Financial Options Valuation

11/27/16

Data Analytics using R

Contents

Executive Summary: 2
Data Overview: 2
Project approach: 3
a. Forecasting model- Stochastic trend - Random walk: 3
b. Shiny Application: Stock Forecasting 8
Conclusion: 11
References: 11

Executive Summary:

Individual investors have more investment options than they often realize. Options like” stock options” allow us to make money if the stock market is going up and it also causes loss if the stock market is going down. As the name suggests, options give you the option to buy or sell a security (stocks, extra-traded funds, indices, commodities etc.) at some point in the future. Our team has decided to forecast the stock market for the investors (our users) to make a wise decision. It is said that regardless of any technique, accurately forecasting stock market performance is more a matter of luck than technique. However, our team has decided to forecast the stock price using “Random walk” forecasting model.

- **Call Options:** It provide the holder the right to purchase an underlying asset at a specified price for a certain period of time. If the stock fails to meet the strike price before the expiration date, the option expires and becomes worthless. Investors buy calls when they think the share price of the underlying security will rise or sell a call if they think it will fall. Selling an option is also referred to as "writing" an option.
- **Put options:** It give the holder the right to sell an underlying asset at a specified price (the strike price). The seller (or writer) of the put option is obligated to buy the stock at the strike price. Put options can be exercised at any time before the option expires. Investors buy puts if they think the share price of the underlying stock will fall, or sell one if they think it will rise.

Random walk is a stock market theory that states that the past movement or direction of the price of a stock or overall market cannot be used to predict its future movement. Originally examined by Maurice Kendall in 1953, the theory states that stock price fluctuations are independent of each other and have the same probability distribution, but that over a period of time, prices maintain an upward trend. In short, random walk says that stocks take a random and unpredictable path. The chance of a stock's future price going up is the same as it going down. A follower of random walk believes it is impossible to outperform the market without assuming additional risk.

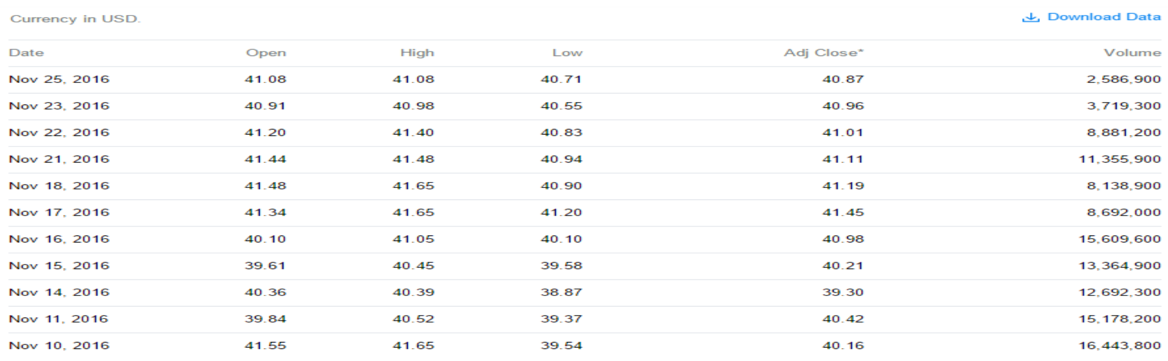
Data Overview:

Incorporated businesses sell shares of stock to investors to raise capital. Public stock exchanges provided a marketplace for the original purchasers of stock to sell their shares to other investors, and for investors to trade shares among themselves. Independent stock exchanges operate all over the world; the NYSE, AMEX and NASDAQ are the three stock exchanges located in the United States.

- **New York Stock Exchange(NYSE) -** The NYSE is the largest American stock exchange by volume.
- **American Stock Exchange(AMEX) -** The AMEX is a smaller exchange than NYSE, and it has always been favored by smaller companies which cannot meet NYSE's strict listing and reporting requirements.

- **National Association of Securities Dealers (NASDAQ) -** Unlike the other American exchanges, NASDAQ does not operate with a physical trading floor. NASDAQ trades take place solely online, increasing the exchange's cost efficiency and providing equal access to individual and institutional traders around the world.

  Historical data of Stock Exchange consist of below fields: High refers to the highest price the stock touched the very same day, Low refer to the lowest price the stock was traded on the same day ,Adj Close refers to the closing price of that particular stock and the volume refer to the number of share traded that day.


Currency in USD. Download Data

Date	Open	High	Low	Adj Close*	Volume
Nov 25, 2016	41.08	41.08	40.71	40.87	2,586,900
Nov 23, 2016	40.91	40.98	40.55	40.96	3,719,300
Nov 22, 2016	41.20	41.40	40.83	41.01	8,881,200
Nov 21, 2016	41.44	41.48	40.94	41.11	11,355,900
Nov 18, 2016	41.48	41.65	40.90	41.19	8,138,900
Nov 17, 2016	41.34	41.65	41.20	41.45	8,692,000
Nov 16, 2016	40.10	41.05	40.10	40.98	15,609,600
Nov 15, 2016	39.61	40.45	39.58	40.21	13,364,900
Nov 14, 2016	40.36	40.39	38.87	39.30	12,692,300
Nov 11, 2016	39.84	40.52	39.37	40.42	15,178,200
Nov 10, 2016	41.55	41.65	39.54	40.16	16,443,800

Project approach:

The objective of the project is not only to predict the future outcome but also to make it available to the public in order for them to understand future market. So, the first part of the project is to forecast the stock price and second part of the project is to create a application to make forecasts available to the user along with some stock markets insights

a. Forecasting model- Stochastic trend - Random walk:

Financial options valuation is mostly trend based such that most of the times they will depend on the previous day or previous weeks' value. By keeping this intuition, we have modeled the live streaming stocks options data using a lag value of 1(which states we are using yesterday's value for today's predictions). Primarily to compute this we used monte Carlo simulations of size n for each day's forecast. To start with the forecasting the default valuation we used is the current price of the stock to predict tomorrows price.

$$Stock\ Price = Current\ Stock\ Price * random\ normal\ distribution(probability, mean, std)$$

$$std = standard\ deviation\ of\ increase\ from\ lag1\ to\ present\ value\ over\ the\ past\ 365\ days$$
$$mean = \ mean\ increase\ from\ lag1\ to\ present\ value\ over\ the\ past\ 365\ days$$
$$probability = random\ probability\ between\ 0\ and\ 1\ using\ uniform\ distribution$$

This approach is called random walk where every step is dependent only on the previous step, which in our case stock price is dependent on the lag 1 stock price. To reduce the bias, we have introduced simulations with real world trend. Our approach build forecast for each period with n simulations, which we will send as a parameter to the model building function. Finally, we iterate through every day doing simulations.

As the financial options (call option) produces profits only when the determined stock target is met, we have added a function to compute the stock price increase compared to the forecasted price, when the actual stock forecast is better than the pre-determined value, will use the predicted increase or else we will use 0 for that point. This way we will see the mean increase for this date. By computing the number of times the actual stock is going more than the initially determined price we can decide whether to invest or not. The objective of the project is to forecast stock price. Model is built from base R functions such as random numbers and arithmetic operations. We also optimized the iterations in the simulated model to linear level of n, which reduced the computation required to power of n.

One outer function is built which is getting called from Shiny application. The name of main function is "callingRenderingStockForecast". This main function consists of other functions and the working flow of these functions are: **callingRenderingStockForecast--> prepareForecast--> stockForecast**

Overall "Portfolio.r" program contained the code for forecasting the stock prices using random walk model and the steps are:

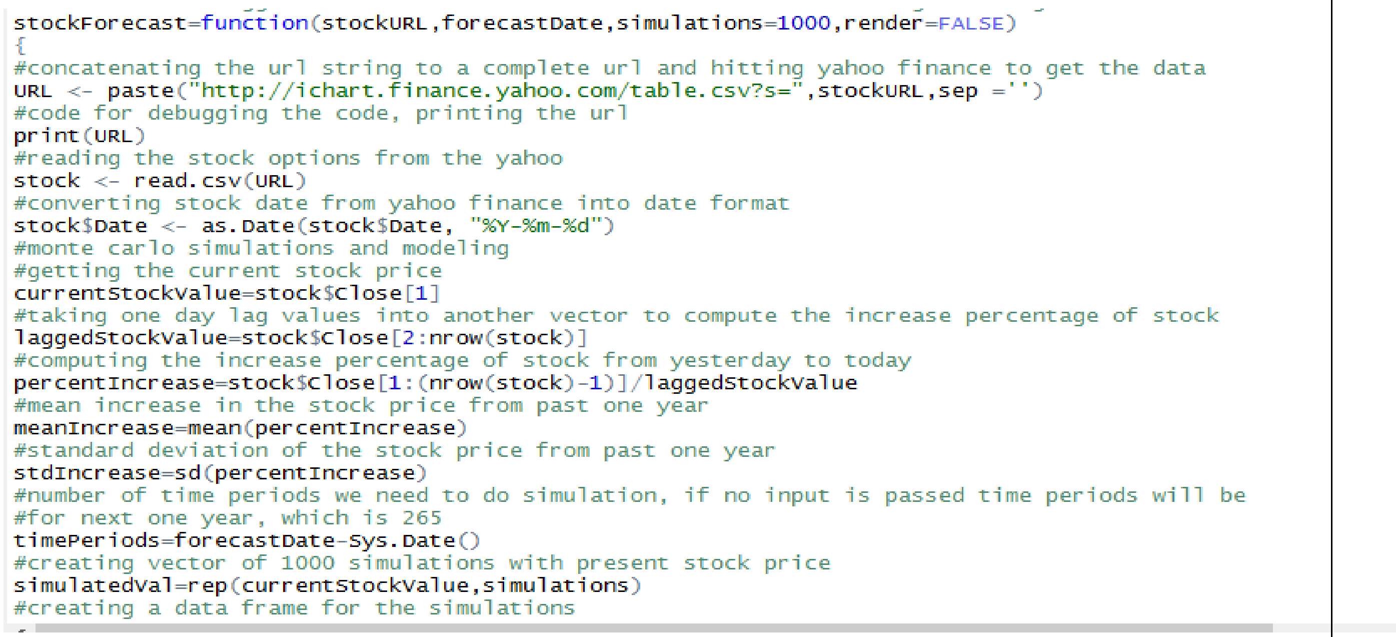
- Reading all ticker symbols i.e. stock option names and valuations from the yahoo finance
- Creating a function "stockForecast" where our final model is built and the steps involved in our forecasting are mentioned below, this function has 4 parameters stockURL, forecastDate, simulations, & render

stockURL	stock url consists of the url to get data from yahoo finance, this has stock option symbols, from date for forecasting, and to date to be considered (by default we are considering past one year data for predicting the future values)
forecastDate	This is the date for which the user is aiming to buy a call option, based on this we forecast the respective stock to check if the call option is valuable or not
simulations	Number of simulations required for computing each day's forecasts from random walk model by default we are using 1000 simulations
render	it is a toggle button for functions default Stocks and calling Rendering Stock Forecast

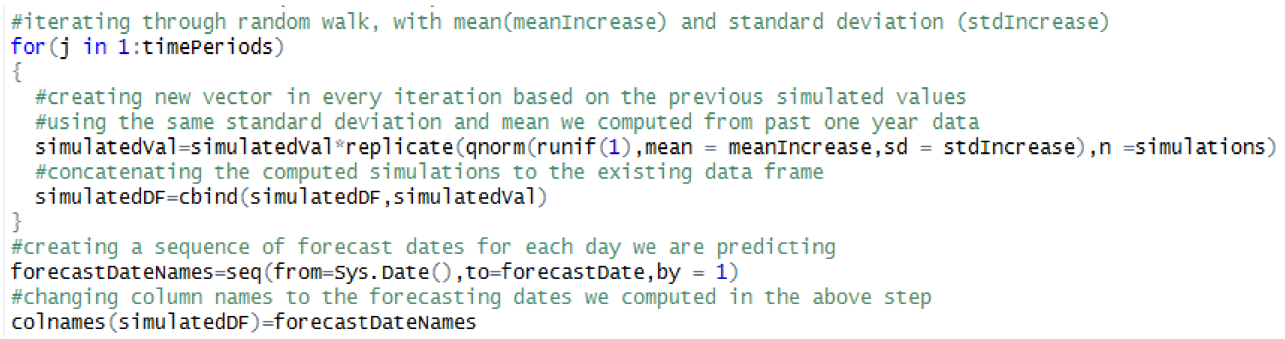
Function steps for Stock Forecast:

1. Concatenating the url string to a complete url and hitting yahoo finance to get the data
2. Reading the stock options from the yahoo
3. Converting stock date from yahoo finance into date format
4. Monte Carlo simulations and modeling
5. Getting the current stock price
6. Taking one day lag values into another vector to compute the increase percentage of stock
7. Computing the increase percentage of stock from yesterday to today
8. Store mean increase and SD in the stock price from past one year

9. Store number of time periods we need to do simulation, if no input is passed time periods will be for next one year, which is 365
10. Creating vector of 1000 simulations with present stock price which acts as default starting price for the forecasts
11. Creating a data frame of the simulations of dimensions (simulations * time period's)
12. Forecasted with random walk model, with mean (mean change in the price from past on year) and standard deviation (standard deviation). This iteration will be creating new vector in every iteration based on the previous simulated values using the same standard deviation and mean we computed from past one year data
13. We will do the above step recursively for all the time periods and save the forecasts to the data frame.
14. Creating a sequence of forecast dates for each day we are predicting to add the as column names for the matrix

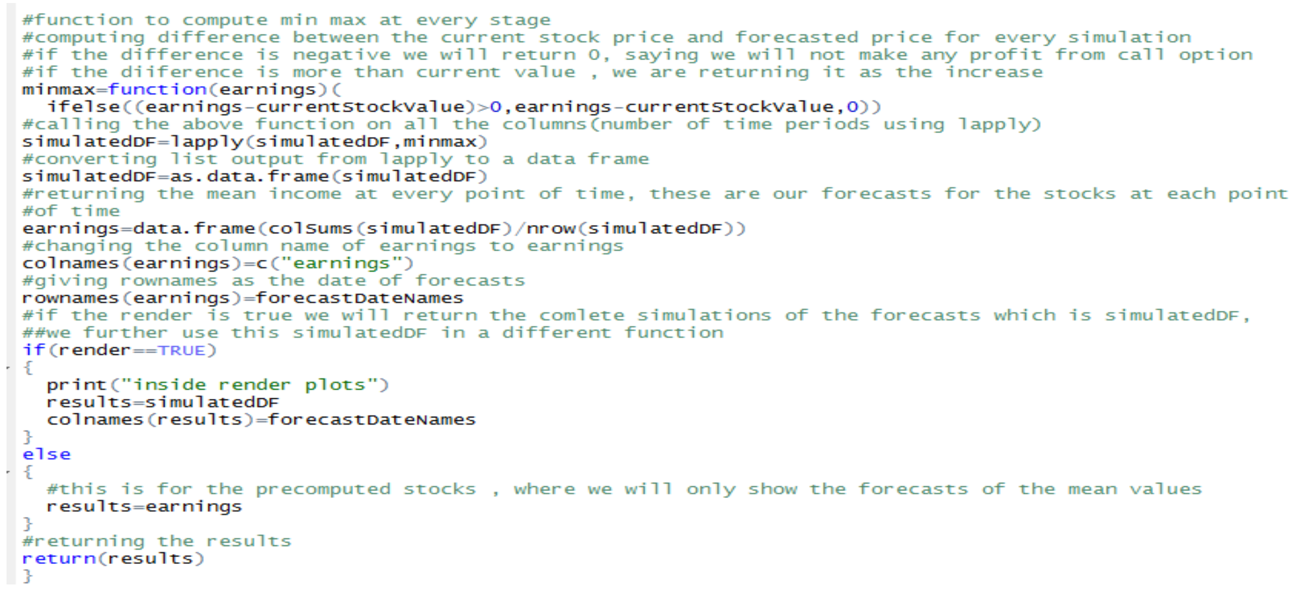


```
stockForecast=function(stockURL,forecastDate,simulations=1000,render=FALSE)
{
#concatenating the url string to a complete url and hitting yahoo finance to get the data
URL <- paste("http://ichart.finance.yahoo.com/table.csv?s=",stockURL,sep ='')
#code for debugging the code, printing the url
print(URL)
#reading the stock options from the yahoo
stock <- read.csv(URL)
#converting stock date from yahoo finance into date format
stock$Date <- as.Date(stock$Date, "%Y-%m-%d")
#monte carlo simulations and modeling
#getting the current stock price
currentStockValue=stock$Close[1]
#taking one day lag values into another vector to compute the increase percentage of stock
laggedStockValue=stock$Close[2:nrow(stock)]
#computing the increase percentage of stock from yesterday to today
percentIncrease=stock$Close[1:(nrow(stock)-1)]/laggedStockValue
#mean increase in the stock price from past one year
meanIncrease=mean(percentIncrease)
#standard deviation of the stock price from past one year
stdIncrease=sd(percentIncrease)
#number of time periods we need to do simulation, if no input is passed time periods will be
#for next one year, which is 265
timePeriods=forecastDate-Sys.Date()
#creating vector of 1000 simulations with present stock price
simulatedVal=rep(currentStockValue,simulations)
#creating a data frame for the simulations
```



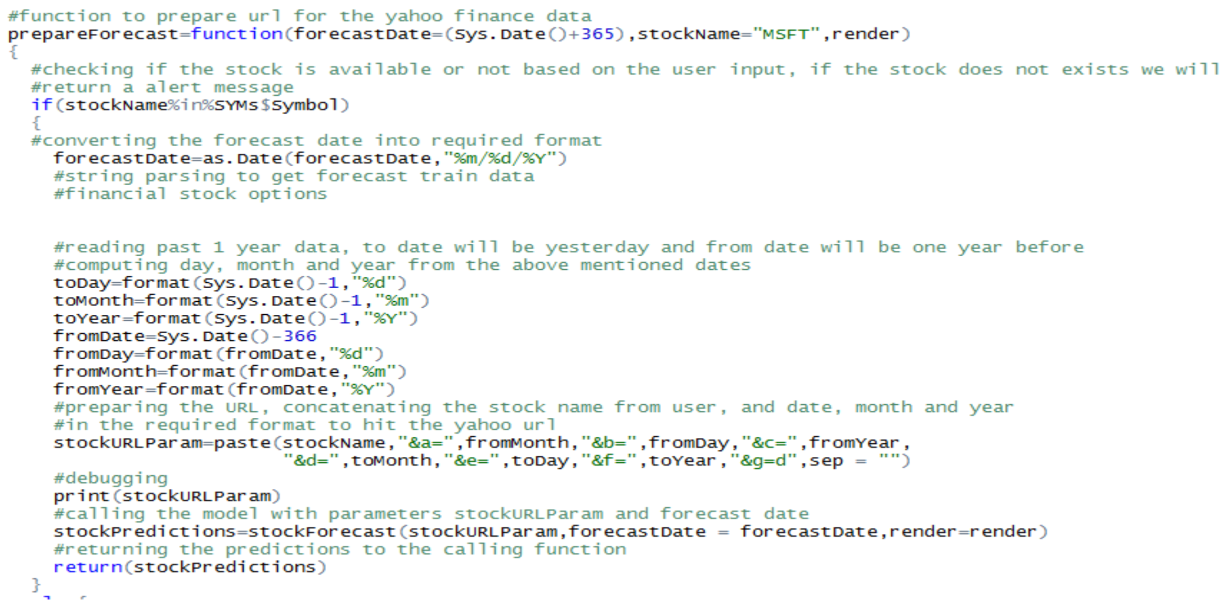
```
#iterating through random walk, with mean(meanIncrease) and standard deviation (stdIncrease)
for(j in 1:timePeriods)
{
  #creating new vector in every iteration based on the previous simulated values
  #using the same standard deviation and mean we computed from past one year data
  simulatedVal=simulatedVal*replicate(qnorm(runif(1),mean = meanIncrease,sd = stdIncrease),n =simulations)
  #concatenating the computed simulations to the existing data frame
  simulatedDF=cbind(simulatedDF,simulatedVal)
}
#creating a sequence of forecast dates for each day we are predicting
forecastDateNames=seq(from=Sys.Date(),to=forecastDate,by = 1)
#changing column names to the forecasting dates we computed in the above step
colnames(simulatedDF)=forecastDateNames
```

- **Function steps for MinMax :** to compute the difference between the current stock price and forecasted price for every simulation. This function computes min max at every stage
  1. If the difference is negative we will return 0, saying we will not make any profit from call option.
  2. if the difference is more than current value, we are returning it as the increase



```
#function to compute min max at every stage
#computing difference between the current stock price and forecasted price for every simulation
#if the difference is negative we will return 0, saying we will not make any profit from call option
#if the diiference is more than current value , we are returning it as the increase
minmax=function(earnings)(
  ifelse((earnings-currentStockValue)>0,earnings-currentStockValue,0))
#calling the above function on all the columns(number of time periods using lapply)
simulatedDF=lapply(simulatedDF,minmax)
#converting list output from lapply to a data frame
simulatedDF=as.data.frame(simulatedDF)
#returning the mean income at every point of time, these are our forecasts for the stocks at each point
#of time
earnings=data.frame(colSums(simulatedDF)/nrow(simulatedDF))
#changing the column name of earnings to earnings
colnames(earnings)=c("earnings")
#giving rownames as the date of forecasts
rownames(earnings)=forecastDateNames
#if the render is true we will return the comlete simulations of the forecasts which is simulatedDF,
##we further use this simulatedDF in a different function
if(render==TRUE)
{
  print("inside render plots")
  results=simulatedDF
  colnames(results)=forecastDateNames
}
else
{
  #this is for the precomputed stocks , where we will only show the forecasts of the mean values
  results=earnings
}
#returning the results
return(results)
}
```

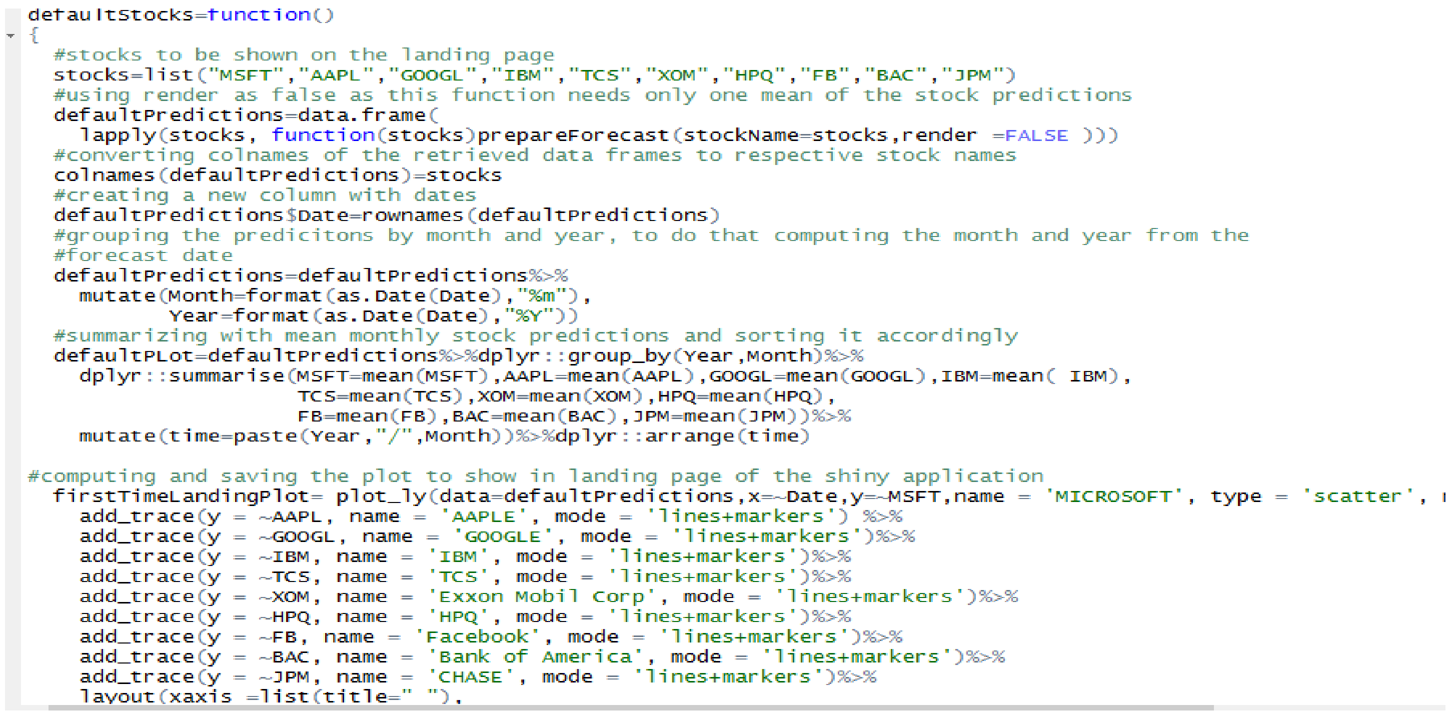
- **Function steps for prepareForecast:** To prepare url for the yahoo finance data. Steps of this function is as follows:
    1. Checking if the stock is available or not based on the user input, if the stock does not exist we will return an alert message
    2. Converting the forecast date into required format
    3. Reading past 1 year data, to date will be yesterday and from date will be one year before computing day, month and year from the above-mentioned dates
    4. Preparing the URL, concatenating the stock name from user, and date, month and year in the required format to hit the yahoo url
    5. Calling the “stockPredictions “model with parameters stockURLParam and forecast date
    6. Return the predictions to the calling function else show message that “No stock registered on the name of<given name>.



```
#function to prepare url for the yahoo finance data
prepareForecast=function(forecastDate=(Sys.Date()+365),stockName="MSFT",render)
{
  #checking if the stock is available or not based on the user input, if the stock does not exists we will
  #return a alert message
  if(stockName%in%SYMs$Symbol)
  {
  #converting the forecast date into required format
    forecastDate=as.Date(forecastDate,"%m/%d/%Y")
    #string parsing to get forecast train data
    #financial stock options


    #reading past 1 year data, to date will be yesterday and from date will be one year before
    #computing day, month and year from the above mentioned dates
    toDay=format(Sys.Date()-1,"%d")
    toMonth=format(Sys.Date()-1,"%m")
    toYear=format(Sys.Date()-1,"%Y")
    fromDate=Sys.Date()-366
    fromDay=format(fromDate,"%d")
    fromMonth=format(fromDate,"%m")
    fromYear=format(fromDate,"%Y")
    #preparing the URL, concatenating the stock name from user, and date, month and year
    #in the required format to hit the yahoo url
    stockURLParam=paste(stockName,"&a=",fromMonth,"&b=",fromDay,"&c=",fromYear,
                        "&d=",toMonth,"&e=",toDay,"&f=",toYear,"&g=d",sep = "")
    #debugging
    print(stockURLParam)
    #calling the model with parameters stockURLParam and forecast date
    stockPredictions=stockForecast(stockURLParam,forecastDate = forecastDate,render=render)
    #returning the predictions to the calling function
    return(stockPredictions)
  }
```

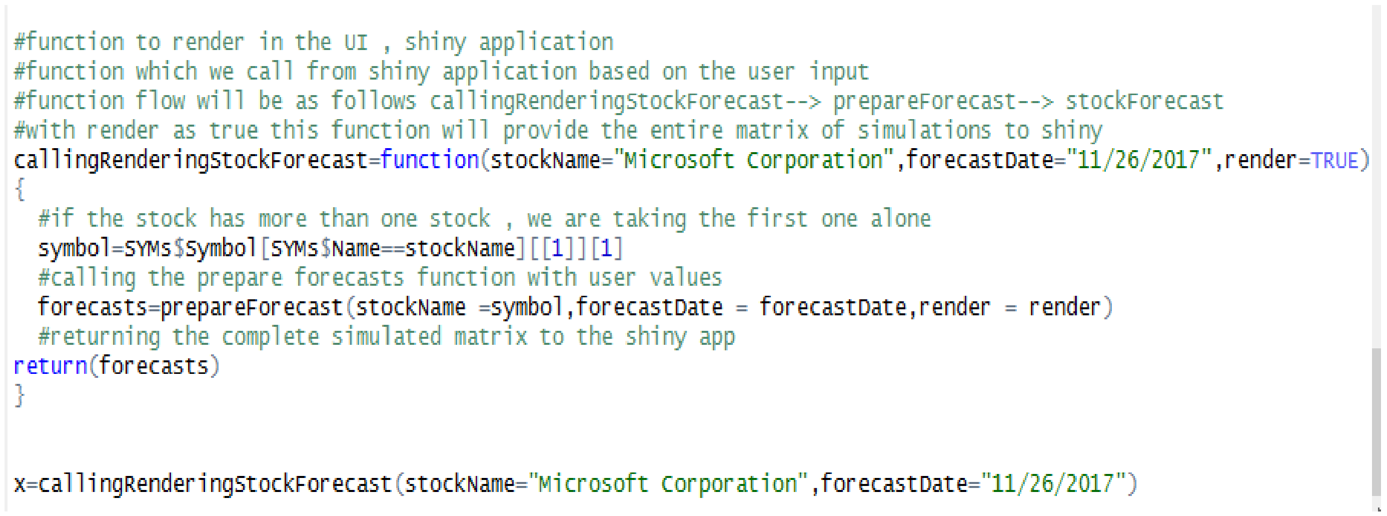
- **Function steps for defaultStocks:** to compute the forecasts for top 10 stocks. Here are the steps:
    1. Consider following list of stocks: MSFT, AAPL, GOOGL, IBM, TCS, XOM, HPQ, FB, BAC, JPM
    2. Generating forecast for the stocks
    3. Grouping the predictions by month and year
    4. Summarizing with mean monthly stock predictions and sorting it accordingly
    5. Computing and saving the plot to show in landing page of the shiny application
    6. Returning the plot to the UI



```
defaultStocks=function()
{
  #stocks to be shown on the landing page
  stocks=list("MSFT","AAPL","GOOGL","IBM","TCS","XOM","HPQ","FB","BAC","JPM")
  #using render as false as this function needs only one mean of the stock predictions
  defaultPredictions=data.frame(
    lapply(stocks, function(stocks)prepareForecast(stockName=stocks,render =FALSE )))
  #converting colnames of the retrieved data frames to respective stock names
  colnames(defaultPredictions)=stocks
  #creating a new column with dates
  defaultPredictions$Date=rownames(defaultPredictions)
  #grouping the predicitons by month and year, to do that computing the month and year from the
  #forecast date
  defaultPredictions=defaultPredictions%>%
    mutate(Month=format(as.Date(Date),"%m"),
           Year=format(as.Date(Date),"%Y"))
  #summarizing with mean monthly stock predictions and sorting it accordingly
  defaultPLot=defaultPredictions%>%dplyr::group_by(Year,Month)%>%
    dplyr::summarise(MSFT=mean(MSFT),AAPL=mean(AAPL),GOOGL=mean(GOOGL),IBM=mean( IBM),
                     TCS=mean(TCS),XOM=mean(XOM),HPQ=mean(HPQ),
                     FB=mean(FB),BAC=mean(BAC),JPM=mean(JPM))%>%
    mutate(time=paste(Year,"/",Month))%>%dplyr::arrange(time)

#computing and saving the plot to show in landing page of the shiny application
  firstTimeLandingPlot= plot_ly(data=defaultPredictions,x=~Date,y=~MSFT,name = 'MICROSOFT', type = 'scatter', 
    add_trace(y = ~AAPL, name = 'AAPLE', mode = 'lines+markers') %>%
    add_trace(y = ~GOOGL, name = 'GOOGLE', mode = 'lines+markers')%>%
    add_trace(y = ~IBM, name = 'IBM', mode = 'lines+markers')%>%
    add_trace(y = ~TCS, name = 'TCS', mode = 'lines+markers')%>%
    add_trace(y = ~XOM, name = 'Exxon Mobil Corp', mode = 'lines+markers')%>%
    add_trace(y = ~HPQ, name = 'HPQ', mode = 'lines+markers')%>%
    add_trace(y = ~FB, name = 'Facebook', mode = 'lines+markers')%>%
    add_trace(y = ~BAC, name = 'Bank of America', mode = 'lines+markers')%>%
    add_trace(y = ~JPM, name = 'CHASE', mode = 'lines+markers')%>%
    layout(xaxis =list(title=" "),
```

- At the end function "**callingRenderingStockForecast**" is created to combine all other function logic. This final function will be called from stock market forecasting application.



```
#function to render in the UI , shiny application
#function which we call from shiny application based on the user input
#function flow will be as follows callingRenderingStockForecast--> prepareForecast--> stockForecast
#with render as true this function will provide the entire matrix of simulations to shiny
callingRenderingStockForecast=function(stockName="Microsoft Corporation",forecastDate="11/26/2017",render=TRUE)
{
  #if the stock has more than one stock , we are taking the first one alone
  symbol=SYMs$Symbol[SYMs$Name==stockName][[1]][1]
  #calling the prepare forecasts function with user values
  forecasts=prepareForecast(stockName =symbol,forecastDate = forecastDate,render = render)
  #returning the complete simulated matrix to the shiny app
return(forecasts)
}


x=callingRenderingStockForecast(stockName="Microsoft Corporation",forecastDate="11/26/2017")
```

b. Shiny Application: Stock Forecasting

- Shiny is an R package that makes it easy to build interactive web applications (apps) straight from R. Shiny applications are automatically "live" in the same way that spreadsheets are live. Outputs change instantly as users modify inputs, without requiring a reload of the browser. Shiny app has three components:

<u>1.User interface: </u>Packages used to build a shiny application are given below:

```
library(shiny)
library(shinydashboard)
library(shinyjs)
```

A shiny dashboard has three parts: a dashboardheader, a dashboardsidebar, and a dashboardbody. The following is the possible UI for a dashboard page:

```
ui=dashboardPage(
  # Application page heading
  dashboardHeader(title = "Financial Options Valuation"),
  dashboardSidebar(disable = T),
  dashboardBody(
  fluidPage(sidebarLayout(fluid = TRUE,
```

The first function is the fluidpage() holds the page layout. Each page will have sidebarpanel() and mainpanel() layout. Sidebarpanel() will contain input buttons like select button, dropdown button slider button, text/numeric input button etc.. Each button can be arranged as rows/column inside sidebar. The command for sidebar row will be as below:

```
 sidebarPanel(
   fluidRow(column(6,dateInput('date',label = 'Forecast Date', value = Sys.Date()+365,start=Sys.Date()+1)),
        column(6,selectInput('ExchangeValue', 'Exchange', c(unique(SYMs$Exchange))))),
        #writing the ui elements
        uiOutput('Sector'),
        uiOutput("Industry"),
        uiOutput("Stock"),
```

The columns are mapped as:

```
column(6,dateInput('date',label = 'Forecast Date', value = Sys.Date()+365,start=Sys.Date()+1)),
column(6,selectInput('ExchangeValue', 'Exchange', c(unique(SYMs$Exchange))))),
```

Each column will hold width value 1 to 12. Above screen shot shows that button for gender is holding 1 unit place in the row. Similarly mainPanel() can also be divided into rows and expected outputs can be showed in the output panel.

```
mainPanel(plotlyOutput("defaultStocks")),
#ui calender input
sidebarPanel(
  fluidRow(column(6,dateInput('date',label = 'Forecast Date', value = Sys.Date()+365,start=Sys.Date()+1)),
           column(6,selectInput('ExchangeValue', 'Exchange', c(unique(SYMs$Exchange))))),
       #writing the ui elements
       uiOutput('Sector'),
       uiOutput("Industry"),
       uiOutput("Stock"),
       fluidRow(column(6,selectInput("predictedIncrease","Option Value",callOptions)),br(),
      column(6,actionButton("button", "Show"))))),
fluidRow(
        column(6,plotlyOutput("densityPlot")),br(),br(),
        column(6,
              div(style = 'overflow-y: scroll',dataTableOutput("table"))))
```

<u>2. Server: </u> The server code will act as a function which will capture input values from UI and provide output to UI. Here is the sample code for server: Selection screen for server consist of Forecast Date, Exchange value such as AMEX,NASDAQ,NYSE. We have provided the Sector, Industry and stock name corresponding to Stock Exchange.

```
#server design

#calling defaults stocks plot
defaultStocksPlot=defaultStocks()

server=function(input, output){
  output$Sector = renderUI({
    selectInput('Sector', 'Sector',unique(SYMs$Sector[SYMs$Exchange==input$ExchangeValue]))
  })
  output$Industry = renderUI({
    selectInput('Industry', 'Industry Name',unique(SYMs$Industry[SYMs$Sector==input$Sector]))
  })
  output$Stock    <- renderUI({ selectInput( "Stock",'stock name',
                                             unique(SYMs$Name[SYMs$Industry==input$Industry]))})
  observe({
    if (input$button == 0)
    {
      output$defaultStocks=renderPlotly(defaultStocksPlot)

    }})
```

The output fields mention in the UI – main panel can be accessed in the server code. By rendertext/renderimage keyword all imputations can be made available in the output page.

<u>3.Running application:</u> To call the application, the following code is used:

```
#calling the application
shinyApp(ui=ui,server=server)
```

Designed application takes 6 input variables: Forecast Date, Exchange, Sector, Industry Name, Stock Name and Option Value. Let us consider the following inputs on our shiny financial option dashboard:

Forecasted Date: 11-27-2017

Exchange: NASDAQ

Sector: Technology

Industry Name: Computer Manufacturing

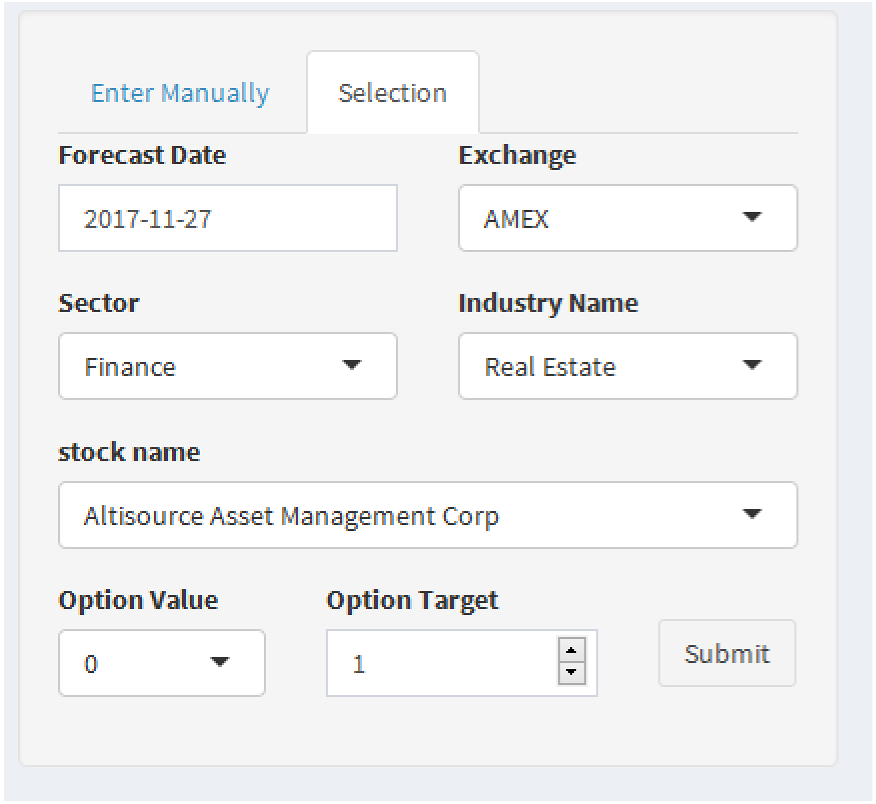
Stock Name: Apple Inc.

Option Value: 1

Option Target: Target of the Stock to a Forecasting Date, if it is zero we will take current price as default and forecast the increase

Enter Manually-Menu Bar: To Select or Enter the Stock Name and Submit It from Selection Menu

Selection Menu Bar: A Normal Way of Filtering the Stocks and Forecasting

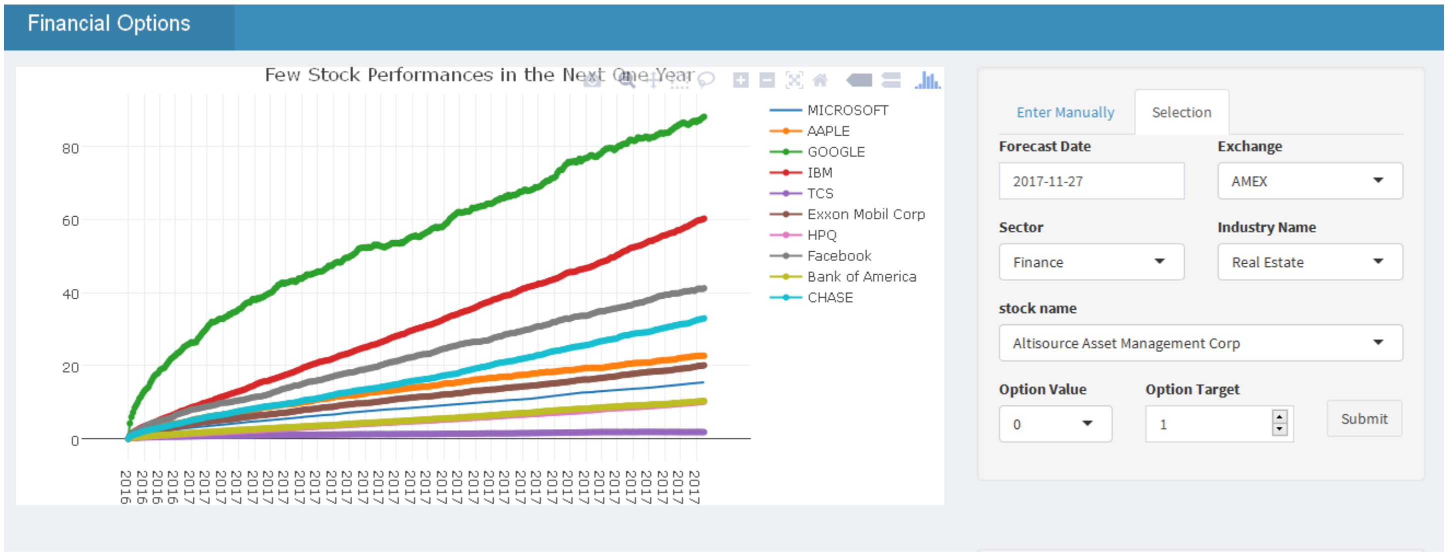


Enter Manually | Selection

Forecast Date	2017-11-27
Exchange	AMEX
Sector	Finance
Industry Name	Real Estate
stock name	Altisource Asset Management Corp
Option Value	0
Option Target	1

Submit

Below is the screenshot of the shiny dashboard of Financial options valuation where the left side shows the increase in value of few stocks which are listed beside the graph in the next one year. We see that the rise in stock value for google would be approximately 84.794 on 11-27-2017, which is the highest as compared to all others.



Enter Manually | Selection

Forecast Date	2017-11-27
Exchange	AMEX
Sector	Finance
Industry Name	Real Estate
stock name	Altisource Asset Management Corp
Option Value	0
Option Target	1

Submit

Users: This application takes the above inputs and shows the increase in stock value for that stock name, density plot of distribution of earnings on the given forecasted date and the table containing the forecasted increase in the stock value for each day until the input date specified. This application will also show the possible risk of investing in the stocks by knowing the ratio of profit to call option value given in the density plot. If the ratio of profit to call option value > 10, then it is considered safe to invest in that stock. For Apple Inc., the ratio is 21.76, which is why it is viable to invest for Apple Inc.

Conclusion:

1. A simple but powerful tool for evaluating the financial options before investing, we can easily find the options which are undervalued and invest into them.
2. We can effectively manage our options investments in wide variety of sector's and industries using this application, this will potentially reduce the risk involved.
3. Model is updated in every instance which makes it more desirable than a model which is pre-built and implemented in real-time.
4. Mostly a simulation of the trend, which makes the number of assumptions in the stock prices to be minimal.

References:

http://www.investopedia.com/exam-guide/cfa-level-1/derivatives/options-calls-puts.asp

http://www.forbes.com/2006/08/23/investools-options-ge-in_wh_0823investools_inl.html

http://www.investinganswers.com/financial-dictionary/stock-market/random-walk-theory-907

http://www.investopedia.com/terms/s/strikeprice.asp?lgl=no-infinite