2016

----------------------------------------------------------------

OPIM 5503 Data Analytics using R

TEAM -Make R great again

Danish Chappanga Thottathil

Siddharth Kajampady

Sindhushree Sonnad Matt

Mownika Chalichama

Yoganand Guttikonda

Project Report

Package quantmod

(Quantitative Financial Modelling Framework)

**Executive Summary**

Financial models play a key role in almost all major business decisions. They are useful tools that allow business options and risks to be evaluated in a cost-effective manner and help identify optimal solutions in evaluating financial returns. In this project we have introduced a package called quantmod, which is built specifically to analyze financial data, build charts, prepare models and forecast. With the help of Dow Jones Industrial Average (DJIA) stock index data, we would be demonstrating the features of quantmod package and also forecast closing stock prices for one of the top performing companies.

**Data Description**

The dataset consists of daily stock values for the 30 publicly owned companies listed in Dow Jones Industrial Average, for the duration 09/01/2015 through 09/30/2016. This data has been extracted from Yahoo Finance by using a function in quantmod package. The function returns the Opening, High, Low, Closing, Volume and Adjusted closing stock values of the 30 companies listed. However, for this analysis, we used just the time series daily data for the Closing stock values for all the companies.

**Getting Data into R**

The package quantmod present in CRAN library is installed and loaded into R using the below commands



Once the package is loaded, we load the data into R with the help of *getSymbols()* function. We include certain arguments in the function to specify the data that needs to be retrieved and the time duration. The code used in the function for our project is as below-

First, we create a vector containing the symbols of the 30 companies listed in DJIA



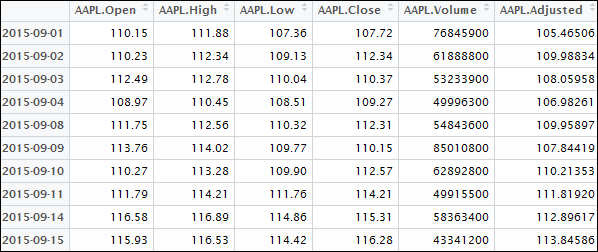
A symbol is a unique code for each company listed in finance websites. Here symbol AAPL refers to Apple, AXP for American Express Company, NKE for Nike and so on.

**getSymbol() function code:**



The first argument is the vector consisting of symbols, second arguments we specify the source to be Yahoo finance and would need the stock price data for the dates 2015-09-01 through 2016-09-30 (13months, 274 observations).

This function connects to yahoo finance and retrieves the stock price data for the symbols(companies) passed in the argument. This code creates an xts object (extensible time series) for each of the symbols and we get the data directly into R environment. The screenshot of the time series data is given below-



6 different stock price variants are returned for everyday from 2015-09-01 to 2016-09-30.

Open🡪 The price of the stock at the beginning of the day (Stock exchange open)

High🡪 Highest value of the stock for that day

Low🡪 Lowest value of the stock for that day

Close🡪 The price of the stock at the end of the day (Stock exchange close)

Volume🡪 The number of shares/contracts traded for that day

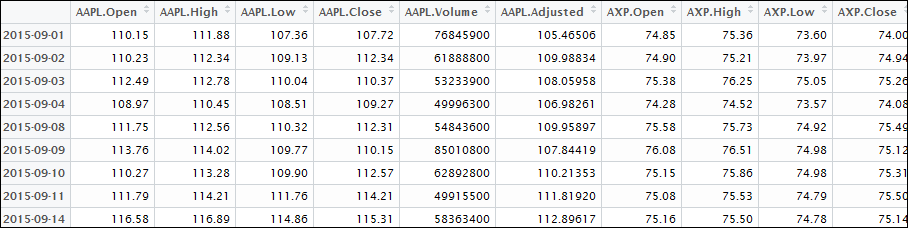
Adjusted🡪 Closing price of the stock which has been amended to include any changes prior to next day.

**Stock Price Dataframe**

The stock prices obtained for all the 30 companies are merged and stored in a dataframe. The code for doing this is given below



We perform a merge operation on the xts objects so as to have a single time series variable and all the stock price variants for different symbols in a single dataframe. Screenshot of the dataframe **DOWJO30data**



**Financial Charts**

Using quantmod functions, we can visualize financial data using the standard financial charting tools. The function chartSeries can plot financial time series data and also additional finance indicators.

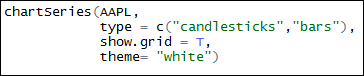
Suppose we want to analyze stock values for Apple Inc. The syntax for the function is given below:



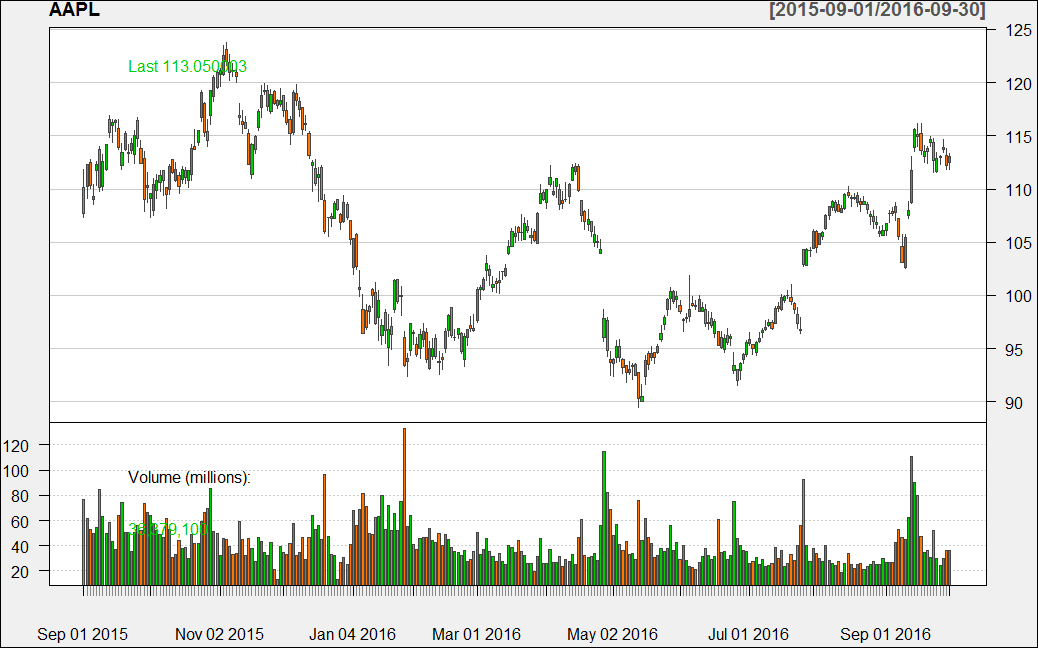
Since we have already downloaded stock price data for this symbol, we get the below chart-



By providing values to the function argument, we can change the chart features and theme



Here, we specify the type of chart to be candlesticks, showing gridlines and the overall theme as white.



We can add some finance indicators such as Bollinger bands and Exponential Moving Average bands to the chart.

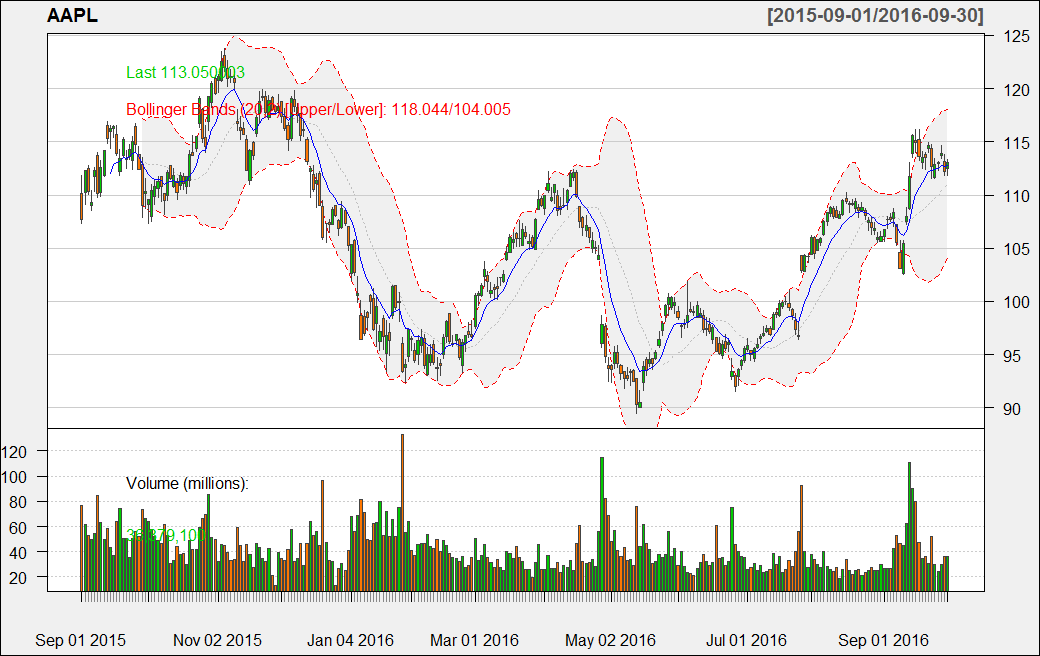
**Bollinger Band**- An indicator for volatility which is based on the standard deviation of the stock prices. The bands widen if volatility increases and narrows down when volatility decreases.

**Exponential Moving Average-** This moving average is used to identify the direction of the trend and define potential resistance levels.

To add these 2 indicators to the chart, we run the below code:



The chart with the Bollinger bands and EMA looks like below-



**Financial Modelling**

The stock price values for all the 30 companies have been merged and stored in a dataframe. Next, we would want to forecast the closing value of stock price for the companies. Forecasting for all 30 companies would not be feasible, hence we forecast stock price values for the top 5 performing companies. We have chosen metrics ‘Daily Return’, ‘Weekly Return’ and ‘Monthly Return’ and calculate the weighted moving average for each of the metric to order the performance of companies. The definition for these terms are listed below-

Daily Return🡪 The change in stock value on a daily basis. It is the gain or loss of a stock price value in a period of 1 day.

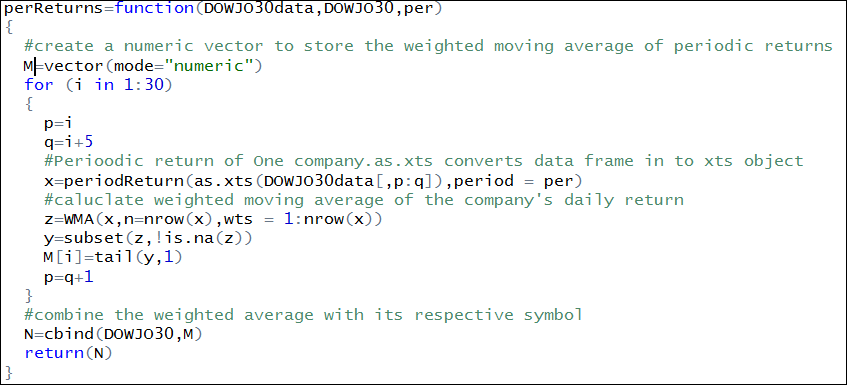
Weekly Return🡪 The change in stock value in a week Period-Gain or loss of stock price value in one week

Monthly Return-🡪The change in stock value in 1-month time Period-Gain or loss of price in a month.

Formula- **Daily Return= (Current Value – Last day’s value)/Current Value**

The same formula applies to monthly and daily returns (Instead of last day we have last week or last month). Using a custom function created by our team, we find the weighted moving average of the periodic returns

**Function for computing periodic returns (perReturns())**



We pass the argument values to the function as follows:

DAILY RETURN



WEEKLY RETURN



MONTHLY RETURN

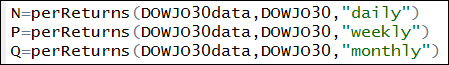


The first argument is the dataframe consisting of stock price values for all 30 Dow Jones IA companies.

Second argument is the vector consisting of symbols for all the 30 companies.

In the third argument we specify if we want to compute the average daily, weekly or monthly return for the companies.

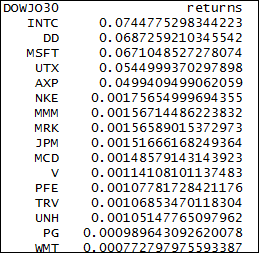
We then run the function for the three different returns and store it in different dataframes.



Next, we sort the dataframe in descending order so that those companies having higher values for returns are listed at the top. The code used for doing this is as follows:

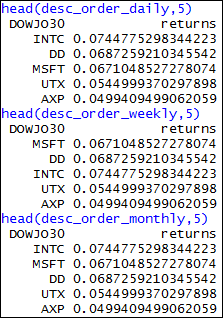


The sorted data frame will list companies with higher returns on the top. Below is a screenshot of the data frame created for daily returns. Two more data frames for weekly and monthly returns are created similarly:



As decided, we select the top 5 companies having the highest daily, weekly and monthly returns average using the following command. The head() function returns the observations from the beginning of the data frame.

We specify 5 in the argument to get the top 5 companies with high returns for stock prices.

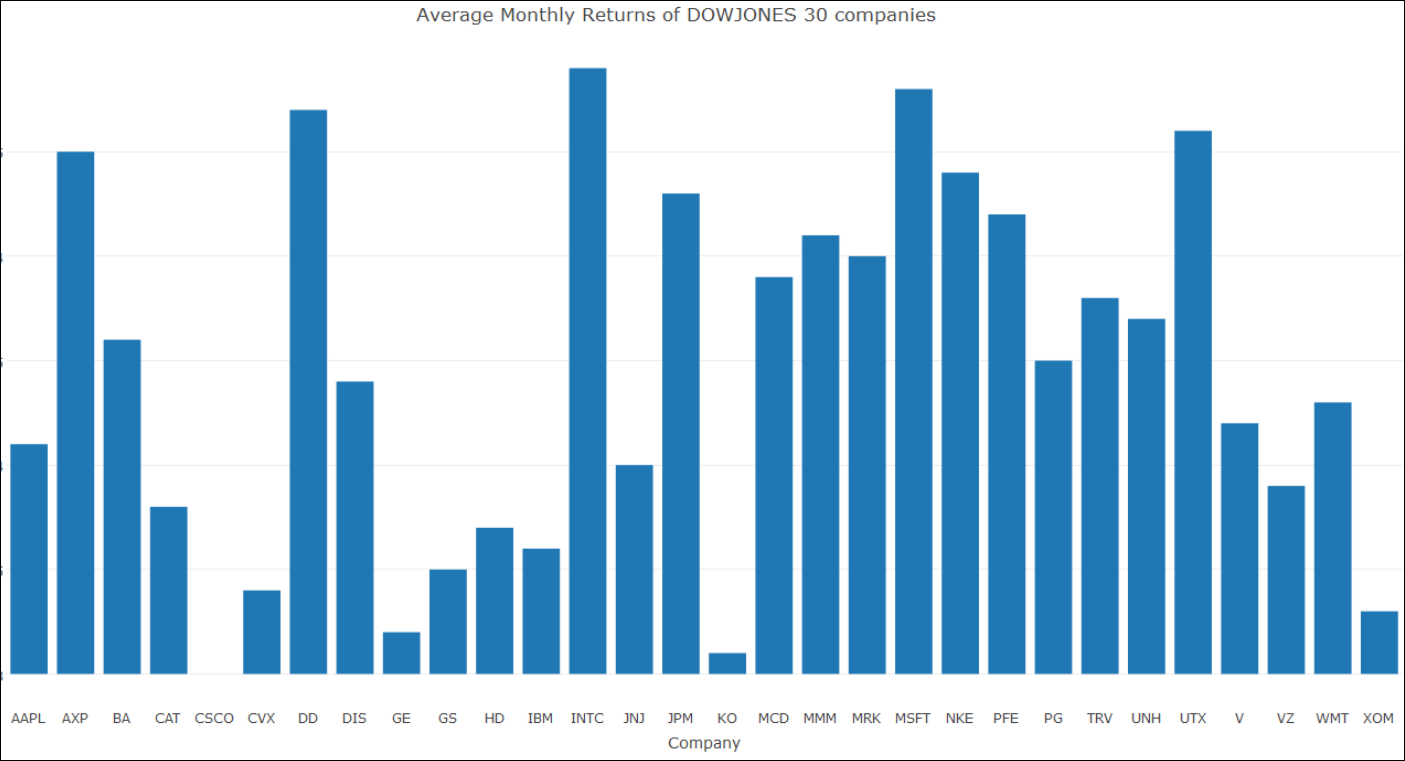


We can see that Intel Corporation(INTC), du pont de Nemours Company(DD), Microsoft Corporation(MSFT), United Technologies Corporation(UTX) and American Express Company(AXP) appear as the top 5 companies for all the average returns.

**GRAPHS**

Using the package plotly, we plot Weighted Moving Average of Daily, Weekly and Monthly Returns for DOWJONES30 companies:

Monthly returns:

****

Based on the graphs we select one company for which we will build a trading model. Since INTC has highest Weighted Average of Returns, we consider the company INTC (Intel Corporation).

**Building a trade model for Intel Corporation**

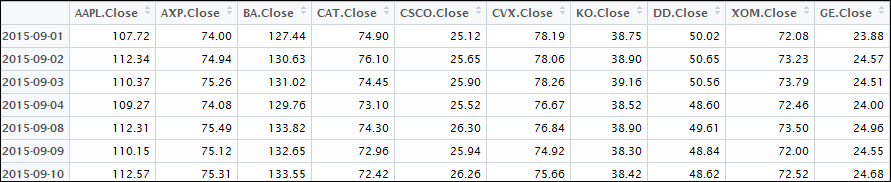
**Methodology:** Based on one-week historic stock prices and the closing prices of all the 30 companies, we forecaste the next day closing price for Intel Corporation

It is understood that the stock closing prices of all other companies in Dow Jones would impact the stock price of Intel Corportion, hence we would be including this factor in our forecast trading model.

By using Dplyr package, we select the closing price for all the 30 companies using the below code:



The data frame DOWJO30close looks as below-

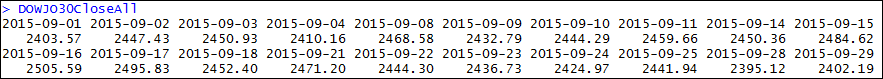


Further, we find the row wise sum of the closing prices of all the companies. We believe that this stock sum is a good indicator of the stock prices for Intel. Hence this is one of the predictor for the model.

The code for finding the sum of closing prices for 30 companies is given below-



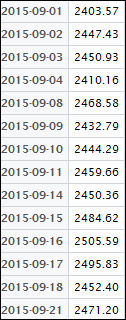
Once this is run, we get the daily sum total of closing stock prices of all companies –



For modelling purposes, we need to convert the xts object into a dataframe. We do this using code



The dataframe DOWJO30Closeexts consists of daily closing stock price sum as given below



**Function specifyModel()**

We include the model specifications, basically the formula which describes model parameters in this function. This will act as a reusable model specification function. The syntax is given below



We are forecasting the next day’s closing value of Intel corporation by regressing today’s closing value, the sum of closing stock values for all 30 companies and with a lag period of 7 days, which means we are considering historic stock price values for one week.

Response Variable🡪 Next(Cl(INTC))

Predictor 1🡪 Cl(INTC)

Predictor 2🡪 lag(Cl(INTC),1:7)

Predictor 3🡪 DOWJO30Closexts

**Function BuildModel()**

We then use the built-in function of quantmod to build a fitted model and forecast closing stock price value of Intel Corporation.

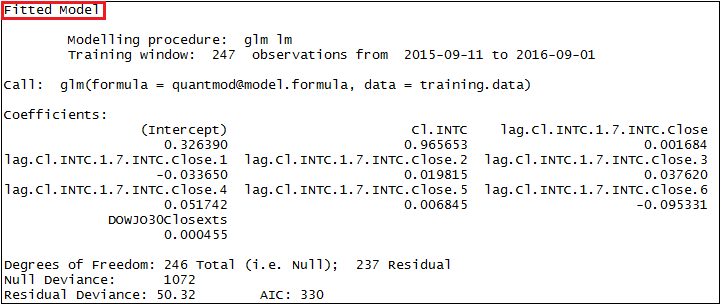
We have already the specified model parameters using the specifyModel() function. Hence we would call the variable model\_param in the BuildModel function and also specify the time duration for training data. The function is passed the arguments as below.



Different models like linear model, decision tree, random forest can be fitted by specifying the method.

For forecasting Intel stock price, we would be using the method= ‘glm’ which uses the Generalized Linear Model. A GLM model allows the outcome variable to have an error distribution, rather than a normal distribution. The training period data is for 12 months starting from 09/01/2015.

The model co-effecients and fit statistics are shown below-



Using these co-effecients, we obtain next day’s closing stock price value for Intel Corporation. The equation is given by

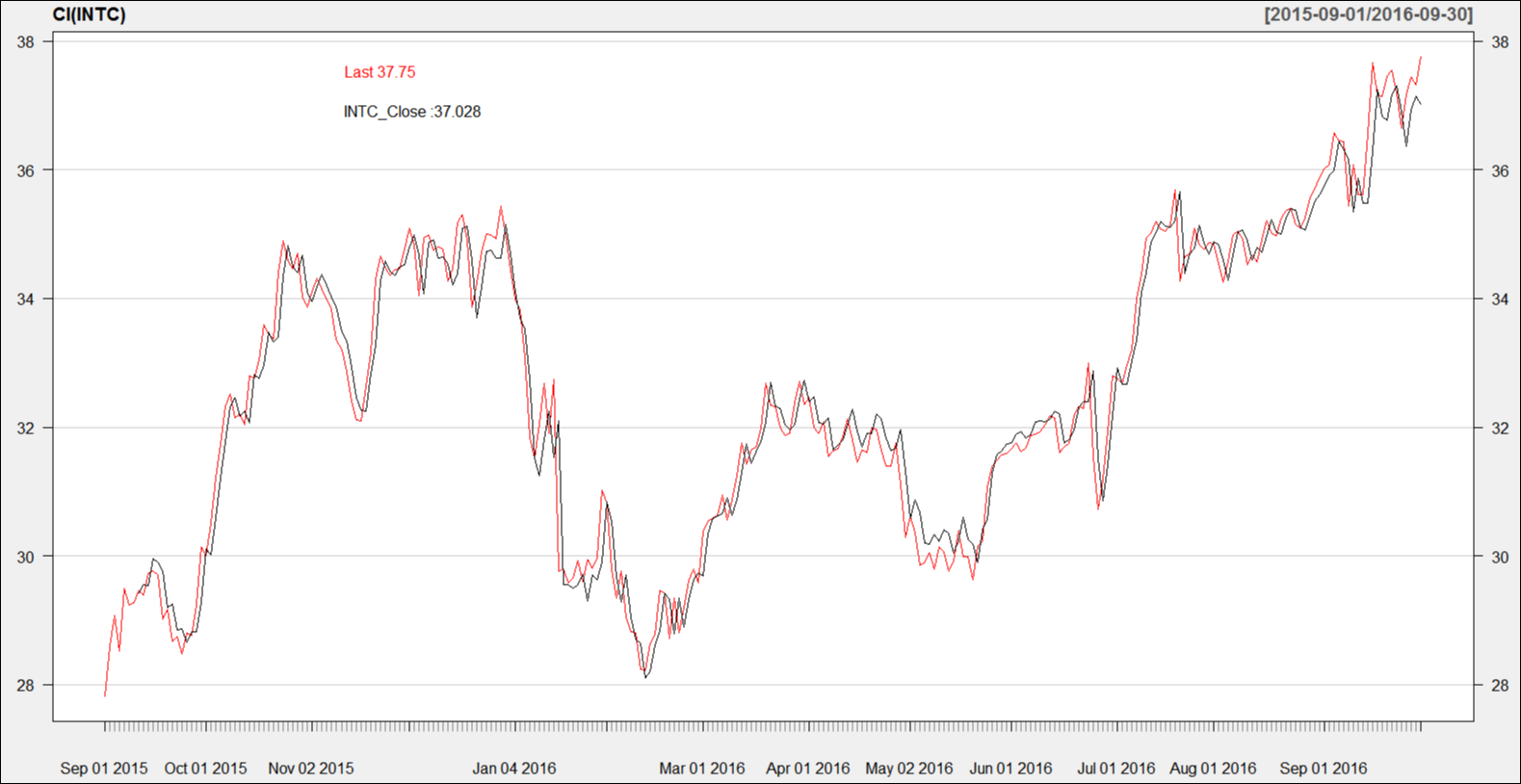


**Forecast Plot**

We used a function defined in the quantmod package (lineChart) to plot the current stock closing prices of Intel Corporation and the forecasted values obtained from the trade model. We use the code as shown below:



**Plot**



We see that the forecasted stock prices (Blue Line) closely follows the current stock prices of Intel corporation.

**Scope for improvement**

From the exploration of package quantmod done by our team, we noticed a few shortcomings that can further improve the usage of it. Stating below are few such areas of improvement:

* Quantmod build model function does not have new modelling techniques like boosting and bagging to talk about.
* New graphics for risk, return and significance analysis would give better insights.
* Addition of new data source methods to getsymbols like ODBC and additional database drivers.
* Addition of mechanisms to manage stored models.
* Addition of a function to explore random portfolios
* Results of Quantmod tools such as trademodel, modelReturns, modelRisk, modelAnalysis etc are difficult to understand and interpret. A detailed explanation would be better.
* Addition of a model significance function, to include Monte-Carlo simulation.

**References**

**Images**

<https://www.google.com/search?q=Dow+jones+industrial+average+finance&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjlqoK3gtTPAhUM6CYKHZUrBnMQ_AUICSgC&biw=1366&bih=638>

<http://www.quantmod.com/examples/charting/#chartseries>

**Study Material**

<http://www.quantmod.com/>

<http://www.quantmod.com/documentation/specifyModel.html>

<http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_averages>

<http://amunategui.github.io/wallstreet/>