70901 15.19751 15.24460 15.61768 15.73738 15.46351 15.71286
15.93292 16.40786 17.10174
 [ reached getOption("max.print") – omitted several entries]
```

```
plot(appl_close_ts, col="grey",
     main="Apple Stock Price",
     ylab="AAPL Sotck price in $", xlab="Time")
lines(seasadj(appl_stl),col="green",ylab="Seasonally adjusted")
```

OUTPUT:



3. Mean Model

The forecasts of all future values are equal to the mean of the historical data.

If we let the historical data be denoted by ☐1, … , ☐T, then we can write the forecasts as

$$\hat{y}_{T+h|T} = \bar{y} = (y_1 + \cdots + y_T)/T.$$

meanf(y, h)
# y contains the time series
# h is the forecast horizons

Code for Mean Model:

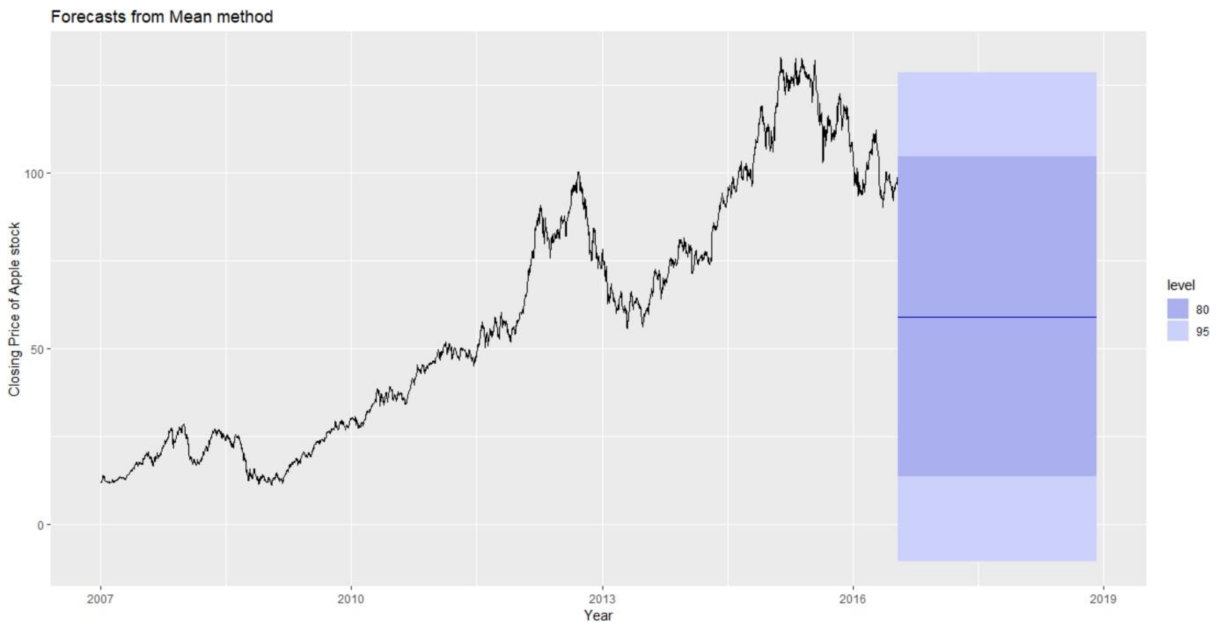
Train Mean model of Closing Price:

```
appl_mean <- meanf(appl_training,h=603)
```

Forecast of Apple Closing Price using Mean method:

```
autoplot(appl_mean) +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts from Mean method")
```

OUTPUT:

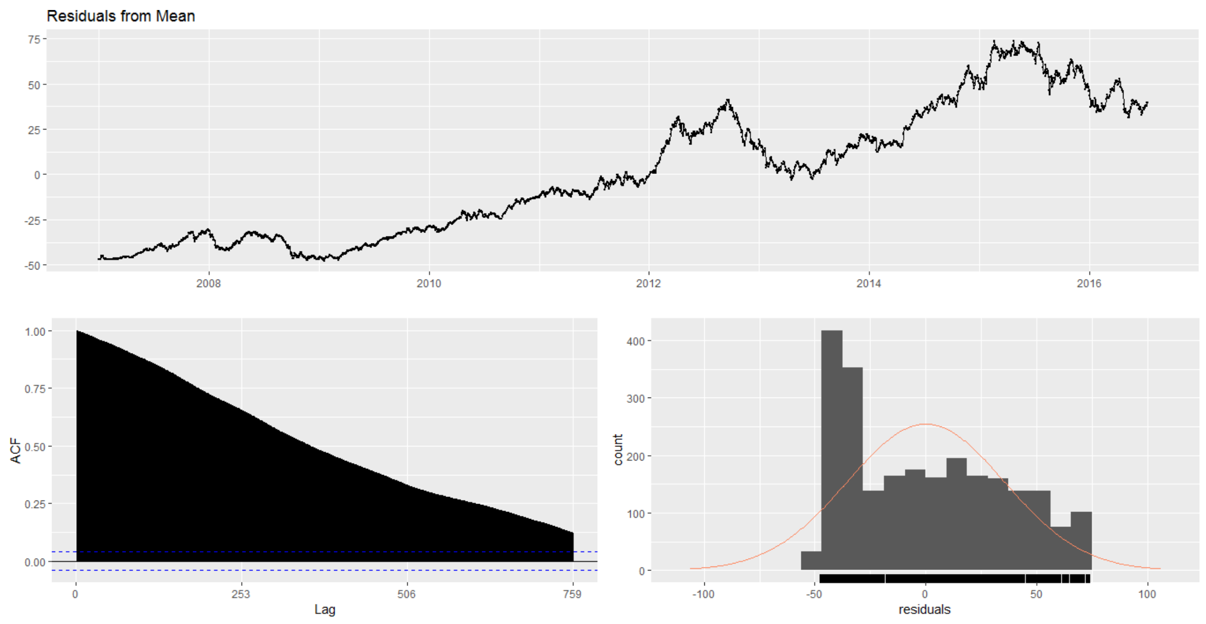


Check Residuals:

```
checkresiduals(appl_mean)
```

OUTPUT:

```
Ljung-Box test
data: Residuals from Mean
Q* = 615100, df = 481.4, p-value < 2.2e-16
Model df: 1. Total lags used: 482.4
```

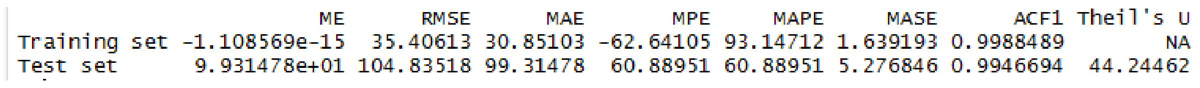


NOTE: As seen above the residuals are not randomly distributed. There is a clear trend in the residuals and no normal distribution in histogram. Therefore, this is not a good model for our dataset.

Compute forecast accuracy measures of mean method:

accuracy(appl_mean, appl_testing)

OUTPUT:



```
                        ME      RMSE      MAE       MPE     MAPE     MASE      ACF1 Theil's U
Training set -1.108569e-15  35.40613 30.85103 -62.64105 93.14712 1.639193 0.9988489        NA
Test set      9.931478e+01 104.83518 99.31478  60.88951 60.88951 5.276846 0.9946694  44.24462
```

4. Naïve Model

For naïve forecasts, we simply set all forecasts to be the value of the last observation. This model is optimal for efficient stock markets as closing price for the stocks is the starting price for the next day.

Forecasts: $\hat{y}_{n+h|n} = y_n$

Code for Naïve Model:

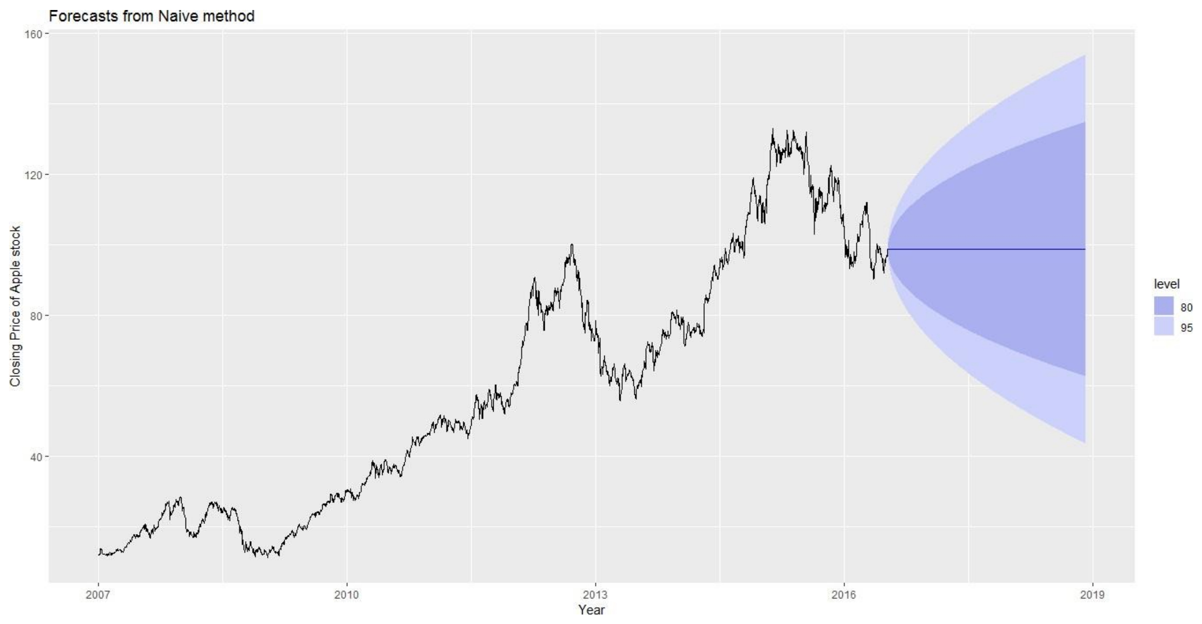
Naive model on training data of Closing Price:

```
appl_naive <- rwf(appl_training,h=603)
```

Forecast of Apple Closing Price using Naive method:

```
autoplot(appl_naive) +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts from Naive method")
```

OUTPUT:

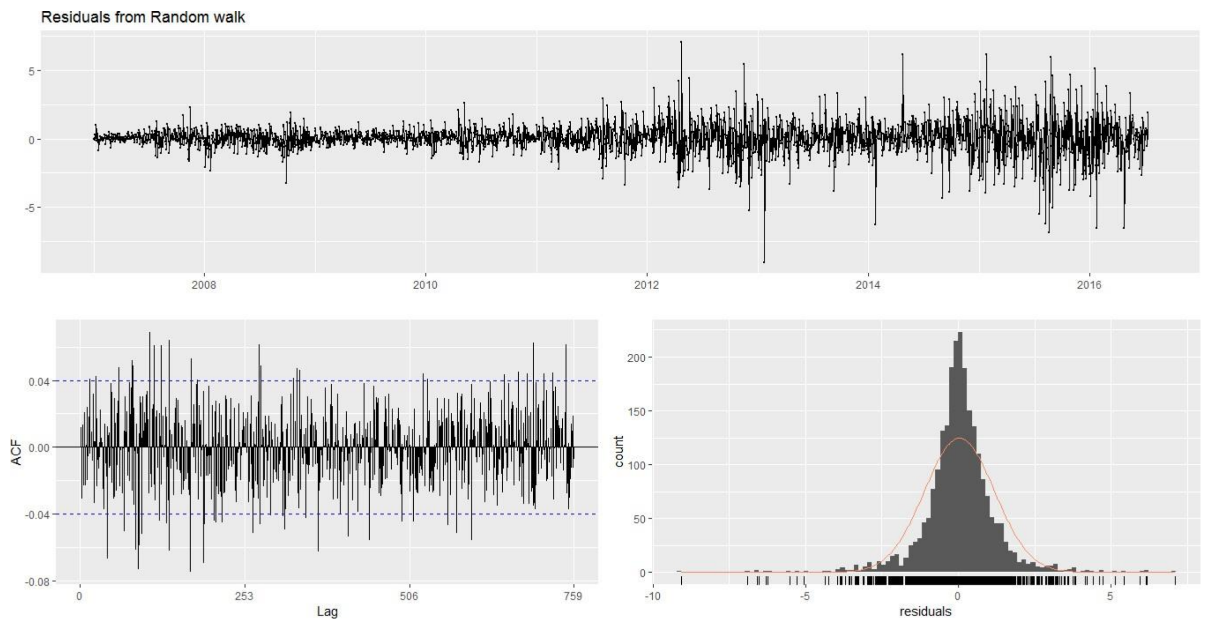


Check Residuals:

```
checkresiduals(appl_naive)
```

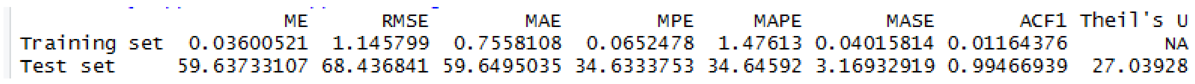
OUTPUT:

```
Ljung-Box test
data: Residuals from Random walk
Q* = 701.68, df = 482.4, p-value = 2.371e-10
Model df: 0. Total lags used: 482.4
```



Computing forecast accuracy measures of Naive method:

accuracy(appl_naive, appl_testing)



```
                      ME      RMSE        MAE        MPE     MAPE       MASE       ACF1 Theil's U
Training set  0.03600521  1.145799  0.7558108  0.0652478  1.47613 0.04015814 0.01164376        NA
Test set     59.63733107 68.436841 59.6495035 34.6333753 34.64592 3.16932919 0.99466939  27.03928
```

5. Seasonal Naive Model

We set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year).
Formally, the forecast for time T+h is written as

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

Where m= the seasonal period and k is the integer part of (h−1)/m (i.e., the number of complete years in the forecast period prior to time T+h)

Code for Seasonal Naive Model:

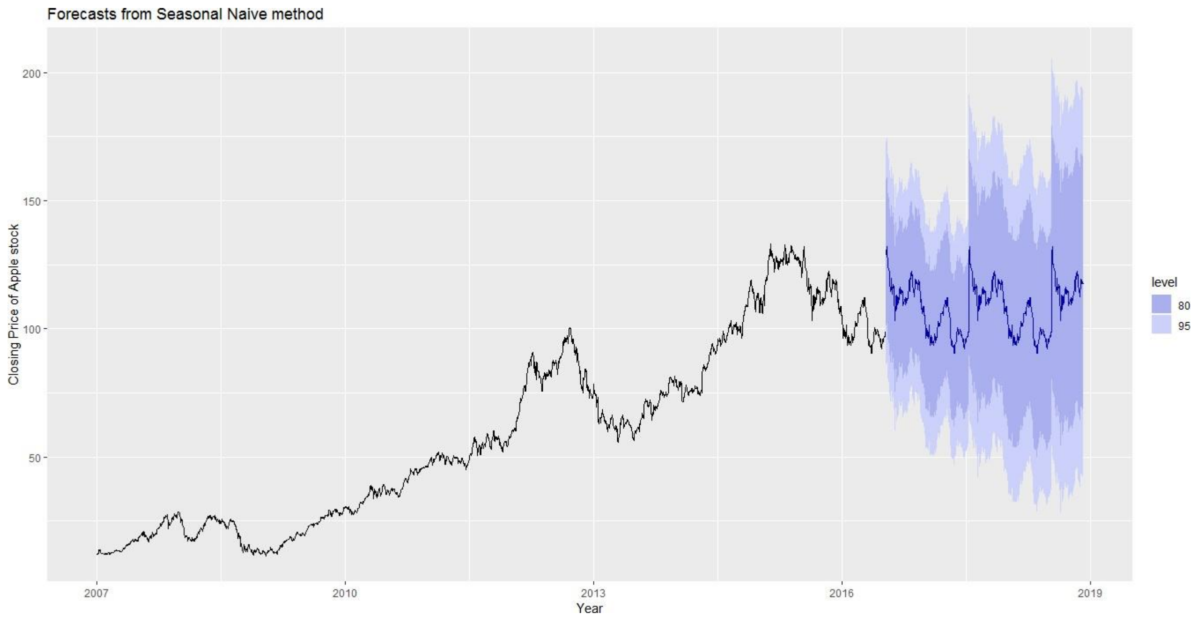
Seasonal Naive model of Closing Price on training data:

```
appl_snaive <- snaive(appl_training,h=603)
```

Forecast of Apple Closing Price using Seasonal Naive method:

```
autoplot(appl_snaive) +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts from Seasonal Naive method")
```

OUTPUT:

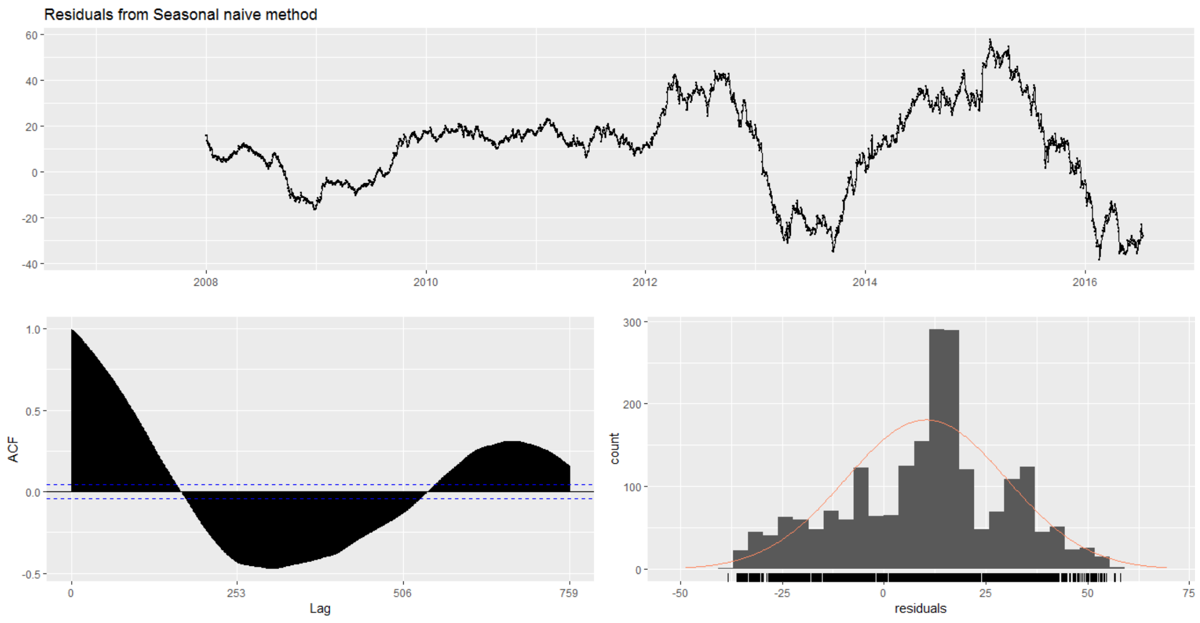


Checking Residuals:

```
checkresiduals(appl_snaive)
```

OUTPUT:

```
Ljung-Box test
data: Residuals from Seasonal naive method
Q* = 249980, df = 482.4, p-value < 2.2e-16
Model df: 0. Total lags used: 482.4
```



NOTE: As seen above the residuals are not randomly distributed. There is a clear trend in the residuals and no normal distribution in histogram. Therefore, this is not a good model for our dataset.

Computing forecast accuracy measures of Seasonal Naive method:

accuracy(appl_snaive, appl_testing)

```
                   ME     RMSE      MAE      MPE     MAPE     MASE      ACF1 Theil's U
Training set 10.35607 22.25247 18.82086 13.18451 32.88206 1.000000 0.9959151        NA
Test set     50.29479 61.45184 52.62289 28.26428 30.47388 2.795987 0.9907513  24.33266
```

6. Drift Model

A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the **drift**) is set to be the average change seen in the historical data. Thus, the forecast for time T+h is given by

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right).$$

This is equivalent to drawing a line between the first and last observations, and extrapolating it into the future.
rwf(y,h,drift=TRUE)

Code for Drift Model:

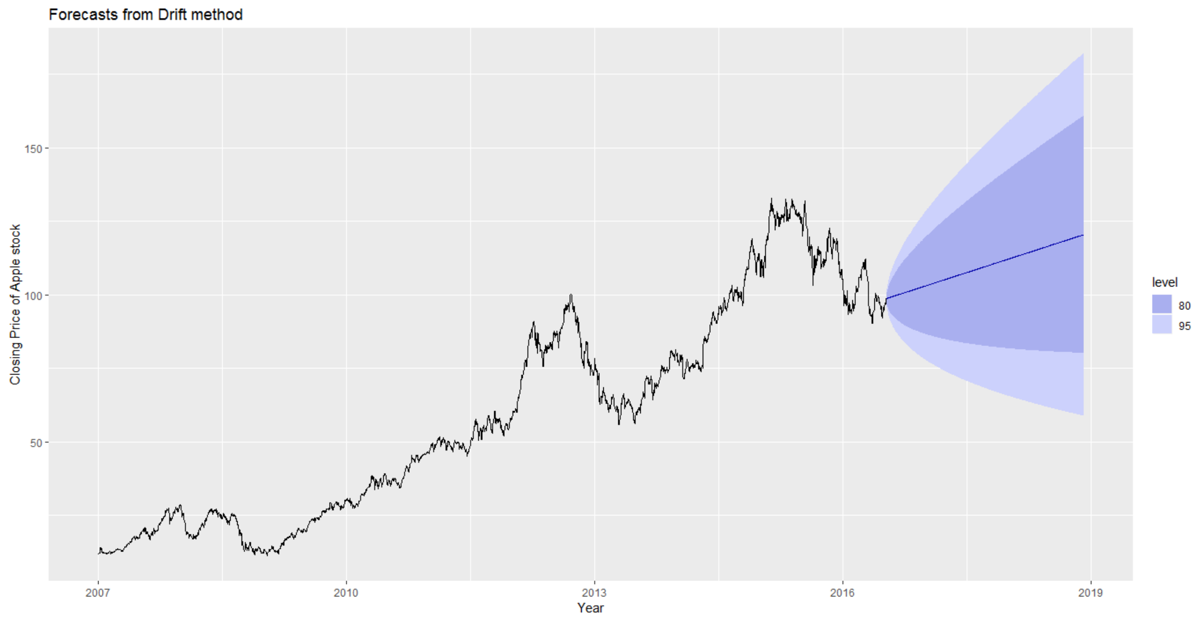
Train Drift model of Closing Price:

```
appl_drift <- rwf(appl_training,h=603,drift=TRUE)
```

Forecast of Apple Closing Price using Drift method:

```
autoplot(appl_drift) +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts from Drift method")
```

OUTPUT:

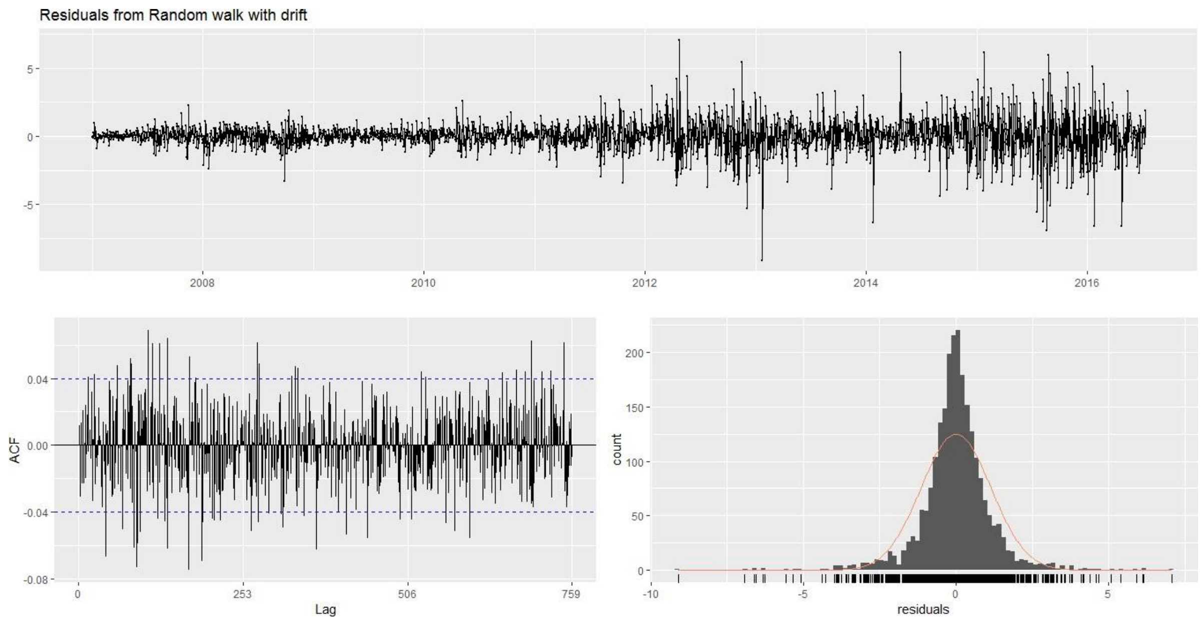


Check Residuals:

checkresiduals(appl_drift)

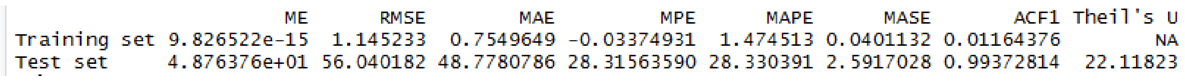
OUTPUT:

Ljung-Box test
data: Residuals from Random walk with drift
Q* = 701.68, df = 481.4, p-value = 1.954e-10
Model df: 1. Total lags used: 482.4



Computing forecast accuracy measures of Drift method:

accuracy(appl_drift, appl_testing)



```
                       ME      RMSE        MAE         MPE      MAPE      MASE       ACF1 Theil's U
Training set 9.826522e-15  1.145233  0.7549649 -0.03374931  1.474513 0.0401132 0.01164376        NA
Test set     4.876376e+01 56.040182 48.7780786 28.31563590 28.330391 2.5917028 0.99372814  22.11823
```

7. Regression Model

Simple Linear Regression:

In the simplest case, the regression model allows for a linear relationship between the forecast variable y and a single predictor variable x:

$$y_t=\beta_0+\beta_1 x_t+\varepsilon_t.$$

The coefficients β0 and β1 denote the intercept and the slope of the line respectively.
The intercept β0 represents the predicted value of y when x=0
The slope β1 represents the average predicted change in y resulting from a one unit increase in x.

Multiple Linear Regression:

When there are two or more predictor variables, the model is called a **multiple regression model**. The general form of a multiple regression model is

$$y_t=\beta_0+\beta_1 x_{1,t}+\beta_2 x_{2,t}+\cdots+\beta_k x_{k,t}+\varepsilon_t,$$

where y is the variable to be forecast and $x_1,\ldots,x_k$ are the k predictor variables. Each of the predictor variables must be numerical. The coefficients $\beta_1,\ldots,\beta_k$ measure the effect of each predictor after taking into account the effects of all the other predictors in the model. Thus, the coefficients measure the *marginal effects* of the predictor variables.

Code for Regression Model:

Train regression model on Apple closing price:
Running regression model with Trend and season as the predictor variables:

```
appl_reg <- tslm(appl_training ~ trend + season)
```

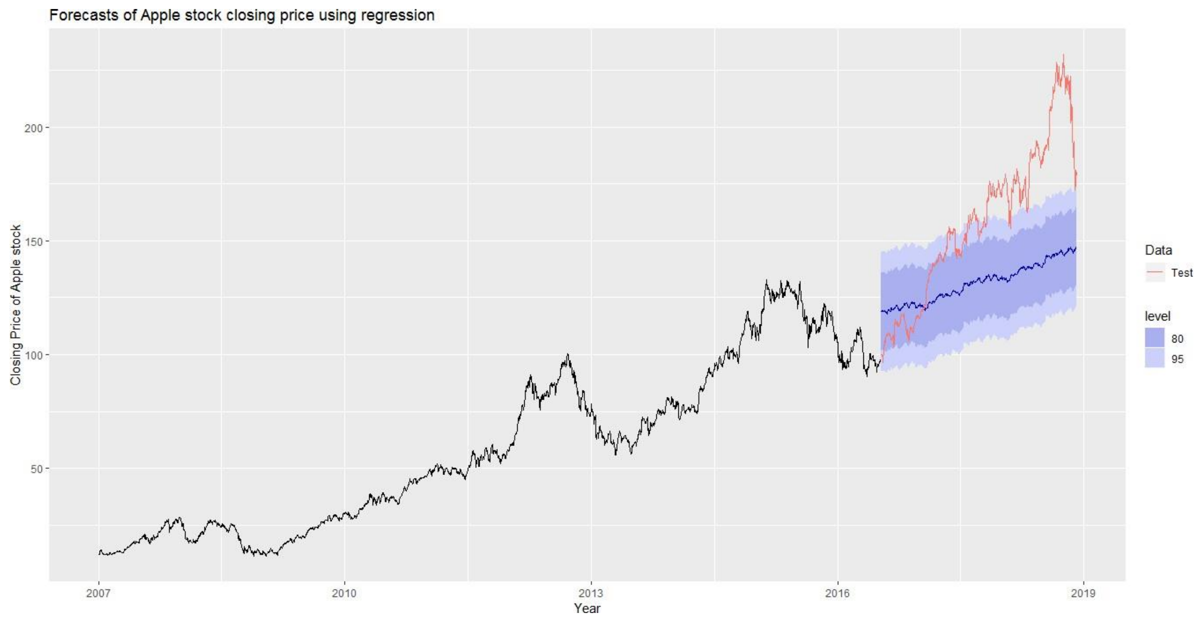
Forecast of Apple Closing Price using Regression Model:

```
appl_reg_forecast <- forecast(appl_reg, h= 603)
```

Plot of forecast to test data for regression:

```
autoplot(appl_reg_forecast) +
autolayer(appl_testing, series = "Test") +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts of Apple stock closing price using regression") +
  guides(colour = guide_legend(title = "Data"))
```

OUTPUT:

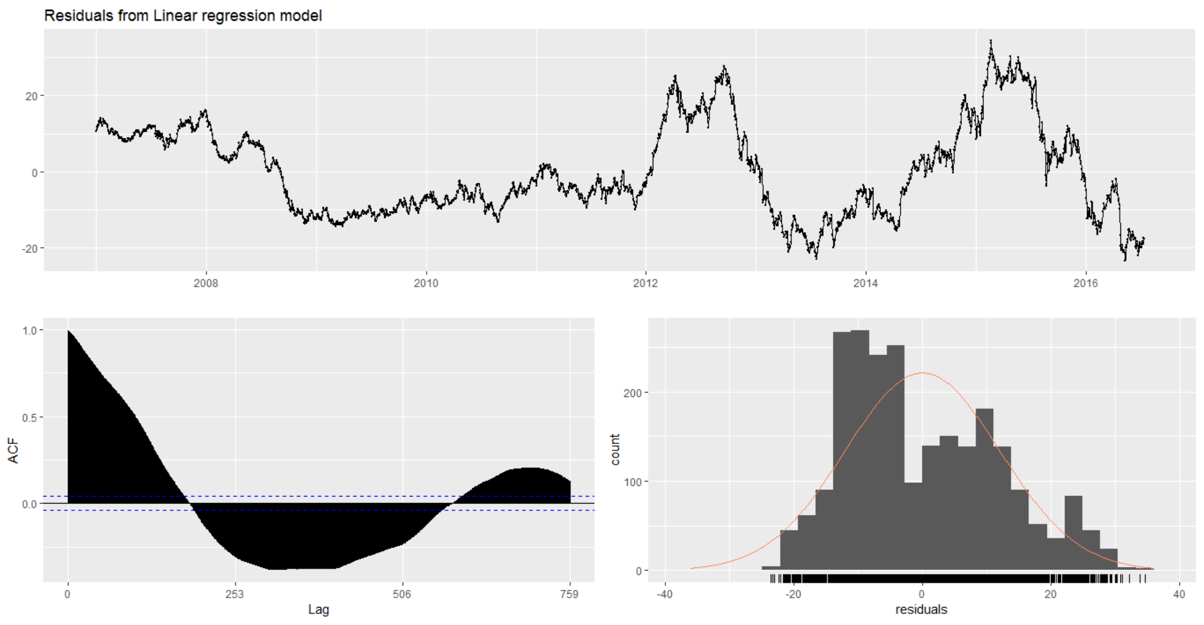


Check Residuals:

checkreciduals(appl_reg)

OUTPUT:

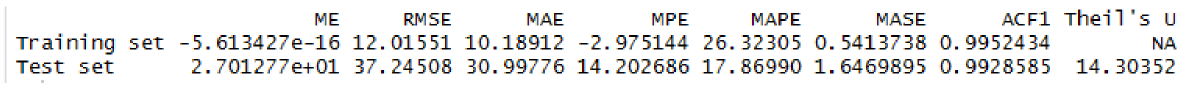
Breusch-Godfrey test for serial correlation of order up to 482
data: Residuals from Linear regression model
LM test = 2397.2, df = 482, p-value < 2.2e-16



NOTE: As seen above the residuals are not randomly distributed. There is a clear trend in the residuals and no normal distribution in histogram. Therefore, this is not a good model for our dataset.

Computing forecast accuracy measures of Regression method:

accuracy(appl_reg_forecast, appl_testing)



```
                        ME     RMSE      MAE       MPE     MAPE      MASE      ACF1 Theil's U
Training set -5.613427e-16 12.01551 10.18912 -2.975144 26.32305 0.5413738 0.9952434        NA
Test set      2.701277e+01 37.24508 30.99776 14.202686 17.86990 1.6469895 0.9928585  14.30352
```

8. Holt Linear Model

Holt Linear also known as double exponential smoothing computes a trend equation through the data using a special weighting function that places the greatest emphasis on the most recent time periods. The forecasting equation changes from period to period.
The forecasting algorithm makes use of the following formulas:

$$\mathbf{F_t=a_t+b_t}$$

$$\mathbf{a_t = X_t + (1-\alpha)_{2et}}$$

$$\mathbf{b_t=b_{t-1} + \alpha_2 e_t \; e_t=F_t-X_t}$$

The smoothing constant, α , dictates the amount of smoothing that takes place. It ranges from zero to one. The forecast at time period T for the value at time period T+k is $a_T + b_T k$.
We are using this method to observe the trend of the closing price.

Code for Holt Linear Model:

Forecast of Apple Closing Price using Holt's linear trend method:

```
appl_holt <- holt(appl_training, h= 603)
```

Smoothing Parameters:

```
appl_holt[["model"]]
```

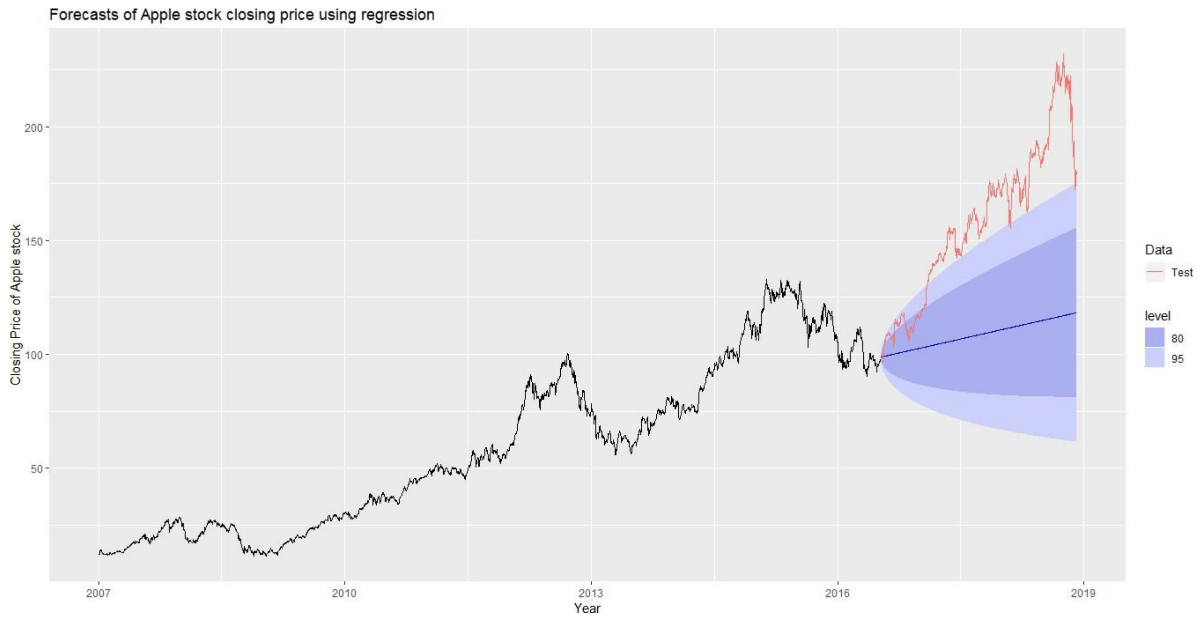
OUTPUT:

```
Holt's method

Call:
 holt(y = appl_training, h = 603)

  Smoothing parameters:
    alpha = 0.9999
    beta = 1e-04

  Initial states:
    l = 13.3848
    b = 0.0316

  sigma: 1.1464

     AIC     AICc      BIC
19450.20 19450.23 19479.15
```

Plot of forecast to test data for Holt's linear:

```
autoplot(appl_holt, series = "Forecast") +
autolayer(appl_testing, series = "Test") +
  xlab("Year") + ylab("Closing Price of Apple stock") +
  ggtitle("Forecasts of Apple stock closing price using regression") +
  guides(colour = guide_legend(title = "Data"))
```

OUTPUT:

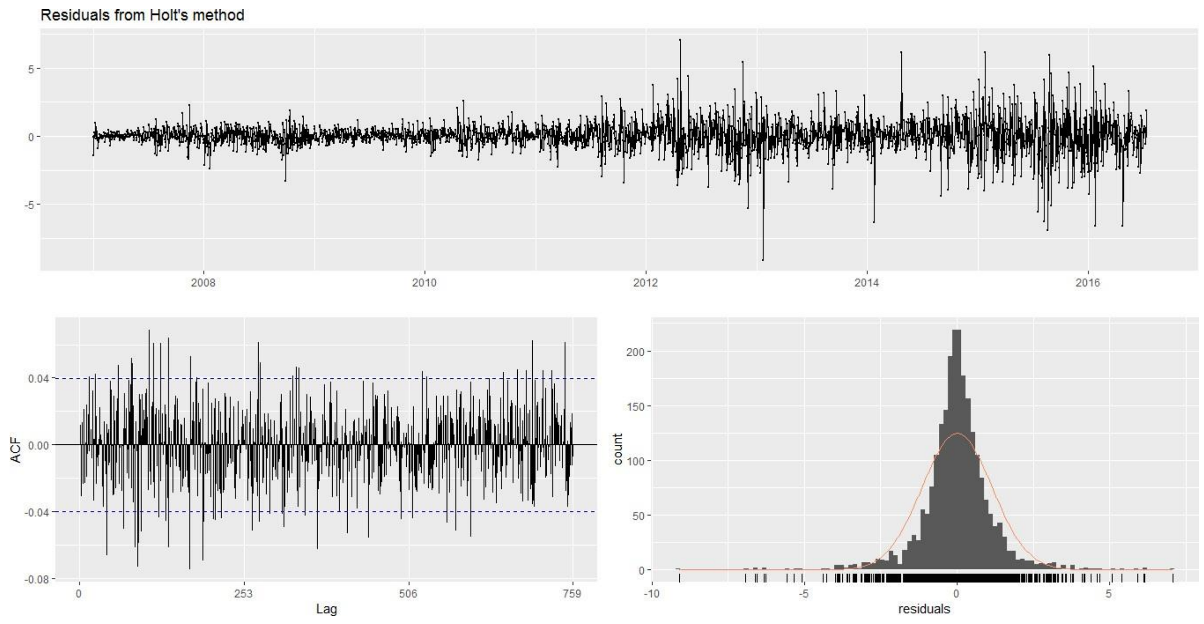


Check Residuals:

checkresiduals(appl_holt)

OUTPUT:

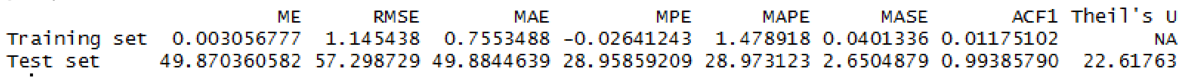
```
Ljung-Box test
data: Residuals from Holt's method
Q* = 701.59, df = 478.4, p-value = 1.105e-10
Model df: 4. Total lags used: 482.4
```



Computing forecast accuracy measures of Holt's linear:

accuracy (appl_holt, appl_testing)

OUTPUT:



```
                      ME      RMSE       MAE         MPE     MAPE      MASE       ACF1 Theil's U
Training set  0.003056777  1.145438 0.7553488 -0.02641243 1.478918 0.0401336 0.01175102        NA
Test set     49.870360582 57.298729 49.8844639 28.95859209 28.973123 2.6504879 0.99385790  22.61763
```

9. ARIMA

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.

Auto Regressive (AR) Model:

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of *past values of the variable*. The term *auto*regression indicates that it is a regression of the variable against itself.
Thus, an autoregressive model of order p can be written as

$$y_t=c+\phi_1 y_{t-1}+\phi_2 y_{t-2}+\cdots+\phi_p y_{t-p}+\varepsilon_t,$$

where εt is white noise. This is like a multiple regression but with *lagged values* of yt as predictors. We refer to this as an **AR**(p) **model**, an autoregressive model of order p.
Autoregressive models are remarkably flexible at handling a wide range of different time series patterns.

Moving Averages (MA) Model:

Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t=c+\varepsilon_t+\theta_1\varepsilon_t+\theta_2\varepsilon_{t-2}+\cdots+\theta_q\varepsilon_{t-q},$$

where εt is white noise. We refer to this as an **MA(**q**) model**, a moving average model of order q. Of course, we do not *observe* the values of εt, so it is not really a regression in the usual sense.
Notice that each value of yt can be thought of as a weighted moving average of the past few forecast errors.
A moving average model is used for forecasting future values, while moving average smoothing is used for estimating the trend-cycle of past values.

Code for ARIMA Model:
Using Auto.Arima to fit best ARIMA model (finding p,d,q)

```
fit <- auto.arima(appl_training)
fit
```

OUTPUT:

```
Series: appl_training
ARIMA(0,1,0) with drift

Coefficients:
     drift
     0.0360
s.e. 0.0233

sigma^2 estimated as 1.312: log likelihood=-3748.01
AIC=7500.02 AICc=7500.03 BIC=7511.
```

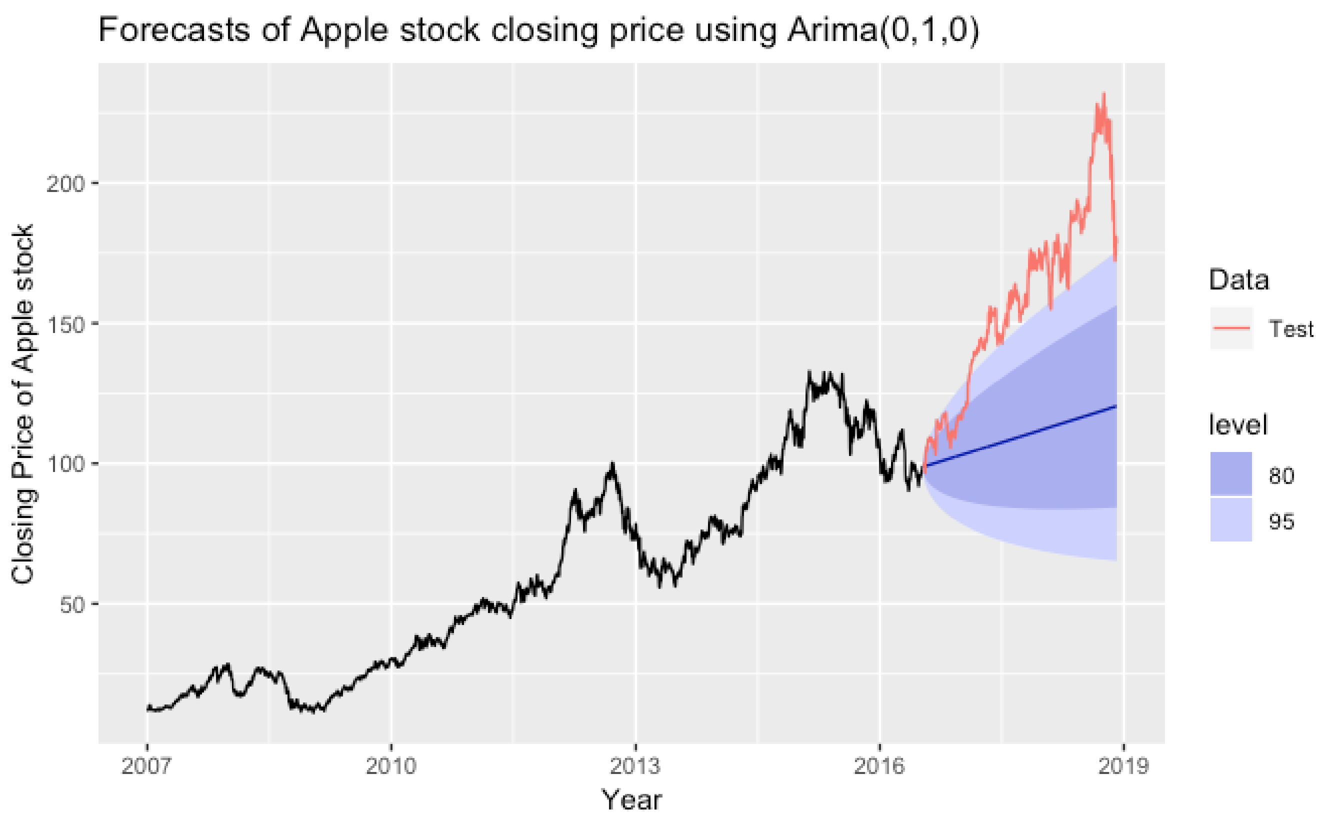
Fitting ARIMA(0,1,0) for apple training data:

```
fit2 <- Arima(appl_training,order=c(0,1,0),include.drift=TRUE)
appl_arima <- forecast(fit2, h=603)
```

Plot of forecast to test data for Arima:

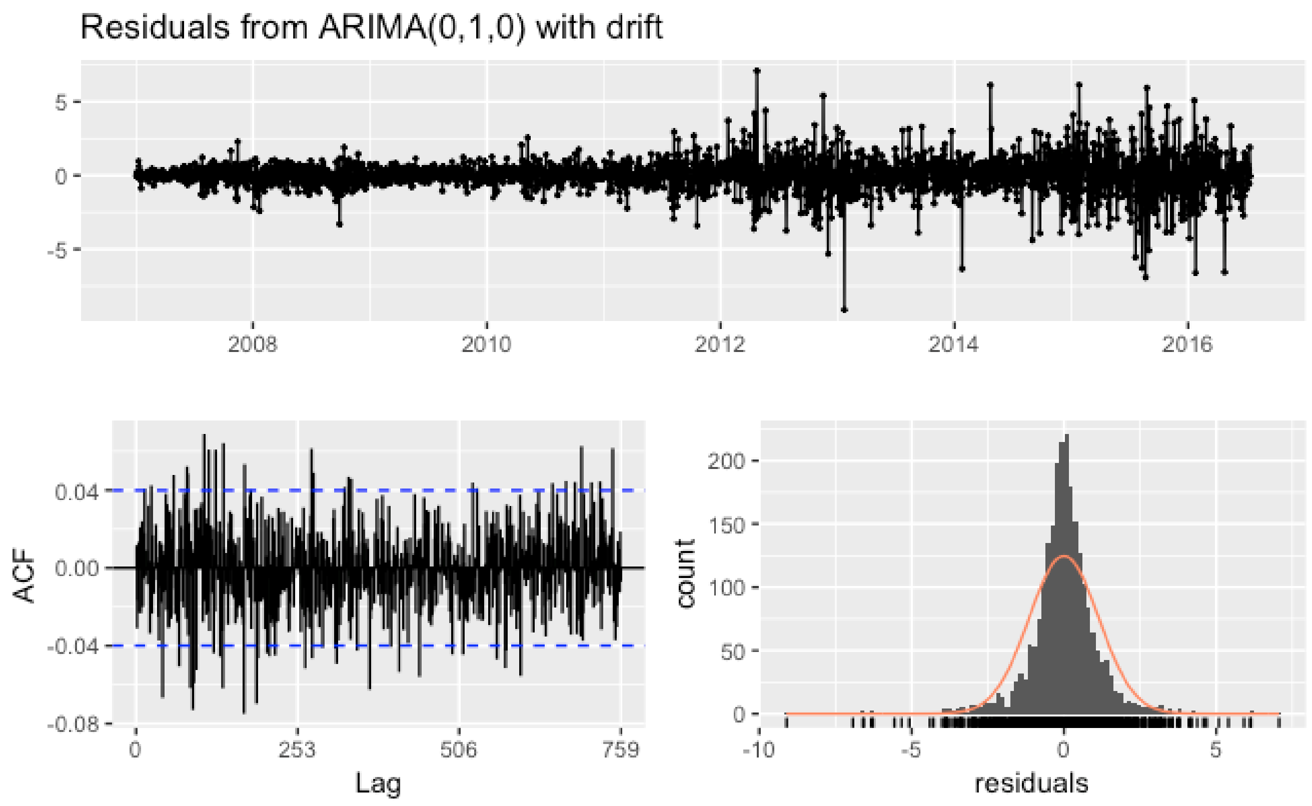
```
autoplot(appl_arima, series = "Forecast") +
    autolayer(appl_testing, series = "Test") +
    xlab("Year") + ylab("Closing Price of Apple stock") +
    ggtitle("Forecasts of Apple stock closing price using Arima(0,1,0)") +
    guides(colour = guide_legend(title = "Data"))
```

OUTPUT:



Check Residuals:
checkresiduals(appl_arima)

OUTPUT:
```
Ljung-Box test

data:  Residuals from ARIMA(0,1,0) with drift
Q* = 701.94, df = 481.4, p-value = 1.875e-10

Model df: 1.   Total lags used: 482.4
```



Compute forecast accuracy measures of ARIMA(0,1,0) model with drift:
accuracy (appl_arima, appl_testing)

OUTPUT:
```
                       ME      RMSE        MAE         MPE      MAPE       MASE       ACF1 Theil's U
Training set 4.948349e-06  1.144996  0.7546568 -0.03369398  1.473943 0.04009683 0.01164362        NA
Test set     4.876376e+01 56.040182 48.7780786 28.31563590 28.330391 2.59170285 0.99372814  22.11823
> |
```

CONCLUSION:

Which Model to choose?

Mean, Seasonal Naïve, Regression are rejected based on Residuals check.
For the remaining models we compared the RMSE of training data as well as test data set as depicted below:

For Training Set

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Mean	-1.108569e-15	35.40613	30.85103	-62.64105	93.14712	1.639193	0.9988489	NA
Naïve	0.03600521	1.145799	0.7558108	0.0652478	1.47613	0.04015814	0.01164376	NA
Seasonal Naive	10.35607	22.25247	18.82086	13.18451	32.88206	1.000000	0.9959151	NA
Drift	9.826522e-15	1.145233	0.7549649	-0.03374931	1.474513	0.0401132	0.01164376	NA
Regression	-5.613427e-16	12.01551	10.18912	-2.975144	26.32305	0.5413738	0.9952434	NA
Holt Linear	0.003056777	1.145438	0.7553488	-0.02641243	1.478918	0.0401336	0.01175102	NA
ARIMA	4.948349e-06	1.144996	0.7546568	-0.03369398	1.473943	0.04009683	0.01164362	NA

For Test Set

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Mean	9.931478e+01	104.83518	99.31478	60.88951	60.88951	5.276846	0.9946694	44.24462
Naïve	59.63733107	68.436841	59.6495035	34.6333753	34.64592	3.16932919	0.99466939	27.03928
Seasonal Naive	50.29479	61.4518	52.62289	28.26428	30.47388	2.795987	0.9907513	24.33266
Drift	4.876376e+01	56.040182	48.7780786	28.31563590	28.330391	2.5917028	0.99372814	22.11823
Regression	2.701277e+01	37.2450	30.99776	14.202686	17.86990	1.6469895	0.9928585	14.30352
Holt Linear	49.870360582	57.298729	49.8844639	28.95859209	28.973123	2.6504879	0.99385790	22.61763
ARIMA	4.876376e+01	56.040182	48.7780786	28.31563590	28.330391	2.59170285	0.99372814	22.11823

After comparing ***RMSE and complexity*** of ***Naïve, Drift, Holt Linear and ARIMA***, we've arrived on a conclusion that Drift Model is the most suitable one among all the other models which we have implemented as forecasting using Drift model is simpler to implement. Also, the ARIMA model suggested using an automated function auto.arima may not be optimal, so we will predict future values using Drift Model.

Predicting Future Apple Stock Price:

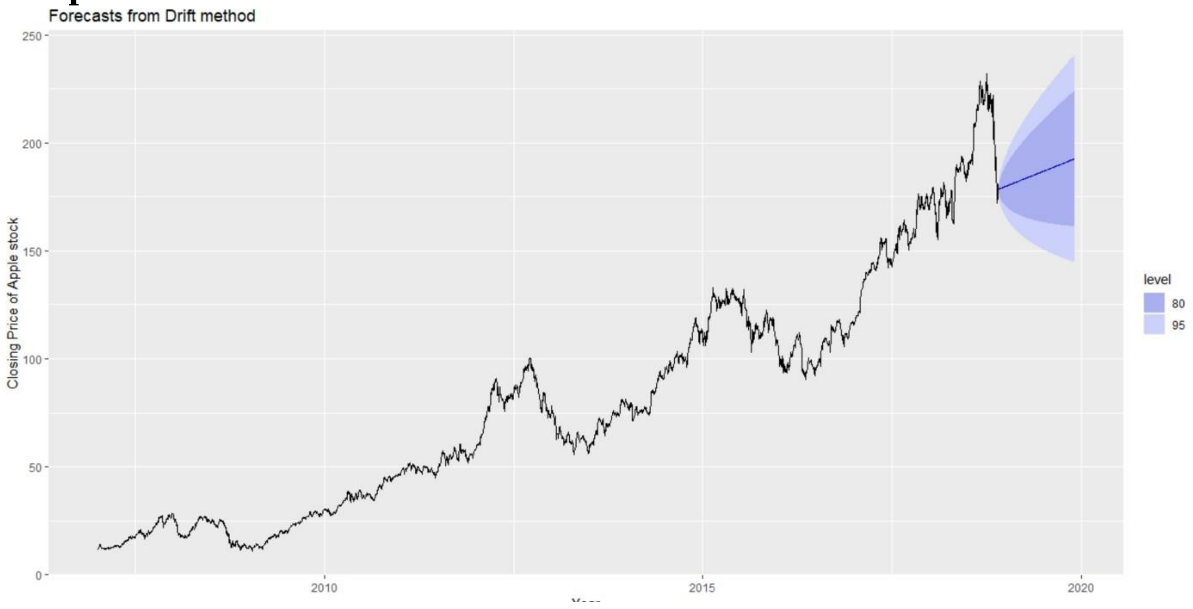
As we compared all the models, we found out that drift method is the best with good accuracy on test data:
Now we will forecast future values:
For that we will train model on the whole dataset:

```
future_drift <- rwf(appl_close_ts,h= 253, drift = TRUE)
```

Plotting Predictted Values:

```
autoplot(future_drift) +
 xlab("Year") + ylab("Closing Price of Apple stock") +
ggtitle("Forecasts from Drift method")
```

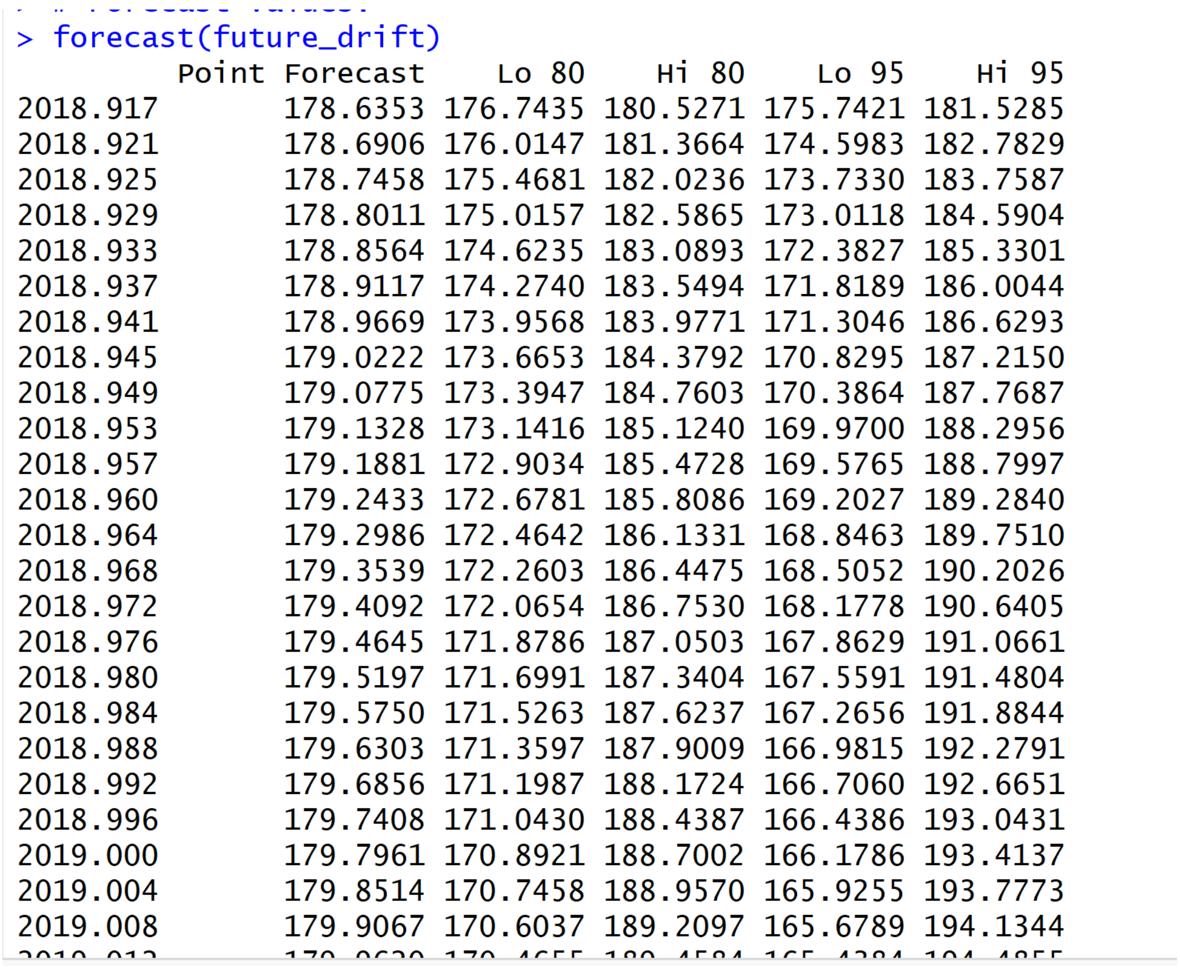
Output:



Forecast values:

forecast(future_drift)

Output:



```
> forecast(future_drift)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2018.917       178.6353 176.7435 180.5271 175.7421 181.5285
2018.921       178.6906 176.0147 181.3664 174.5983 182.7829
2018.925       178.7458 175.4681 182.0236 173.7330 183.7587
2018.929       178.8011 175.0157 182.5865 173.0118 184.5904
2018.933       178.8564 174.6235 183.0893 172.3827 185.3301
2018.937       178.9117 174.2740 183.5494 171.8189 186.0044
2018.941       178.9669 173.9568 183.9771 171.3046 186.6293
2018.945       179.0222 173.6653 184.3792 170.8295 187.2150
2018.949       179.0775 173.3947 184.7603 170.3864 187.7687
2018.953       179.1328 173.1416 185.1240 169.9700 188.2956
2018.957       179.1881 172.9034 185.4728 169.5765 188.7997
2018.960       179.2433 172.6781 185.8086 169.2027 189.2840
2018.964       179.2986 172.4642 186.1331 168.8463 189.7510
2018.968       179.3539 172.2603 186.4475 168.5052 190.2026
2018.972       179.4092 172.0654 186.7530 168.1778 190.6405
2018.976       179.4645 171.8786 187.0503 167.8629 191.0661
2018.980       179.5197 171.6991 187.3404 167.5591 191.4804
2018.984       179.5750 171.5263 187.6237 167.2656 191.8844
2018.988       179.6303 171.3597 187.9009 166.9815 192.2791
2018.992       179.6856 171.1987 188.1724 166.7060 192.6651
2018.996       179.7408 171.0430 188.4387 166.4386 193.0431
2019.000       179.7961 170.8921 188.7002 166.1786 193.4137
2019.004       179.8514 170.7458 188.9570 165.9255 193.7773
2019.008       179.9067 170.6037 189.2097 165.6789 194.1344
```

CHALLENGES:

- The data that we fetched from Quant Mod did not have uniform number of records for each year. Hence, we had to plug in values using the naïve model to make each year have equal number of records.
- Selection of model for prediction of apple stock was another challenge as Drift and Arima forecasting models showed identical accuracies.
- ARIMA model achieved through auto.arima function had a similar accuracy and a low AICc as compared to Holt model.
- auto.arima function may not be optimal as it is automated and we get the model as (0,1,0) with drift. Thus, when c=0 and d=1, the long-term forecasts will go to a non-zero constant. We chose DRIFT model to be the best fit for our data set since rwf with Drift is equivalent by definition to Arima(0,1,0).

REFERENCES:

[1] Kaggle:
https://www.kaggle.com/dgawlik/nyse

[2] Forecasting: Principles and Practice by Rob J Hyndman and George Athanasopoulos:
https://otexts.org/fpp2/

[3] Quamtmod:
https://www.quantmod.com/