**Sentimental Analysis in R:**

**Software = R Studio**

**Objective of the project:**

Fetch the stock news, price and ticker from the news website eg Investing.com, Blumberg / twitter etc.

Analyze the sentiments and predict the stock price trend based on sentiment.

**Implementation/Approach:**

1. Fetch twitter data

2. Data Cleaning

3. Stemming

4. Topic modeling

5. Sentimental Analysis

**Issues Faced:**

I was not able to scrape news from website so changed the scope the project to twitter data only.

**Failures:**

1. Not able to relate news sentiment to stock price. I have created another graph based on stock price (same date as news) and compared two.

**Would like to implement in next phase:**

1. Fetch /scrape stock news from any website which gives relevant information.
2. Depending on the sentiment, predict the stock price trend.

**Code:** Twitter\_analysis\_final.R

**Output files:**

Tweets.csv

Grphs\_result.docx

**Code Description =**

**1. Install Packages (If already installed, skip this)**

#uncomment this when required

#install.packages("twitteR")

#install.packages("tm")

#install.packages("wordcloud")

#install.packages("RColorBrewer")

#install.packages("ROAuth")

#install.packages("RCurl")

#install.packages("SnowballC")

#install.packages("topicmodels")

#install package sentiment140

#install.packages("devtools")

#install\_github("sentiment140", "okugami79")

#install.packages("sentiment")

#install.packages("data.table")

2**. Load Libraries**

# Load this list of libraries

library(data.table)

library(sentiment)

library(devtools)

library(topicmodels)

library(twitteR)

library(tm)

library(wordcloud)

library(RColorBrewer)

library(ROAuth)

library(RCurl)

library(tm)

library(SnowballC)

library(ggplot2)

library(quantmod)

library(TTR)

3. Login to <https://apps.twitter.com/>.

Create an App.

Click on Keys and Access Tokens tab. You will find Consumer Key (API Key),Consumer Key (API Key)

Access Token, Access Token Secret. Copy the values in code below.

# get the credentials

reqURL <- "https://api.twitter.com/oauth/request\_token"

accessURL <- "https://api.twitter.com/oauth/access\_token"

authURL <- "https://api.twitter.com/oauth/authorize"

consumerKey <- "xxxx"

consumerSecret <- "xxxx"

access\_token <- "xxxx"

access\_secret <- "xxxx"

setup\_twitter\_oauth(consumerKey,consumerSecret,access\_token,access\_secret)

**4. Search Twitter public tweets for keyword “infy Infosys”**

# searching for tweets containing the keywords “infy”

gt = searchTwitter("infy infosys",n=1000)

gt

#length

n.gt = length(gt)

n.gt

**#convert to dataframe**

gt.df = twListToDF(gt)

gt.df

View(gt.df)

write.csv(gt.df, file="tweets.csv")

**#Fetch the tweets and write to tweets.csv. No need to fetch tweets again. Uncomment line below and run the code. Do not run lines above this.**

#gt.df <- read.csv("tweets.csv")

**5. Data Cleaning**

# build a corpus, and specify the source to be character vectors

mCorpus <- Corpus(VectorSource(gt.df$text))

# remove URLs

removeURL <- function(x) gsub("http[^[:space:]]\*", "", x)

mCorpus <- tm\_map(mCorpus, content\_transformer(removeURL))

# remove anything other than English letters or space

removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]\*", "", x)

mCorpus <- tm\_map(mCorpus, content\_transformer(removeNumPunct))

# remove stopwords

mStopwords <- stopwords('english')

mCorpus <- tm\_map(mCorpus, removeWords, mStopwords)

# remove extra whitespace

myCorpus <- tm\_map(myCorpus, stripWhitespace)

# keep a copy for stem completion later

mCorpusCopy <- mCorpus

**6. Stemming**

#stemming

newCorpus <- tm\_map(mCorpus, stemDocument) # stem words

writeLines(strwrap(mCorpus[[200]]$content, 60))

stemCompletion2 <- function(x, dictionary) {

x <- unlist(strsplit(as.character(x), " "))

x <- x[x != ""]

x <- stemCompletion(x, dictionary=dictionary)

x <- paste(x, sep="", collapse=" ")

PlainTextDocument(stripWhitespace(x))

}

newCorpus <- lapply(mCorpus, stemCompletion2, dictionary=mCorpusCopy)

newCorpus <- Corpus(VectorSource(mCorpus))

writeLines(strwrap(myCorpus[[200]]$content, 60))

**output: (200th line of dataset)**

RT DeependerSHooda As one started Software Engineer Infosys

I feel sad ongoing Infy needs preserve stood f

**7. Count Word Frequency**

# count word frequency

wordFreq <- function(corpus, word) {

results <- lapply(corpus,

function(x){grep(as.character(x), pattern=paste0("\\<",word)) }

)

sum(unlist(results))

}

n.infy <- wordFreq(mCorpus, "infy")

n.infosys <- wordFreq(mCorpus, "infosys")

cat(n.infy, n.infosys)

**output: 65 38**

new.infy <- wordFreq(newCorpus, "infy")

new.infosys <- wordFreq(newCorpus, "infosys")

cat(n.infy, n.infosys)

**output: 65 38**

tdm <- TermDocumentMatrix(newCorpus,

control = list(wordLengths = c(1, Inf)))

View(as.matrix(tdm))

**# creating document term matrix**

tdm = TermDocumentMatrix(newCorpus,

control = list(removePunctuation = TRUE,

removeNumbers = TRUE, tolower = TRUE))

tdm

idx <- which(dimnames(tdm)$Terms %in% c("infy","infosys","sikka","resign"))

idx

output: **653 666 1285**

as.matrix(tdm[idx,100:200])

**# find frequent words**

(freq.terms <- findFreqTerms(tdm,lowfreq = 30))

freq.terms

**Output:**

[1] "action" "affect" "american" "amp" "artificial" "board"

[7] "buyback" "ceo" "class" "deependershooda" "dream" "edge"

[13] "engineer" "exit" "express" "feel" "firms" "founders"

[19] "gives" "indian" "infosys" "infy" "infys" "intelligence"

[25] "investors" "large" "law" "limited" "losses" "mull"

[31] "murthys" "narayana" "needs" "one" "oneindiaonegst" "ongoing"

[37] "possible" "preserve" "recover" "resignation" "resigns" "robotics"

[43] "sad" "shares" "sikka" "sikkas" "software" "started"

[49] "stood" "suit" "tech" "the" "via" "vishal"

[55] "vsikka" "will"

term.freq <- rowSums(as.matrix(tdm))

term.freq <- subset(term.freq, term.freq >= 30)

df <- data.frame(term = names(term.freq), freq = term.freq)

df

**Output:**

term freq

action action 36

affect affect 236

american american 58

amp amp 125

artificial artificial 221

board board 38

buyback buyback 60

ceo ceo 100

class class 34

deependershooda deependershooda 122

dream dream 56

edge edge 223

engineer engineer 123

exit exit 254

express express 35

feel feel 124

firms firms 46

founders founders 42

gives gives 56

indian indian 50

infosys infosys 625

infy infy 556

infys infys 233

intelligence intelligence 221

investors investors 41

large large 32

law law 56

limited limited 55

losses losses 32

mull mull 33

murthys murthys 30

narayana narayana 33

needs needs 125

one one 141

oneindiaonegst oneindiaonegst 121

ongoing ongoing 123

possible possible 32

preserve preserve 123

recover recover 33

resignation resignation 40

resigns resigns 31

robotics robotics 235

sad sad 134

shares shares 36

sikka sikka 83

sikkas sikkas 55

software software 126

started started 124

stood stood 122

suit suit 35

tech tech 230

the the 61

via via 40

vishal vishal 82

vsikka vsikka 33

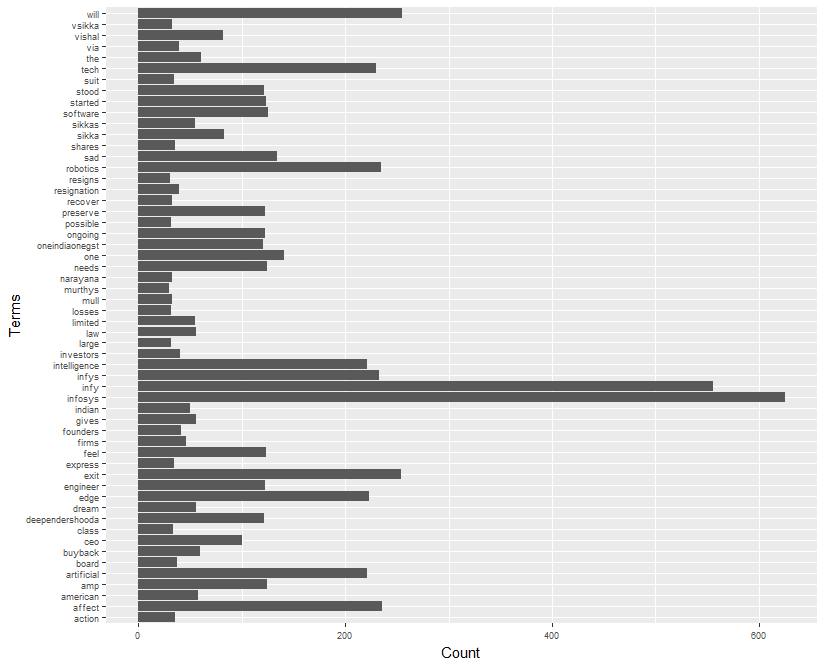
will will 255

**#graph1**

ggplot(df, aes(x=term, y=freq)) + geom\_bar(stat="Identity") +

xlab("Terms") + ylab("Count") + coord\_flip() +

theme(axis.text=element\_text(size=7))



m <- as.matrix(tdm)

# calculate the frequency of words and sort it by frequency

word.freq <- sort(rowSums(m), decreasing = T)

# colors

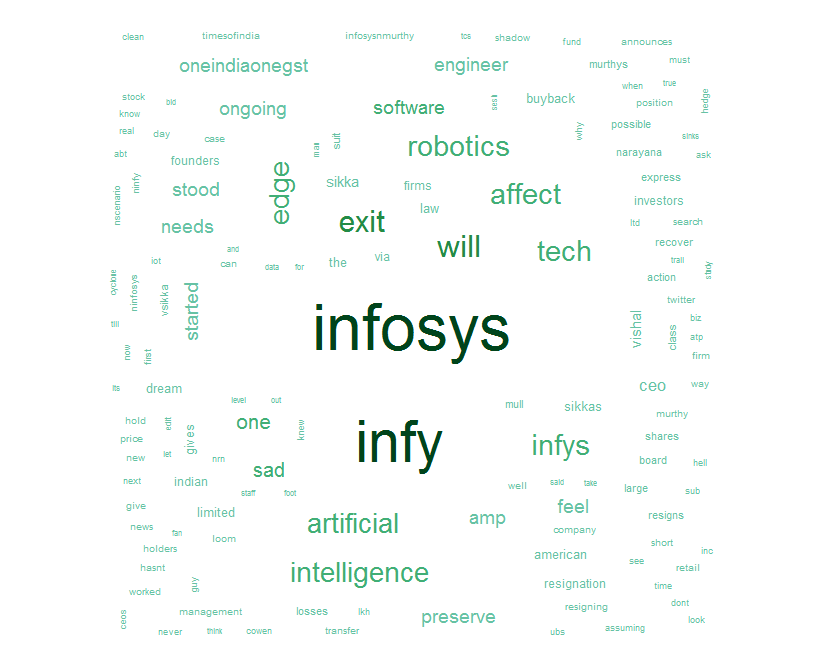
pal <- brewer.pal(9, "BuGn")[-(1:4)]

**# plot word cloud**

**#graph2**

wordcloud(words = names(word.freq), freq = word.freq, min.freq = 3,

random.order = F, colors = pal)



findAssocs(tdm, "murthy", 0.2)

findAssocs(tdm, "infosys", 0.2)

**8. Topic Modelling**

#topic Modelling

dtm <- as.DocumentTermMatrix(tdm)

dtm

**Output:**

DocumentTermMatrix (documents: 3, terms: 1588)>>

Non-/sparse entries: 1589/3175

Sparsity : 67%

Maximal term length: 24

Weighting : term frequency (tf)

#ui = unique(dtm$i)

#dtm.new = dtm[ui,]

lda <- LDA(dtm, k = 8) # find 8 topics

lda

term <- terms(lda, 7) # first 7 terms of every topic

(term <- apply(term, MARGIN = 2, paste, collapse = ", "))

**Output:**

Topic 1

"infy, feel, tech, infys, started, one, robotics"

Topic 2

"intelligence, will, exit, feel, robotics, affect, infy"

Topic 3

"infy, will, exit, robotics, intelligence, artificial, affect"

Topic 4

"infosys, will, tech, feel, affect, robotics, edge"

Topic 5

"infy, edge, robotics, feel, will, tech, exit"

Topic 6

"infosys, infy, intelligence, edge, exit, infys, artificial"

Topic 7

"infy, edge, robotics, ongoing, artificial, infys, started"

Topic 8

"infosys, infy, tech, affect, exit, needs, infys"

**9. Sentimental Analysis**

**# sentiment analysis**

sentiments <- sentiment(gt.df$text)

table(sentiments$polarity)

**output:**

negative neutral positive

170 772 58

**# sentiment plot**

sentiments$score <- 0

sentiments$score[sentiments$polarity == "positive"] <- 1

sentiments$score[sentiments$polarity == "negative"] <- -1

sentiments$date <- as.IDate(gt.df$created)

result <- aggregate(score ~ date, data = sentiments, sum)

result

**Output:**

date score

1 2017-08-18 -4

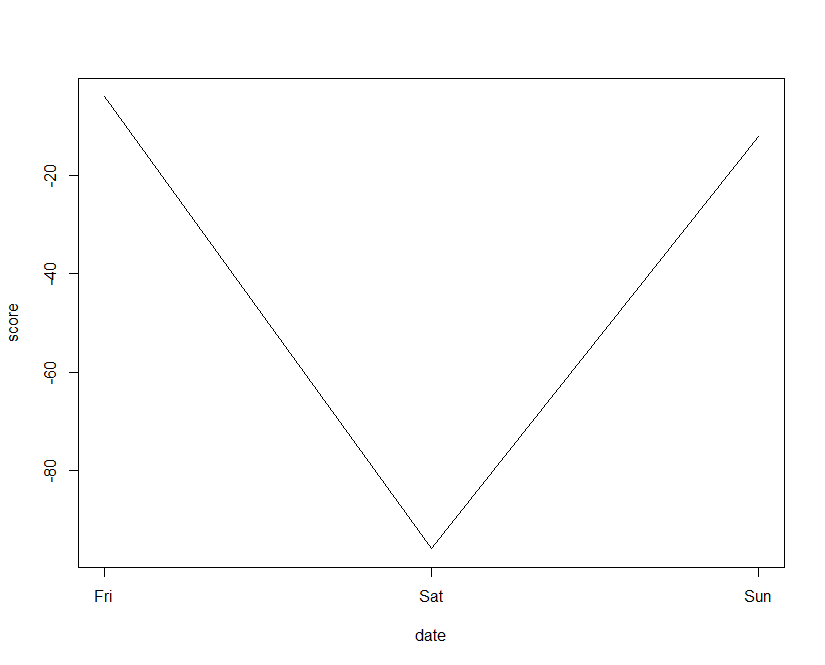
2 2017-08-19 -96

3 2017-08-20 -12

sentiments

#graph3

plot(result, type = "l")



**#draw chart series and find whether result$score plot matched with stock price chart for the same dates**

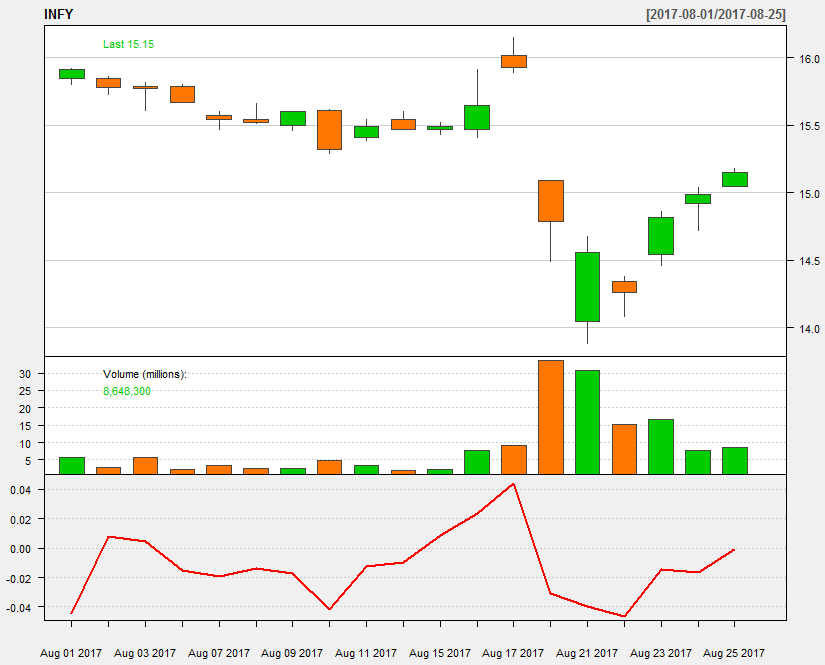
getSymbols("INFY")

INFY['2017-08-17::2017-08-20']

#Graph4

chartSeries(INFY,theme='white',subset = '2017-08')

addROC(n=200)

s

###compare sentiment 140 result with this.

# getting positive and negative words txt file

#positive\_words=scan("C:\\Users\\Public\\Documents\\positive-words.txt",what="character",comment.char=";")

#negative\_words=scan("C:\\Users\\Public\\Documents\\negative-words.txt",what="character",comment.char=";")

#positive\_words = c(positive\_words, "new","nice" ,"good", "horizon")

#positive\_words

#negative\_words = c(negative\_words, "wtf", "behind","resign","feels","ugly", "back","worse" ,"shitty", "bad", "no","freaking","sucks","horrible","miss")

# code for sentiment analysis

#score.sentiment = function(sentences, pos.words, neg.words, .progress='none')

#{

# library(plyr)

# library(stringr)

# we got a vector of sentences. plyr will handle a list

# or a vector as an "l" for us

# we want a simple array ("a") of scores back, so we use

# "l" + "a" + "ply" = "lapply":

# scores = laply(sentences, function(sentence, pos.words, neg.words) {

# clean up sentences with R's regex-driven global substitute, gsub():

# sentence = gsub('[[:punct:]]', '', sentence)

# sentence = gsub('[[:cntrl:]]', '', sentence)

# sentence = gsub('\\d+', '', sentence)

# and convert to lower case:

# sentence = tolower(sentence)

# split into words. str\_split is in the stringr package

# word.list = str\_split(sentence, '\\s+')

# sometimes a list() is one level of hierarchy too much

# words = unlist(word.list)

# compare our words to the dictionaries of positive & negative terms

# pos.matches = match(words, pos.words)

# neg.matches = match(words, neg.words)

# match() returns the position of the matched term or NA

# we just want a TRUE/FALSE:

# pos.matches = !is.na(pos.matches)

# neg.matches = !is.na(neg.matches)

# and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():

# score = sum(pos.matches) - sum(neg.matches)

# return(score)

# }, pos.words, neg.words, .progress=.progress )

# scores.df = data.frame(score=scores, text=sentences)

# return(scores.df)

#}

#test <- ldply(gt,function(t) t$toDataFrame() )

# perform sentiment analysis for the dataset tweets and store it in Results

#Results=score.sentiment(test$text,positive\_words,negative\_words)

#View(Results)

# getting the summary of the Results

#summary(Results)

# plot histogram

#hist(Results$score)

#table(Results$score)

#library(ggplot2)

#qplot(Results$score,xlab = "Score of tweets")