

**Project on**

**Twitter Data Analysis for Enhancement of Customer Service.**

**Subject**

**Software Configuration Management**

**Swe-503**

**Team members**

K. Vinay Reddy 12MSE1013

Y. Karthik 12MSE1052

K. Raveendra 12MSE1101

G. Nithin 12MSE1054

**Under**

Dr. Prof Sakthi Ganesh

SCSE

VITCC

**Abstract**

* To develop a tool which analyse the Twitter data and come to a conclusion what customers are thinking about the product.

**Objective**

* To get the data from the Twitter through Tweet API.
* Extracting the Data.
* Analysing the Tweets.
* Coming to a conclusion what customers are thinking about the Product with Personal Insights.

‘**#’**

* To get overall analysis ‘#’.
* To engage in what People are thinking about the Product.
* Giving Personal Insights to Every tweet made out of.
* Which they can be used to take decision to improve Customer Service.

**About Project**

The steps below will help you set up your twitter account to be able to access the live stream.

* Create a twitter account if you do not already have one.
* Go to [Twitter Dev] (https://dev.twitter.com/apps) and log in with your twitter credentials.
* Click "Create New App"
* Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.
* On the next page, click the "API Keys" tab along the top, then scroll all the way down until you see the section "Your Access Token"
* Click the button "Create My Access Token". You can Read more about Oauth authorization.
* You will now copy four values into `Twitter\_Analysis.R`. These values are your "API Key", your "API secret", your "Access token" and your "Access token secret". All four should now be visible on the API Keys page. (You may see "API Key" referred to as "Consumer key" in some places in the code or on the web; they are synonyms.)
* Open `twitterstream.py` and set the variables corresponding to the api key, api secret, access token, and access secret. You will see code like the below:
  + api\_key = "Enter api key"
  + api\_secret = "Enter api secret"
  + access\_token\_key = "Enter your access token key here"
  + access\_token\_secret = "Enter your access token secret here"
* Run the following and make sure you see data flowing and that no errors occur `$ Twitter\_Analysis.R
* This command pipes the output to a file. Stop the program with Ctrl-C, but wait at least 3 minutes for data to accumulate.
* If you wish, modify the file to use the twitter search API to search for specific terms. For example, to search for the term "Microsoft", you can pass the following url to the twittered function: `https://api.twitter.com/1.1/search/tweets.json?q=microsoft`

**Deriving the sentiment of each tweet**

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

For this part, we will compute the sentiment of each tweet based on the sentiment scores of the terms in the tweet. The sentiment of a tweet is equivalent to the sum of the sentiment scores for each term in the tweet.

The file Twitter\_Analysis.R contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a sentiment score. Each word or phrase that is found in a tweet but not found in Twitter\_Analysis.R should be given a sentiment score of 0.

The file ` Twitter\_Analysis.R ` contains the code used for deriving the sentiment of each tweet.

**Deriving the sentiment of new terms**

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

Here's how you might think about the problem: We know we can use the sentiment-carrying words in Twitter\_Analysis.R to deduce the overall sentiment of a tweet. Once you deduce the sentiment of a tweet, you can work backwards to deduce the sentiment of the non-sentiment carrying words that do not appear in Twitter\_Analysis.R. For example, if the word soccer always appears in proximity with positive words like great and fun, then we can deduce that the term soccer itself carries a positive sentiment.

**Code:**

rm(list=ls())

#install.packages(c("devtools", "rjson", "bit64", "httr"))

library(devtools)

library(twitteR)

library(tm)

library(wordcloud)

library(ggplot2)

library(fpc)

library(qdap)

library(igraph)

library(rjson)

library(bit64)

library(httr)

#Application Details

APIKey="Av57SzksBJUC7DVyxlnPqSrqJ"

APISecret="CcHcLNDWfeSWBsDGfeVVTg7r9TJ8TCQ3KXnTcdqQvgvuAFqpeD"

AccessToken="291732148-i11DB10BufaRpAx3Rj3cw9pXPF4qBPs2SXcwdfyq"

AccessTokenSecret="Q7iAZdKnihYmeDOENOn9A50Q2HAjncyDcB5LzpgHBlKZq"

Auth<-setup\_twitter\_oauth(APIKey,APISecret,AccessToken,AccessTokenSecret)

#Fetching tweets of input Hashtag

fetchHashtag <- function(hashtag,number) {

tweets <- searchTwitter(hashtag, number, lang="en")

return(tweets)

}

makeCorpus <- function(text){ #Function for making corpus and cleaning the tweets fetched

twitterdf <- do.call("rbind", lapply(text, as.data.frame)) #store the fetched tweets as a data frame

twitterdf$text <- sapply(twitterdf$text,function(row) iconv(row, "latin1", "ASCII", sub=""))#Removing emoticons from tweets

twitterCorpus <- Corpus(VectorSource(twitterdf$text)) #Creating Corpus

toSpace <- content\_transformer(function(x, pattern) gsub(pattern, " ", x)) #function to replace a pattern to white space using regex

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "(RT|via)((?:\\b\\W\*@\\w+)+)") #match rt or via

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "@\\w+") #match @

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "[ \t]{2,}") #match tabs

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "[ |\n]{1,}") #match new lines

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "^ ") #match white space at begenning

twitterCorpus <- tm\_map(twitterCorpus, toSpace, " $") #match white space at the end

twitterCorpus <- tm\_map(twitterCorpus, PlainTextDocument)

twitterCorpus <- tm\_map(twitterCorpus, removeNumbers)

twitterCorpus <- tm\_map(twitterCorpus, removePunctuation)

twitterCorpus <- tm\_map(twitterCorpus, toSpace, "http[[:alnum:]]\*") #remove url from tweets

twitterCorpus <- tm\_map(twitterCorpus,removeWords,stopwords("en"))

twitterCorpus <- tm\_map(twitterCorpus, content\_transformer(tolower))

return(twitterCorpus)

}

#for (i in 1:100){

#cat(paste("[[", i, "]] ", sep=""))

#writeLines(strwrap(corp[[i]], width=73))

#}

#Wordcloud

makeWordcloud<-function (getText){ #plotting wordcloud

twicorpus<-makeCorpus(getText)

myTdm<-TermDocumentMatrix(twicorpus, control=list(wordLengths=c(4,Inf))) #Create TDM

matrix<-as.matrix(myTdm)

wordFreq <- sort(rowSums(matrix), decreasing=TRUE)#find frequency of words and sorting them in descending

set.seed(375)

grayLevels <- gray( (wordFreq+10) / (max(wordFreq)+10) )

wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3, random.order=F,colors=grayLevels)

}

freqPlot<-function (getText){ #frequency plot of word count

twicorpus<-makeCorpus(getText)

myTdm<-TermDocumentMatrix(twicorpus, control=list(wordLengths=c(4,Inf)))

matrix<-as.matrix(myTdm)

termFrequency <- rowSums(matrix)

termFrequency <- subset(termFrequency, termFrequency>=10)

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

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

}

#Clustering

hCluster<-function (content){ #hierarchical clustering

twicorpus<-makeCorpus(content)

myTdm<-TermDocumentMatrix(twicorpus, control=list(wordLengths=c(4,Inf)))

myTdm2 <- removeSparseTerms(myTdm, sparse=0.98) #removing sparse terms

m2 <- as.matrix(myTdm2)

distMatrix <- dist(scale(m2)) #calculating distance between terms

fit <- hclust(distMatrix, method="ward.D") #clustering terms

plot(fit)

rect.hclust(fit, k=5) #cutting the tree into 5 clusters

(groups <- cutree(fit, k=5))

}

kMeans<-function (content){ #k-means clustering

twicorpus<-makeCorpus(content)

myTdm<-TermDocumentMatrix(twicorpus, control=list(wordLengths=c(4,Inf)))

myTdm2 <- removeSparseTerms(myTdm, sparse=0.98)

m2 <- as.matrix(myTdm2)

m3 <- t(m2) # creating transpose of matrix

set.seed(122)

k <- 8

kmeansResult <- kmeans(m3, k)

round(kmeansResult$centers, digits=3) #cluster centers

for (i in 1:k) { #printing 15 terms of each cluster

cat(paste("cluster ", i, ": ", sep=""))

s <- sort(kmeansResult$centers[i,], decreasing=T)

cat(names(s)[1:15], "\n")

}

}

kMediod<-function (content){

twicorpus<-makeCorpus(content)

myTdm<-TermDocumentMatrix(twicorpus, control=list(wordLengths=c(4,Inf)))

myTdm2 <- removeSparseTerms(myTdm, sparse=0.95)

m2 <- as.matrix(myTdm2)

m3 <- t(m2)

pamResult <- pamk(m3, metric="manhattan")

return(pamResult)

}

#Sentiment Analysis

tSentimen<-function (content){

twicorpus<-makeCorpus(content)

dataframe<-data.frame(text=unlist(sapply(twicorpus, `[`, "content")), stringsAsFactors=F) # storing corpus as data frame

(poldat <- with(dataframe, polarity(text))) #getting polarity of the tweets

return(poldat)

}

#Just for example, edit the hashtag and number of tweets to fetch

get<-fetchHashtag("#CocaCola",200) # fetching tweets of coca cola

freqPlot(get) #creating frequency plots

corp<-makeCorpus(get) #creating corpus

makeWordcloud(get) #creating wordcloud

hCluster(get) #hierarchical clustering

kMeans(get) #k-means clustering

getSentiment1<-tSentimen(get) #fetching sentiment polarity

table(getSentiment1$all$polarity>0)

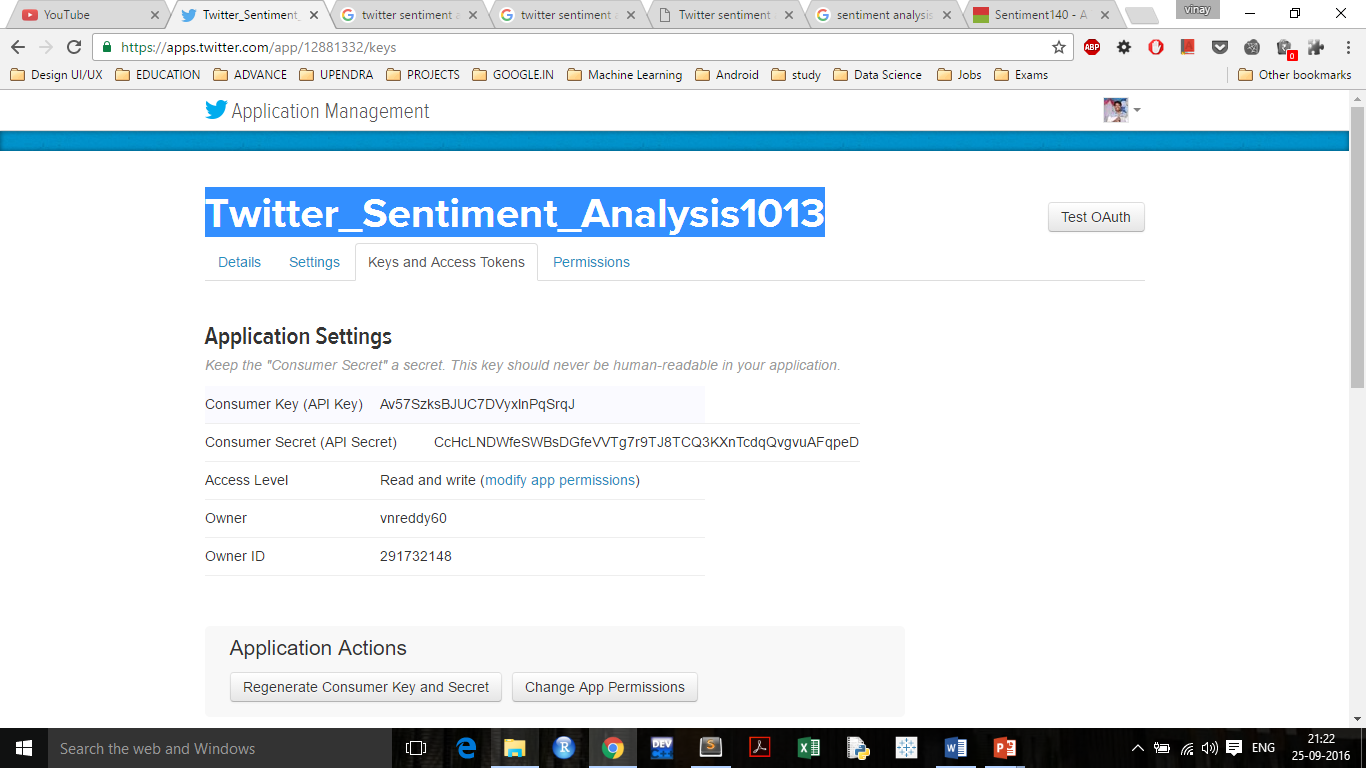
plot(getSentiment1) #polarity plot

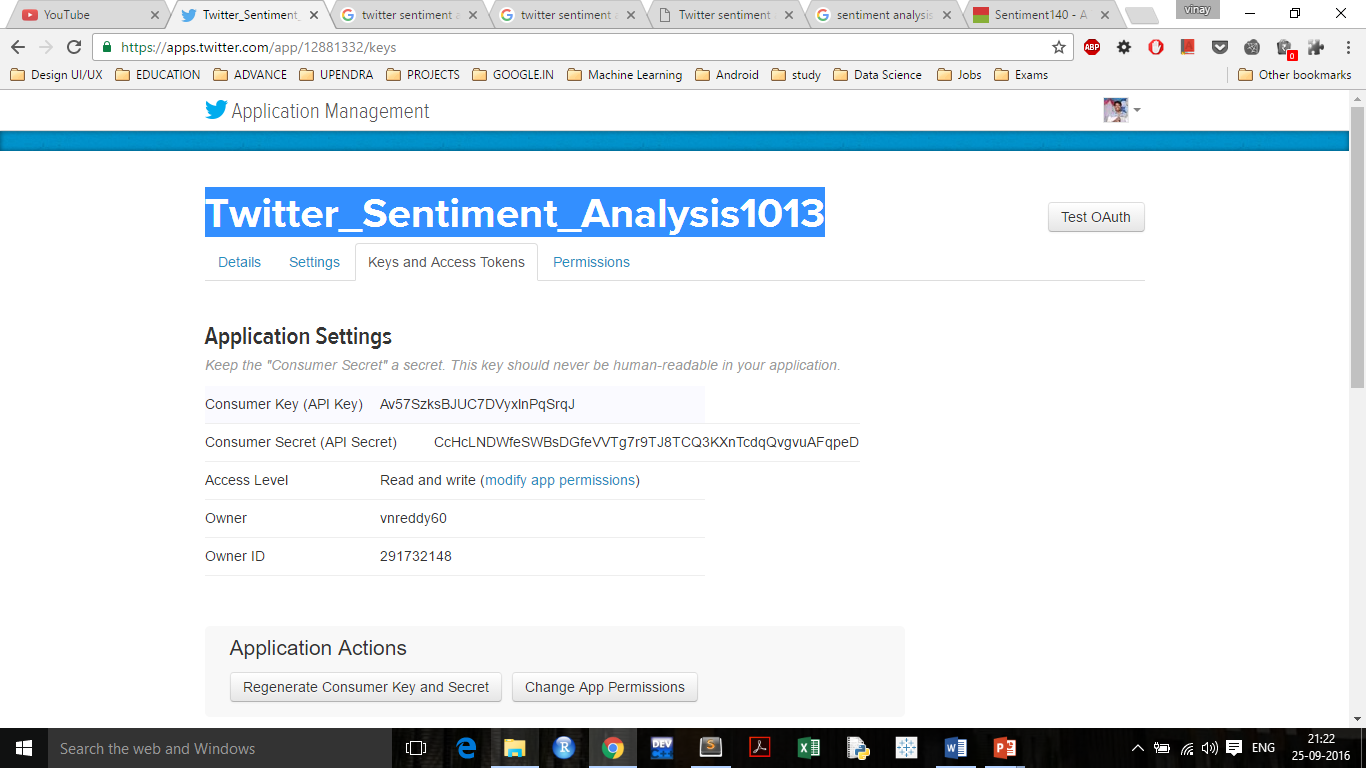
plot(getSentiment1$all$polarity,ylab="All Polarity",xlab="Documents",pch=20,cex=1,col="darkblue") #individual polarity plot

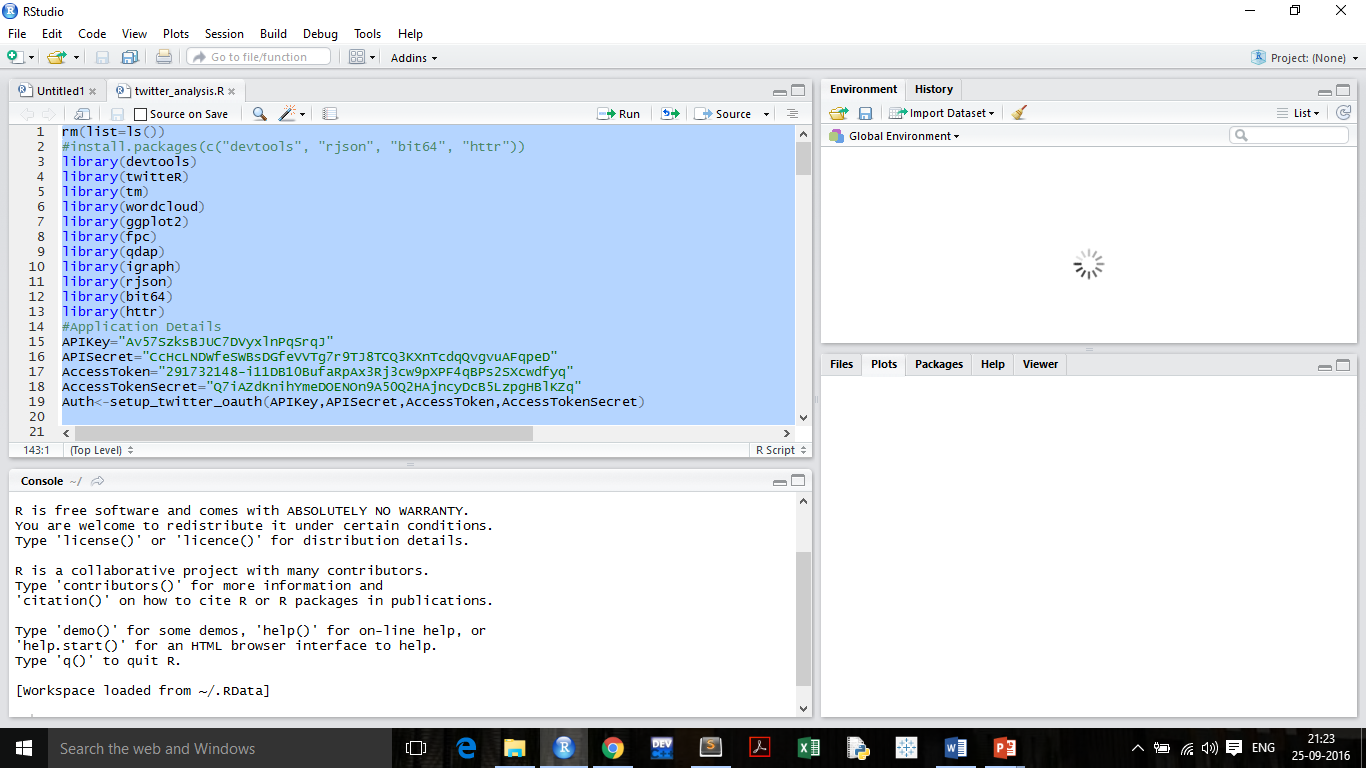
mean(getSentiment1$all$polarity)

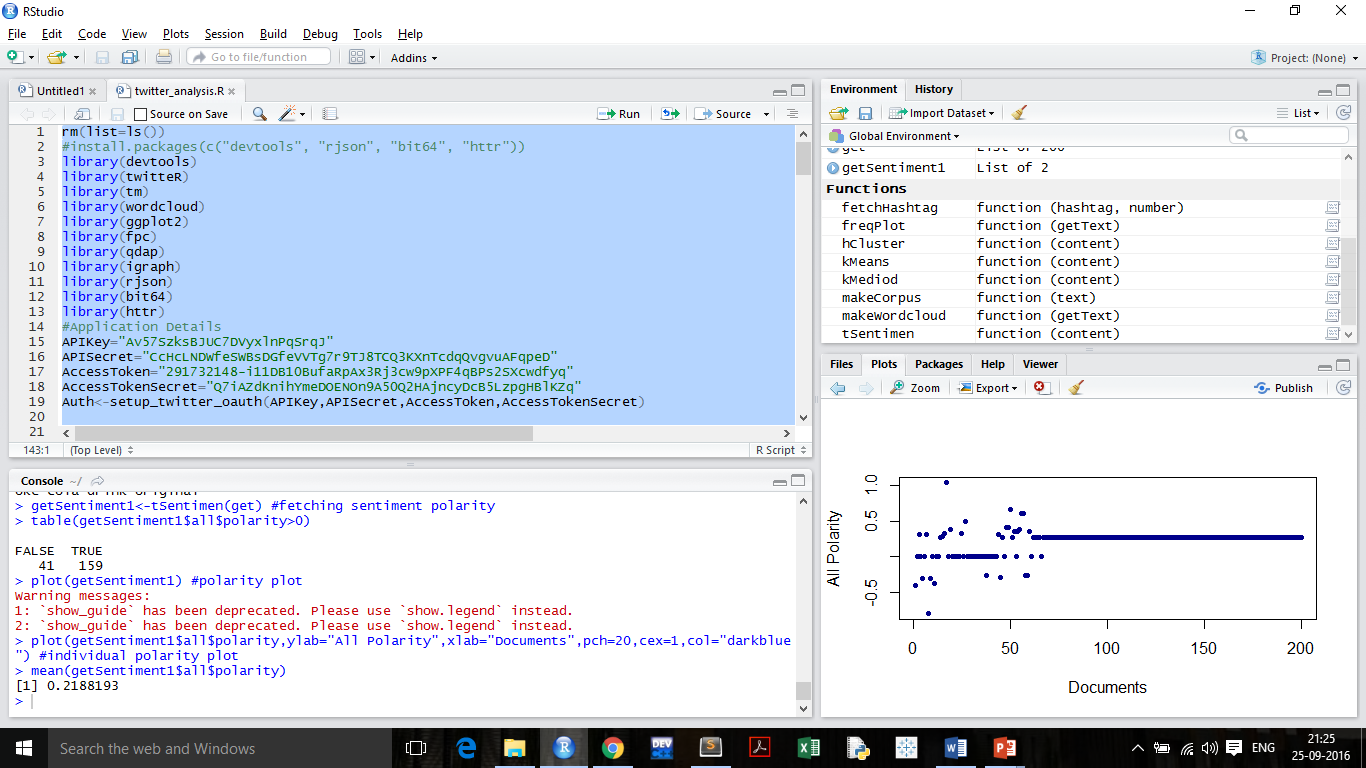
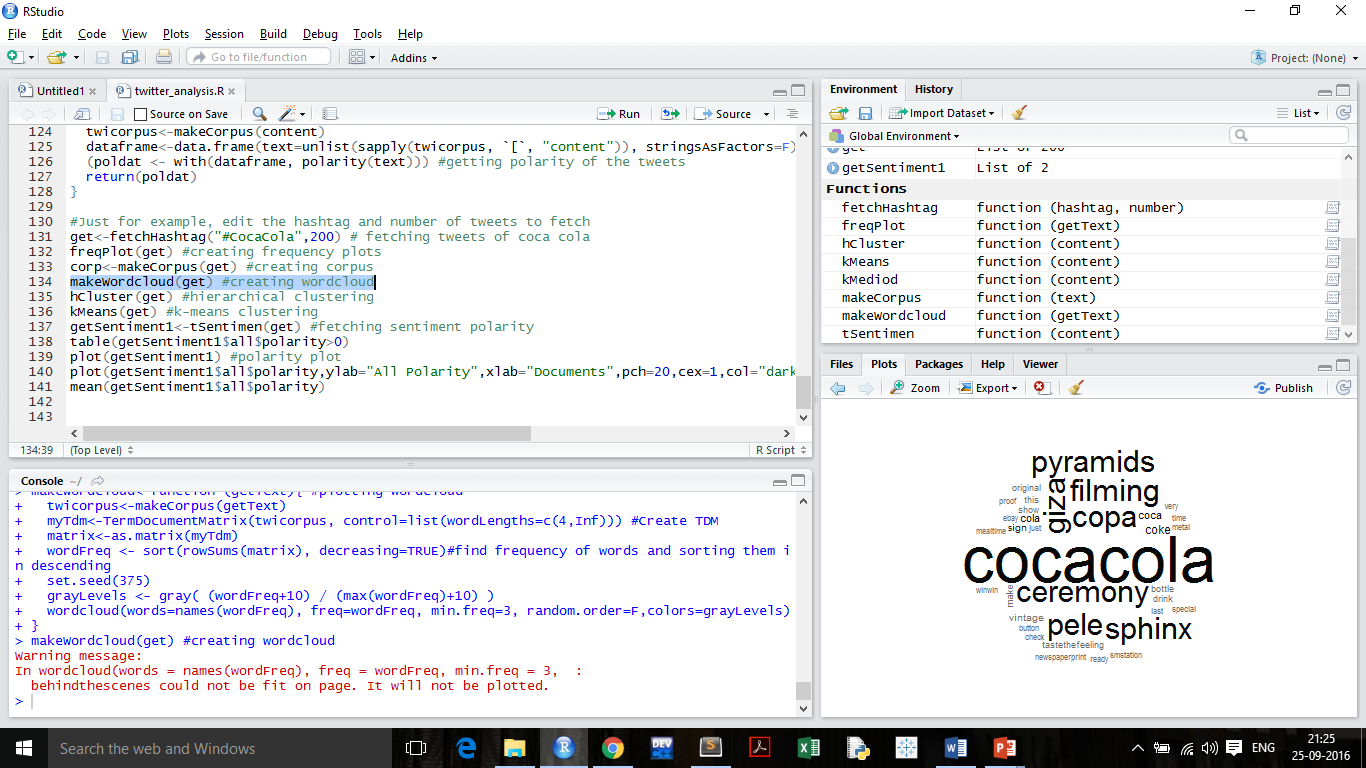
**Output ScreenShots:**

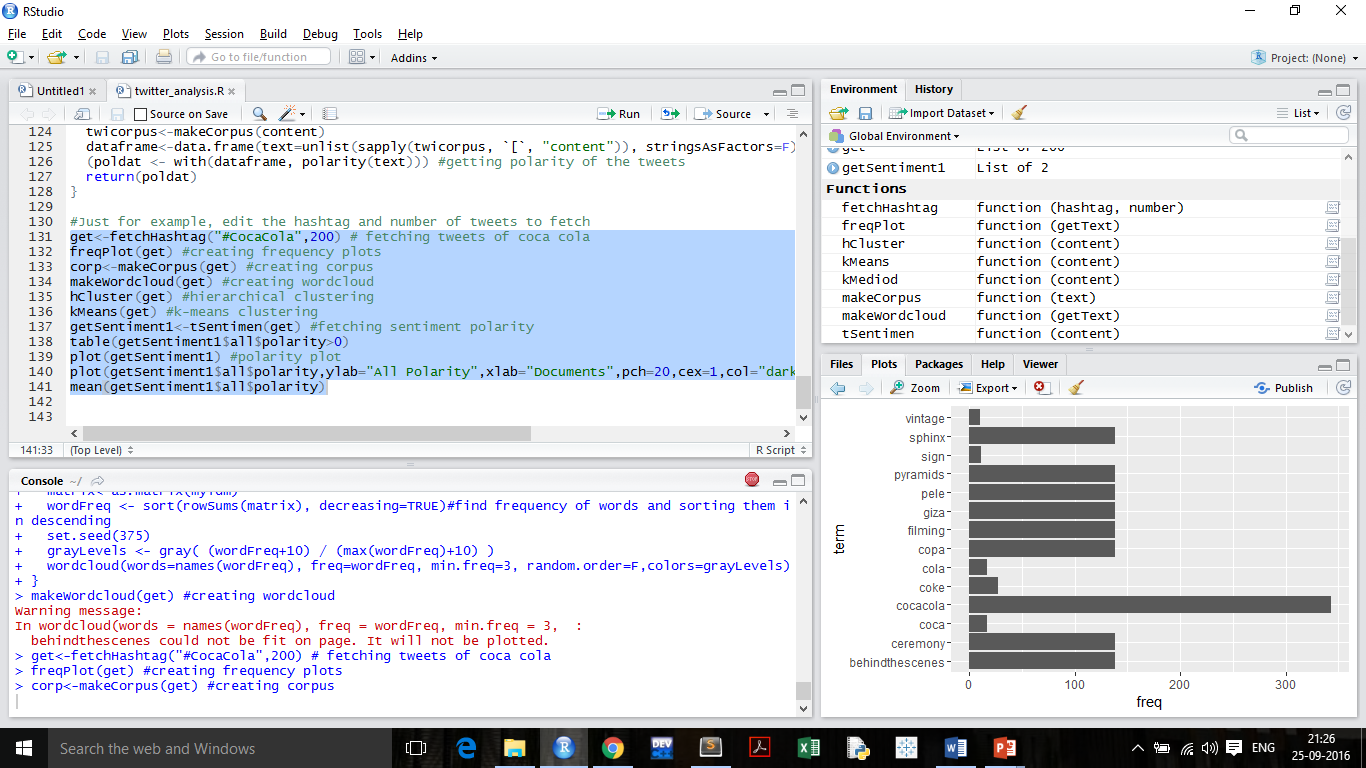
**Twitter\_Sentiment\_Analysis1013 Keys**

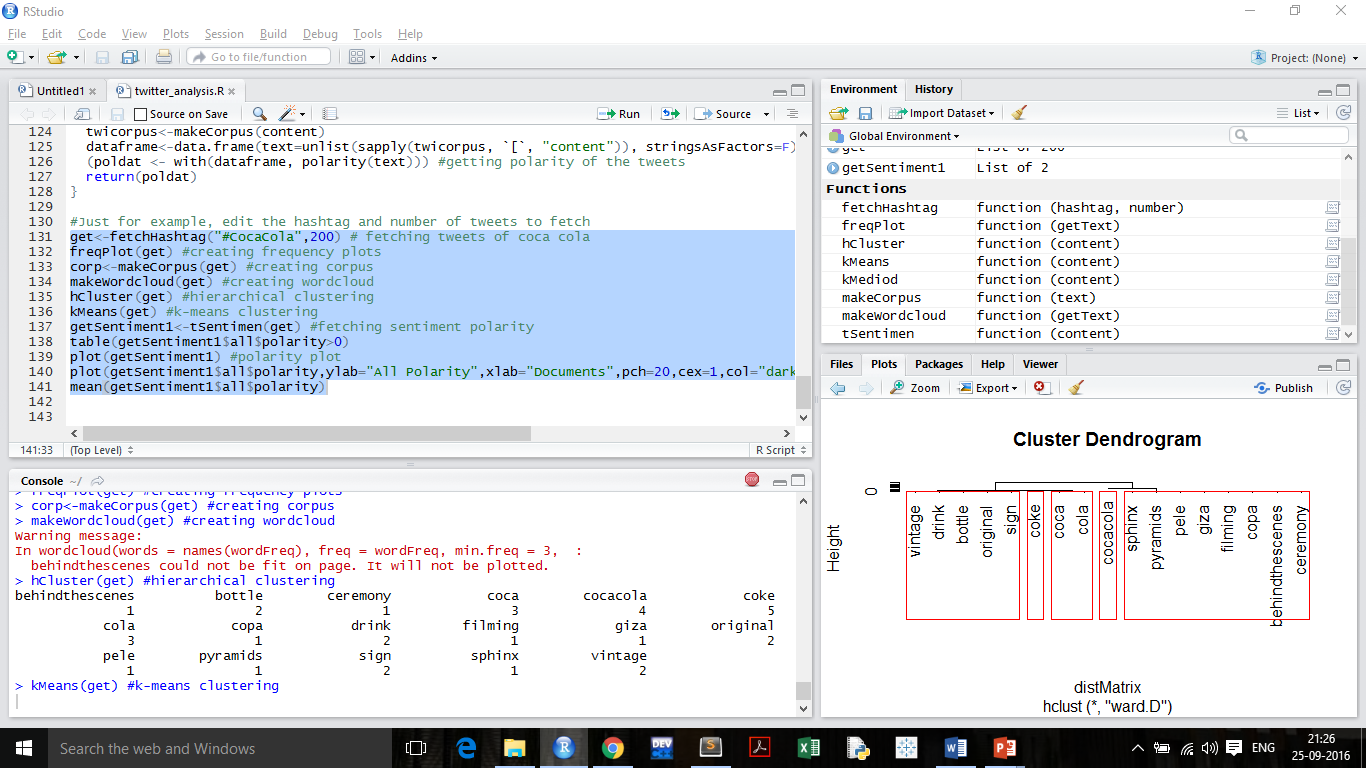
****

****

****

****

****