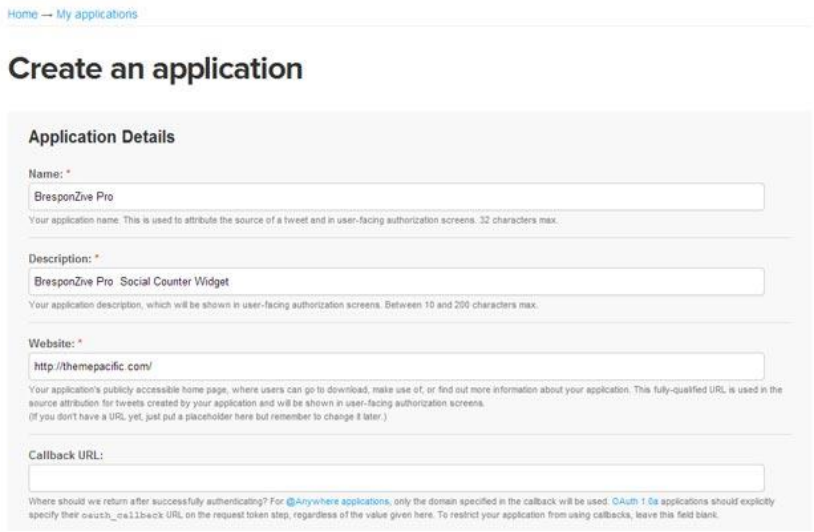
Name: Chitipolu Sri Sudheera
Tool : Rstudio

Flipkart Twitter Analysis:

Before getting into project let's see how to get the twitter API, Consumer Secret, Access Token, Access Secret.

How to Generate API Key, Consumer Token, Access Key for Twitter OAuth

- **Creating a Twitter Application**

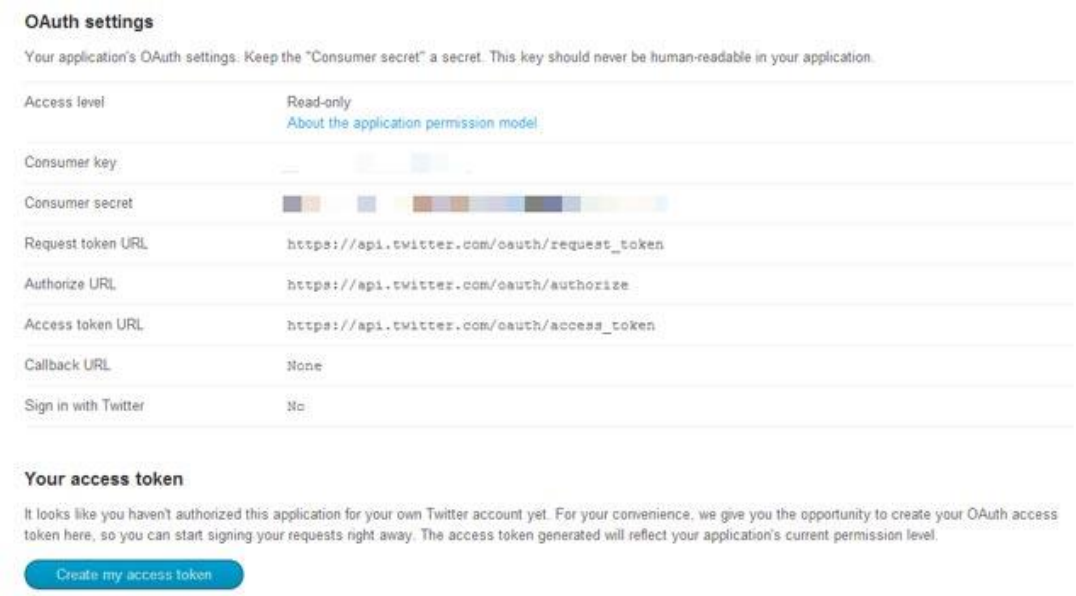


To use Twitter counter widget and other Twitter related widgets, you need OAuth access keys. To get Twitter Access keys, you need to create Twitter Application which is mandatory to access Twitter.

1. Go to https://dev.twitter.com/apps/new and log in, if necessary
2. Enter your Application Name, Description and your website address. You can leave the callback URL empty.
3. Accept the TOS, and solve the CAPTCHA.
4. Submit the form by clicking the **Create your Twitter Application**

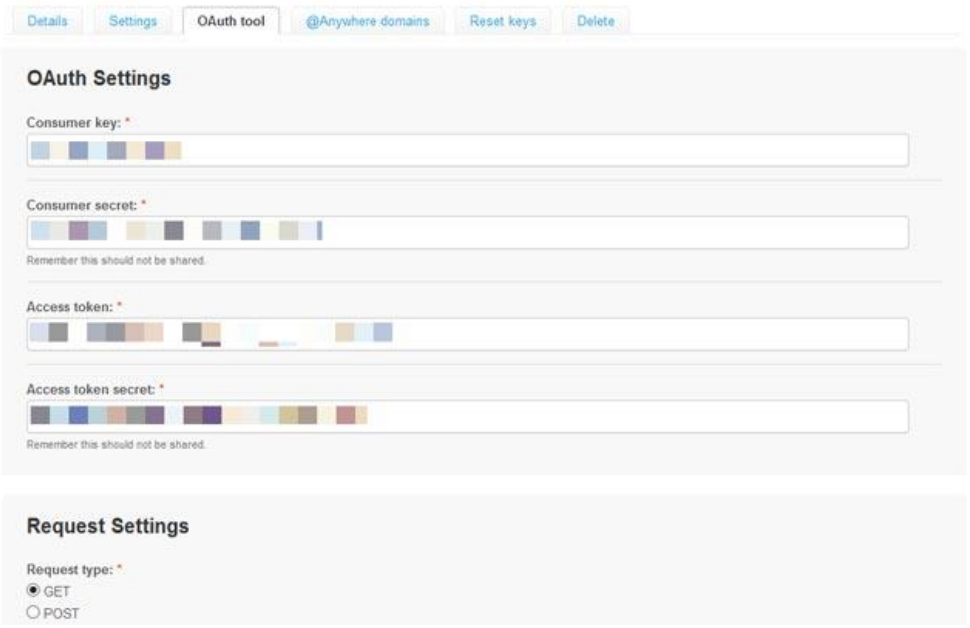
5. Copy the consumer key (API key) and consumer secret from the screen into your application

- **Create Your Access Token for OAuth**



After creating your Twitter Application, you have to give the access to your Twitter Account to use this Application. To do this, click the **Create my Access Token.**

- **Get the Consumer Key, Consumer Secret, Access token, Access Token Secret**



In order to access the Twitter, that is to get recent tweets and twitter followers count, you need the four keys such as Consumer Key, Consumer Secret, Access token, Access Token Secret.

```
#SENTIMENTAL ANALYSIS ON TWITTER


# install.packages("twitteR")
# install.packages("plyr")
# install.packages("stringr")
# install.packages("dplyr")
# install.packages("igraph")
# install.packages("tm")
# install.packages("wordcloud")
# install.packages("SnowballC")
# install.packages("ggplot2")
# install.packages("cluster")
# install.packages("fpc")


library(twitteR)
## Warning: package 'twitteR' was built under R version 3.4.3
library(plyr)
## Warning: package 'plyr' was built under R version 3.4.2
##
## Attaching package: 'plyr'
## The following object is masked from 'package:twitteR':
##
##     id
library(stringr)
## Warning: package 'stringr' was built under R version 3.4.2
library(dplyr, pos=99) # dplyr and igraph in high position to avoid masking p
lyr.
## Warning: package 'dplyr' was built under R version 3.4.2
##
## Attaching package: 'dplyr'
## The following objects are masked _by_ 'package:plyr':
##
```

```
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following objects are masked _by_ 'package:twitteR':
##
##     id, location
## The following objects are masked _by_ 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(igraph, pos=100)
## Warning: package 'igraph' was built under R version 3.4.3
##
## Attaching package: 'igraph'
## The following objects are masked _by_ 'package:stats':
##
##     decompose, spectrum
## The following objects are masked _by_ 'package:dplyr':
##
##     as_data_frame, groups, union
## The following object is masked from 'package:base':
##
##     union
library(tm)
## Warning: package 'tm' was built under R version 3.4.3
## Loading required package: NLP
library(SnowballC)
library(fpc)
## Warning: package 'fpc' was built under R version 3.4.2
library(wordcloud)
## Warning: package 'wordcloud' was built under R version 3.4.3
## Loading required package: RColorBrewer
library(cluster)
```

```
## Warning: package 'cluster' was built under R version 3.4.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
##
```

```
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:NLP':
```

```
##
```

```
##     annotate
```

```
#Collecting Data with twitteR
# #Authentication
# ConsumerKey   <-"jQ22IrA6bkJZMLrFJb4W9jF9d"
# ConsumerSecret<-"Gwy4pKv9jTVCpDijdmc4m4qsw4ytbAJ8N4m3eeEvrIUEqtF9g7"
# accesstoken = "931030805559721984-wBPysMkdCnBGnnsuife0T2qA9ya9mrg"
# accesssecret = "GfLp5fcmIp1WgczfTI4HlCfIavvfp9Kyt7vlFKOnMNwA2"
#
setup_twitter_oauth("jQ22IrA6bkJZMLrFJb4W9jF9d",
                    "Gwy4pKv9jTVCpDijdmc4m4qsw4ytbAJ8N4m3eeEvrIUEqtF9g7",
                     access_token = "931030805559721984-wBPysMkdCnBGnnsuife0T2qA9ya9mrg",
                    access_secret = "GfLp5fcmIp1WgczfTI4HlCfIavvfp9Kyt7vlFKOnMNwA2")
```

```
## [1] "Using direct authentication"
```

```
#Collecting Tweets on Flipkart
Tweets <- searchTwitter("@flipkart",lang="en", n = 500)
Tweets[1:5]
```

```
## [[1]]
```

```
## [1] "bikash_pati: @Flipkart just bought a defective Sony headset(MDR-XB450) after 5 days of wait. This is totally unacceptable. Pleas… https://t.co/hh1A6qwOnM"
```

```
##
```

```
## [[2]]
```

```
## [1] "Shashi528: RT @XiaomiIndia: Mi fans! Last chance to get your hands on the hot selling Mi products with amazing offers! Get it now on @Flipkart https:/…"
```

```
##
```

```
## [[3]]
```

```
## [1] "FlipkartStories: Among our favorite #news headlines: At @TiEDelhi Uni
on Minister @jayantsinha calls for more homegrown startups like… https://t.co
/TsskoAxwdp"

##

## [[4]]

## [1] "vjeens: @Flipkart @amazon Price goof up by flipkart... Cost on produc
t shown as 135, selling at 200...what a #Pinchday https://t.co/g2EHJUp5Lo"

##

## [[5]]

## [1] "couponscottage: Up to 80% off on #Grooming #books #Babycare #sports p
roducts @Flipkart \nhttps://t.co/yUQHAVzNyQ https://t.co/Yqt40nkWKn"
```

```
#Extracting the tweet text

Text = laply(Tweets, function(t) t$getText() )

Text[1:5]
```

```
## [1] "@Flipkart just bought a defective Sony headset(MDR-XB450) after 5 day
s of wait. This is totally unacceptable. Pleas… https://t.co/hh1A6qwOnM"

## [2] "RT @XiaomiIndia: Mi fans! Last chance to get your hands on the hot se
lling Mi products with amazing offers! Get it now on @Flipkart https:/…"

## [3] "Among our favorite #news headlines: At @TiEDelhi Union Minister @jaya
ntsinha calls for more homegrown startups like… https://t.co/TsskoAxwdp"

## [4] "@Flipkart @amazon Price goof up by flipkart... Cost on product shown
as 135, selling at 200...what a #Pinchday https://t.co/g2EHJUp5Lo"

## [5] "Up to 80% off on #Grooming #books #Babycare #sports products @Flipkar
t \nhttps://t.co/yUQHAVzNyQ https://t.co/Yqt40nkWKn"
```

```
#Cleaning the tweets

Clean_Tweet<-function(doc){

  doc <- gsub("/", " ", doc)

  doc <- gsub("@", " ", doc)

  doc <- gsub("\\|", " ", doc)

  doc <- gsub("\u2028", " ", doc)

  doc <-gsub("http[^[:space:]]*", "", doc)

  doc <-gsub("[^[:graph:]]", " ", doc)

  doc <- gsub("Ã", " ", doc)

  doc <- gsub("Â", " ", doc)

  doc <- gsub(",", "", doc)

  doc <- gsub("  ", " ", doc)

  doc <- tolower(doc)
```

```
  doc <- removePunctuation(doc)
  doc <- removeNumbers(doc)
  doc <- stripWhitespace(doc)
  doc <- removeWords(doc,stopwords("english")) # remove stop words
  doc <- unlist(str_split(doc, "\\s+"))
  doc <- removeWords(doc,c("flipkart","flipkartsupport","tco"))#removing othe
r irrelavant words
}


document<-Clean_Tweet(Text)
document[1:20]
```

```
##  [1] ""             ""             "just"         "bought"
##  [5] "defective"    "sony"         "headsetmdrxb" "days"
##  [9] "wait"         "totally"      "unacceptable" "pleas"
## [13] ""             "hhaqwonm"     "rt"           "xiaomiindia"
## [17] "mi"           "fans"         "last"         "chance"
```

visual analytics through WORDCLOUD

```
set.seed(142)
wordcloud(document,scale=c(3,0.3), colors=brewer.pal(6, "Dark2"))
```


```
#Wordcloud for Postive Sentiments
#Use this link for Positive and Negative Words to compare the words:
```

```
#https://github.com/jeffreybreen/twitter-sentiment-analysis-tutorial-201107/tree/master/data/opinion-lexicon-English
pos <- readLines(file.choose()) # Positive words
print("------------Postive Dictionary------------")
## [1] "------------Postive Dictionary------------"
sample(pos, size=10)
##  [1] "revive"      "willingness" "painlessly"  "resilient"   "judicious"
##  [6] "stirringly"  "zippy"       "improves"    "glorious"    "like"
neg <- readLines(file.choose()) # Negative words
print("------------Postive Dictionary------------")
## [1] "------------Postive Dictionary------------"
sample(neg, size=10)
##  [1] "wobbled"     "obscure"     "reprimand"   "solicitude"  "boredom"
##  [6] "throbbing"   "imprecisely" "dirt"        "unlawfully"  "touted"
```

The following code retrieves the positive matching words.

```
##Function for matching postive words

poswords=function(name){
  pmatch=match(t,pos)
  posw=pos[pmatch]
  posw=posw[!is.na(posw)]
  return(posw)
}

##Collection Positive Sentiments into a single data frame.

words=NULL
pos.data=data.frame(words)

for (t in document) {
  pos.data=c(poswords(t),pos.data)
}
```

```
head(pos.data,10)
```

```
## [[1]]
## [1] ""
##
## [[2]]
## [1] "fast"
##
## [[3]]
## [1] "happy"
##
## [[4]]
## [1] ""
##
## [[5]]
## [1] ""
##
## [[6]]
## [1] ""
##
## [[7]]
## [1] ""
##
## [[8]]
## [1] ""
##
## [[9]]
## [1] ""
##
## [[10]]
## [1] ""
```

```
#Function for matching Negative words


negwords=function(name){
  nmatch=match(n,neg)
```

```
  negw=neg[nmatch]
  negw=negw[!is.na(negw)]
  return(negw)
}

#Collection Negative Sentiments into a single data frame.

nwords=NULL
neg.data=data.frame(nwords)

for (n in document) {
  neg.data=c(negwords(n),neg.data)
}
head(neg.data,10)
```

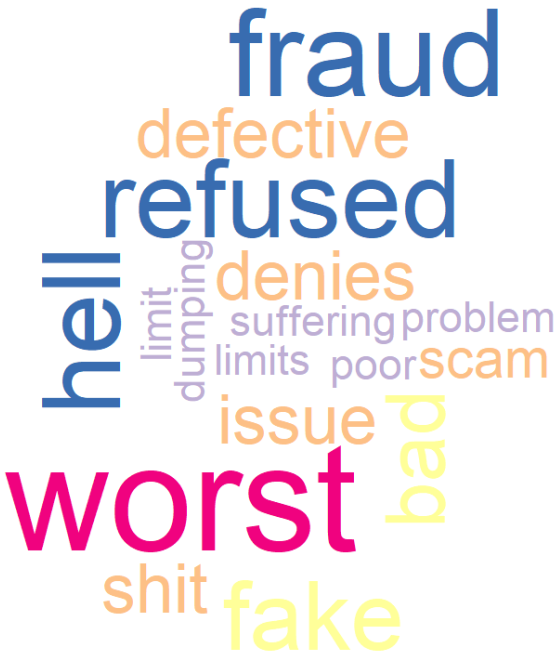
```
## [[1]]
## [1] ""
##
## [[2]]
## [1] ""
##
## [[3]]
## [1] ""
##
## [[4]]
## [1] ""
##
## [[5]]
## [1] ""
##
## [[6]]
## [1] ""
##
## [[7]]
## [1] ""
```

```
##
## [[8]]
## [1] ""
##
## [[9]]
## [1] ""
##
## [[10]]
## [1] "problem"
```

```
#Unlisting the words


PostiveWords<-unlist(pos.data)

NegativeWords<-unlist(neg.data)


#Wordcloud of Positive words


set.seed(123)

wordcloud(PostiveWords,scale=c(5,0.1),random.order = TRUE,rot.per = 0.20,use.
r.layout = FALSE,colors = brewer.pal(6,"Dark2"),max.words = 100)
```


```
#Wordcloud of Negative words


set.seed(578)

wordcloud(NegativeWords,scale=c(5,0.1),random.order = TRUE,rot.per = 0.20,use
.r.layout = FALSE,colors = brewer.pal(6,"Accent"),max.words = 100)
```

#Frequency of Words




```
##Genertaing Corpus


docs<-VCorpus(VectorSource(document))

docs
```

```
## <<VCorpus>>

## Metadata:  corpus specific: 0, document level (indexed): 0

## Content:  documents: 6561
```

```
#Converting the words into an appropriate form(Matrix)form for further proces
sing

tdm <- TermDocumentMatrix(docs)

tdm
```

```
## <<TermDocumentMatrix (terms: 1535, documents: 6561)>>

## Non-/sparse entries: 4665/10066470

## Sparsity           : 100%

## Maximal term length: 21
```

```
## Weighting          : term frequency (tf)
```

```
dtm<- DocumentTermMatrix(docs)
dtm
```

```
## <<DocumentTermMatrix (documents: 6561, terms: 1535)>>
## Non-/sparse entries: 4665/10066470
## Sparsity           : 100%
## Maximal term length: 21
## Weighting          : term frequency (tf)
```

```
#The most and least frequently occurring words.

freq <- colSums(as.matrix(dtm))
length(freq)
```

```
## [1] 1535
```

```
freq[1:10]
```

```
##        aajtak abhishekraanu          able         abpne        absurd
##             1             1             2             1             1
##    acceptable      accepted   accessories         acche       account
##             1             1             1             1             4
```

```
#The distribution of the least-frequently used words.

head(table(freq), 20)  # the 20 highest word frequencies,
```

```
## freq
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  16  17  18  19
## 989 208  90  61  44  29  15  19   9   9  11   4   5   7   4   2   4   2
##  20  21
##   1   2
```

```
tail(table(freq), 20)  # the 20 lowest word frequencies,
```

```
## freq
##  18  19  20  21  24  25  27  32  34  38  48  68  72  74  75  78  80  86
##   4   2   1   2   1   1   1   1   1   1   1   3   2   1   2   1   1   1
## 103 175
##   1   1
```

```
#This will identify all terms that appear frequently (in this case, 50 or mor
e times).
```

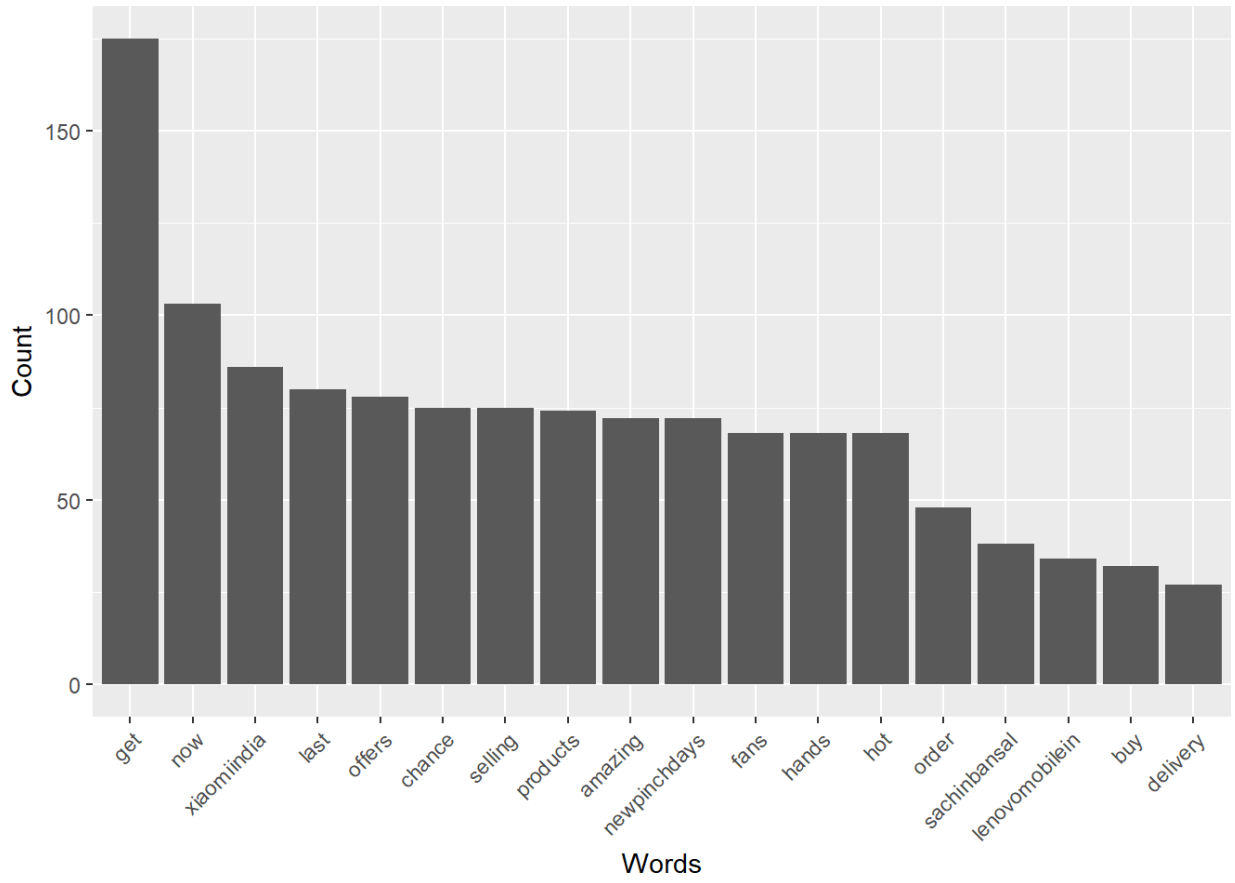
```
findFreqTerms(dtm, lowfreq=25)
```

```
##  [1] "amazing"        "buy"            "chance"         "delivery"
##  [5] "fans"           "get"            "hands"          "hot"
##  [9] "last"           "lenovomobilein" "motorolaindia"  "newpinchdays"
## [13] "now"            "offers"         "order"          "products"
## [17] "sachinbansal"   "selling"        "xiaomiindia"
```

```
#Frequency Table of words


wordfreq <- data.frame(word=names(freq), freq=freq)
head(wordfreq)
```

```
##                         word freq
## aajtak               aajtak    1
## abhishekraanu abhishekraanu    1
## able                   able    2
## abpne                 abpne    1
## absurd               absurd    1
## acceptable       acceptable    1
```

```
#Plot Word Frequencies
#Plot words that appear at least 50 times.
Plot <- ggplot(subset(wordfreq, freq>25), aes(x = reorder(word, -freq), y = f
req)) +
  geom_bar(stat = "identity") +
  theme(axis.text.x=element_text(angle=45, hjust=1))+
  xlab("Words")+
  ylab("Count")


Plot
```




```
#Wordcloud for Frequently occured words

#Can Change the frequency of occuran

set.seed(143)

wordcloud(names(freq),scale=c(5,0.5),colors = brewer.pal(3,"Dark2"),freq, min
.freq=15)
```


```
#Script for analysis. Author: Jeffrey Breen
#' score.sentiment() implements a very simple algorithm to estimate
#' sentiment, assigning a integer score by subtracting the number
#' of occurrences of negative words from that of positive words.
#' sentences vector of text to score
#' pos.words vector of words of postive sentiment
#' neg.words vector of words of negative sentiment
#' progress passed to <code>laply()</code> to control of progress bar.
#' data.frame
#' data.frame of text and corresponding sentiment scores

score.sentiment = function(sentences, pos.words, neg.words, .progress='none')
{
  require(plyr)
  require(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 of scores back, so we use "l" + "a" + "ply" = lapl
y:
  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)
```

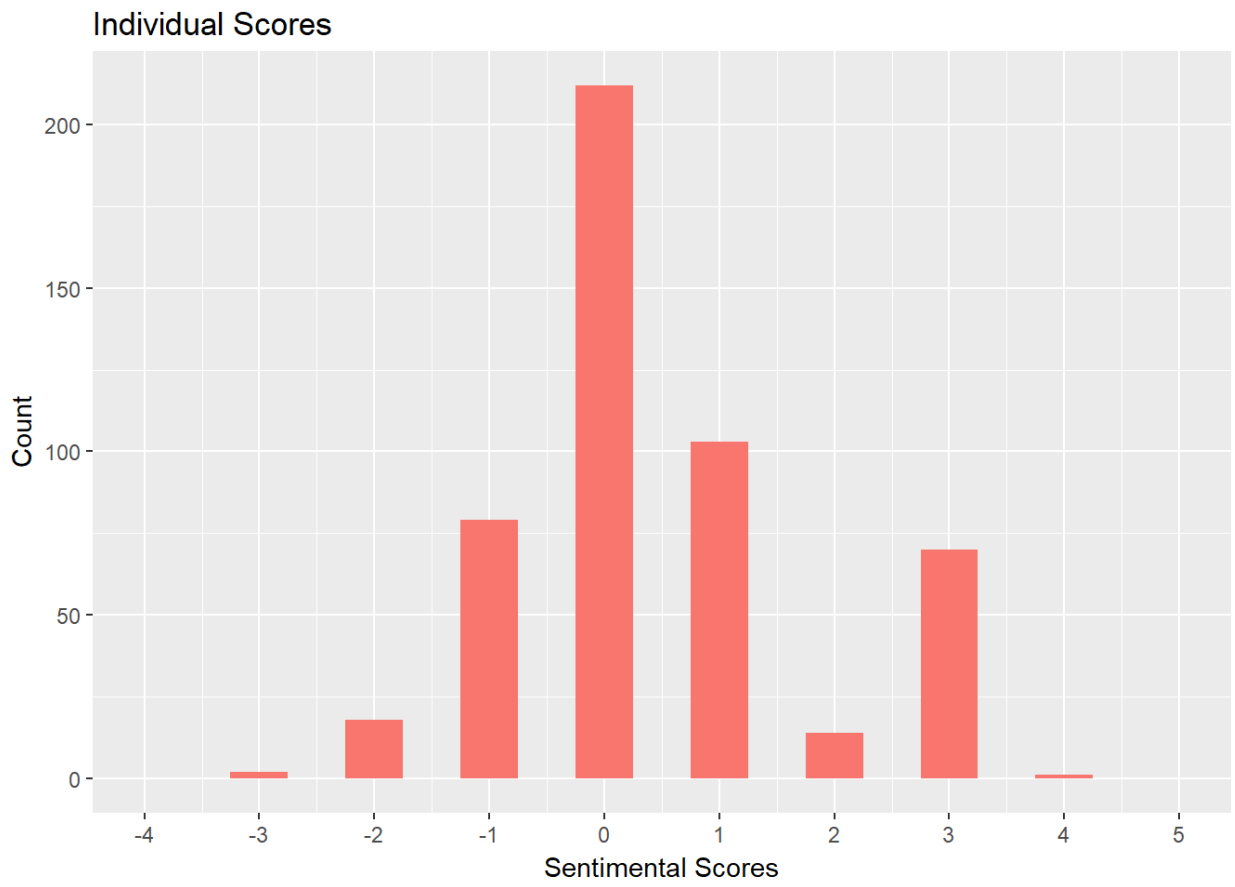
```
}


#Scoring of each words
Words <- score.sentiment(sentences  =Text,                    # Text to score
                         pos.words = pos,                     # Positive words
                         neg.words = neg)$score               # Negative words
Words[1:20]
```

```
##  [1] -2  3  1 -1  0  1  1 -1  1  0  0  0  3  0  0  1  0  0  0  0
```

```
#Scores of each words
table(Words)
```

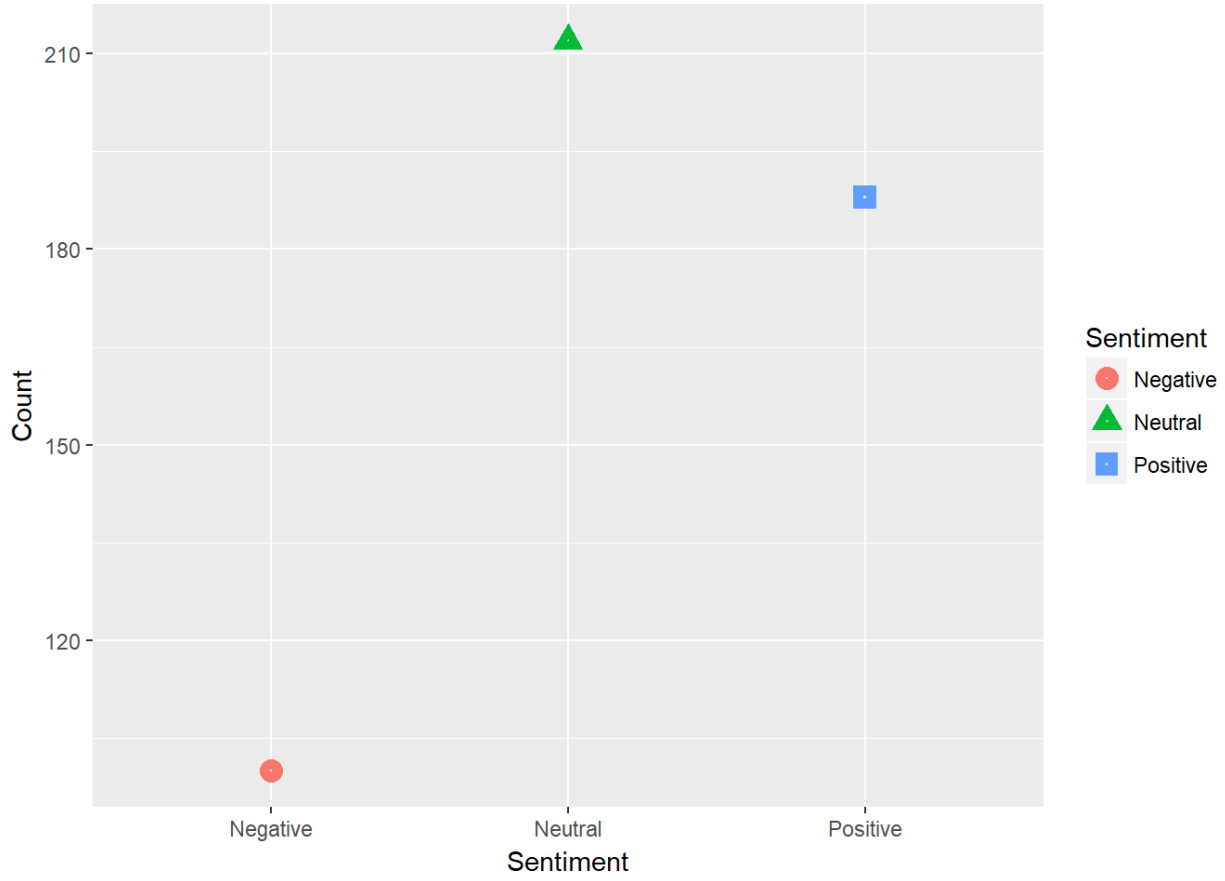
```
## Words
##  -4  -3  -2  -1   0   1   2   3   4
##   1   2  18  79 212 103  14  70   1
```

```
table.df<-as.data.frame(Words)   #Converting the words into a data frame for plotting
#Plotting Individual Scores


Indiv.Scores<-ggplot(table.df, aes(Words,fill="Red")) +
  geom_histogram(binwidth = 0.50,show.legend =FALSE)+
  ggtitle("Individual Scores")+
  xlab("Sentimental Scores")+
  ylab("Count")+
  scale_x_continuous(limits = c(-4, 5),breaks = c(-4,-3,-2,-1,0,1,2,3,4,5))


Indiv.Scores
```



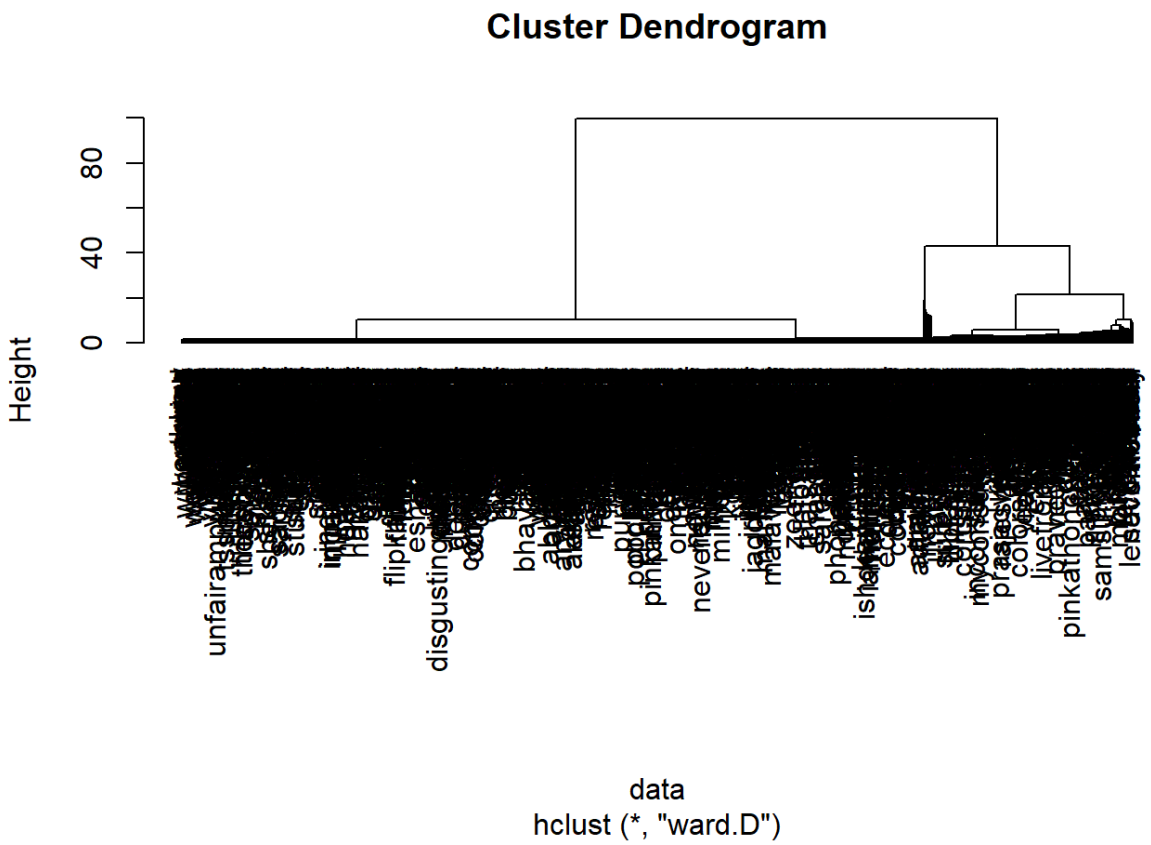

```
#Number of positive,Neutral and Negative responses

##Plotting the responses

neutral <- length(which(Words == 0))      #Words with scores equal to 0 are c
onsidered as Neutral

positive <- length(which(Words > 0))      #Words with scores Greater than 0 a
re considered as Positive

negative <- length(which(Words < 0))      #Words with scores Less than 0 are
considered as Negative

Sentiment <- c("Negative","Neutral","Positive")

Count <- c(negative,neutral,positive)

output <- as.data.frame(Count,Sentiment)  #Converting the words into a data f
rame for plotting


## Visualisation of Sentiments in Bar Graph


Sentimental.Scores <- ggplot(output, aes(x =Sentiment, y =Count,shape = Senti
ment))+geom_point(aes(colour = Sentiment), size = 4) +

  geom_point(colour = "grey90", size = 0.1)
```

```
Sentimental.Scores
```

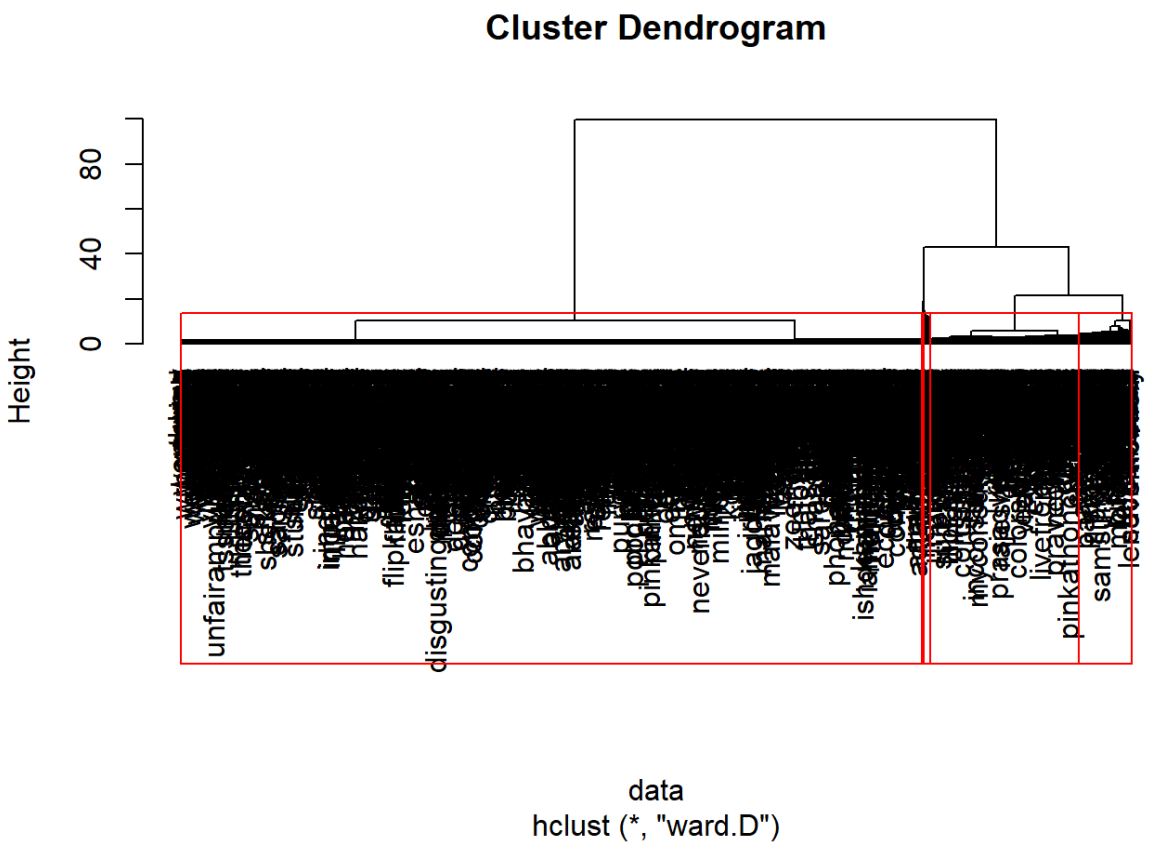



```
#Clustering for Simlilarity
##Hierarchical clustering.

data <- dist(t(dtm), method="euclidian")
model <- hclust(d=data, method="ward.D")
model
##
## Call:
## hclust(d = data, method = "ward.D")
##
## Cluster method   : ward.D
## Distance         : euclidean
## Number of objects: 1535
```

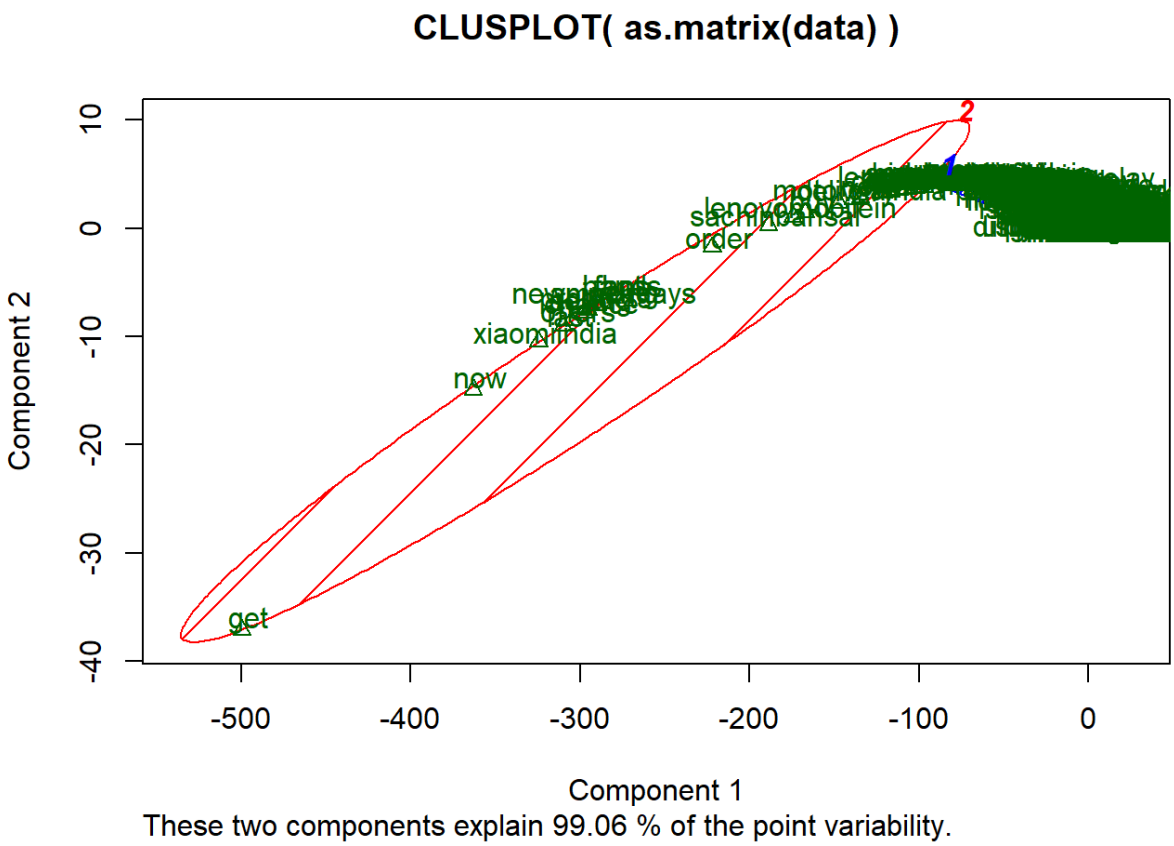
```
plot(model, hang=-1)
```




```
# To clear the plot screen for a new plot


plot.new()

plot(model, hang=-1)

groups <- cutree(model, k=6)    # "k=" defines the number of clusters you are using

rect.hclust(model, k=6, border="red") # draws dendogram with red borders around the 6 clusters
```




```
##clusplot

##Bivariate Cluster Plot (of a Partitioning Object) using the clusplot

##The clusplot uses PCA to draw the data. It uses the first two principal com
ponents to explain the data.


kfit <- kmeans(data, 2)

clusplot(as.matrix(data), kfit$cluster, color=T, shade=T, labels=2, lines=0)
```


CLUSPLOT(as.matrix(data))
Component 2
10
0
-10
-20
-30
-40
-500
-400
-300
-200
-100
0
Component 1
These two components explain 99.06 % of the point variability.
2
1
order
xiaomiindia
now
get