Sentiment Analysis

R is used for sentiment analysis. We will utilise the various text packages to evaluate the data and provide scores for the related terms in the dataset. The ultimate goal is to create a sentiment analysis model that can recognise whether words are good or negative, as well as their magnitude. The code is separated into many sections: importing data, creating a corpus, cleaning text, creating a term-document matrix, visualisation, and sentiment analysis.

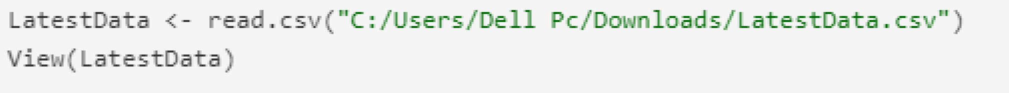
The following main packages are used in this article

- tm for text mining operations like removing numbers, special characters, punctuations and stop words (Stop words in any language are the most commonly occurring words that have very little value for NLP and should be filtered out
- word cloud for generating the word cloud plot.
- syuzhet for sentiment scores and emotion classification
- ggplot2 for plotting graphs

What is Sentiment Analysis?

Sentiment Analysis is the technique of obtaining opinions with various ratings such as positive, negative, or neutral. You may use sentiment analysis to determine the nature of an opinion or a statement in a text. Sentiment Analysis is a sort of categorization in which data is categorised as positive or negative, joyful, sad, furious, and so on.

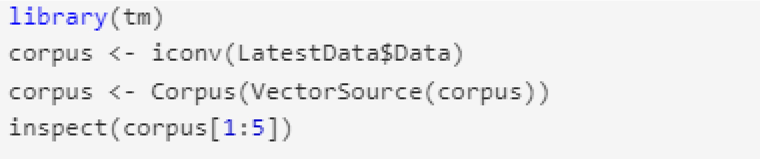
Getting data



```
LatestData <- read.csv("C:/Users/Dell Pc/Downloads/LatestData.csv")
View(LatestData)
```

Build corpus

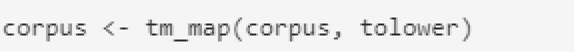
After we've loaded the dataset into R, we'll use that Vector or text data to create a Corpus. In R, we can do the same thing using the tm package.



```
library(tm)
corpus <- iconv(LatestData$Data)
corpus <- Corpus(VectorSource(corpus))
inspect(corpus[1:5])
```

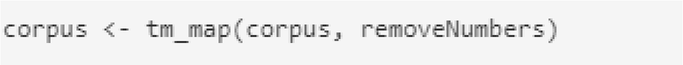
Clean text

To begin the cleaning procedure, we must change all of the text to lower case. Text may be converted to lower case using the tm map function.



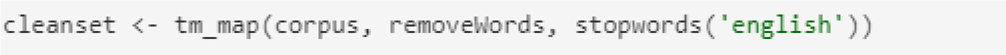
```
corpus <- tm_map(corpus, tolower)
```

Text data cleaning One of the most important aspects is the removal of special characters from the text. This is accomplished by replacing all special characters with the tm map() function.



```
corpus <- tm_map(corpus, removeNumbers)
```

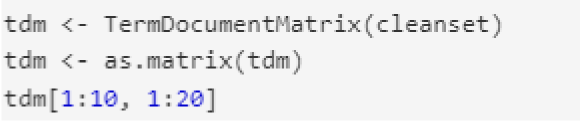
Because numerals are frequently seen in text data, we must eliminate them.
Stop words are the most common words in a language and have limited value when it comes to extracting relevant information from text. Before analysing the text, all stopwords must be removed. Stop words have meanings such as "the, is, at, or on." The tm map() function's stopwords support numerous languages, including English, French, German, Italian, and Spanish.



```
cleanset <- tm_map(corpus, removeWords, stopwords('english'))
```

Term document matrix

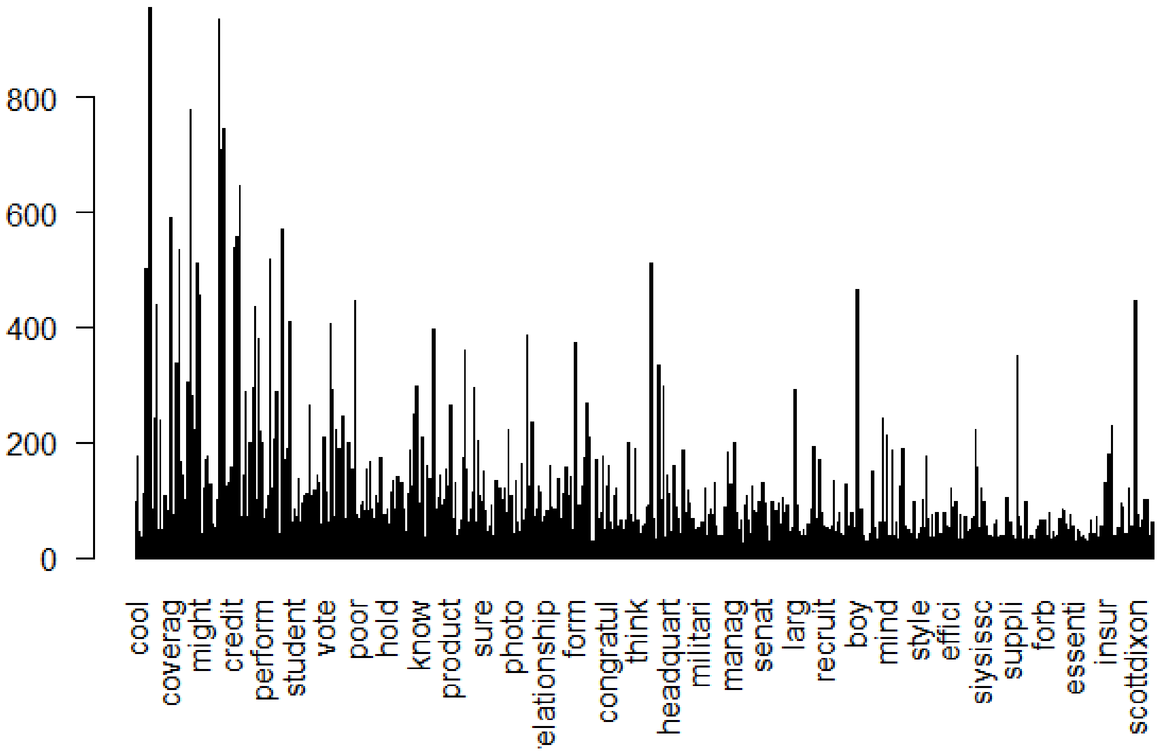
Following the cleaning of the textual content data, the next step is to count the occurrences of each term in order to identify popular or trending subjects. Using the textual content mining package's function TermDocumentMatrix(), you may create a Document Matrix — a table containing the frequency of terms.



```
tdm <- TermDocumentMatrix(cleanset)
tdm <- as.matrix(tdm)
tdm[1:10, 1:20]
```

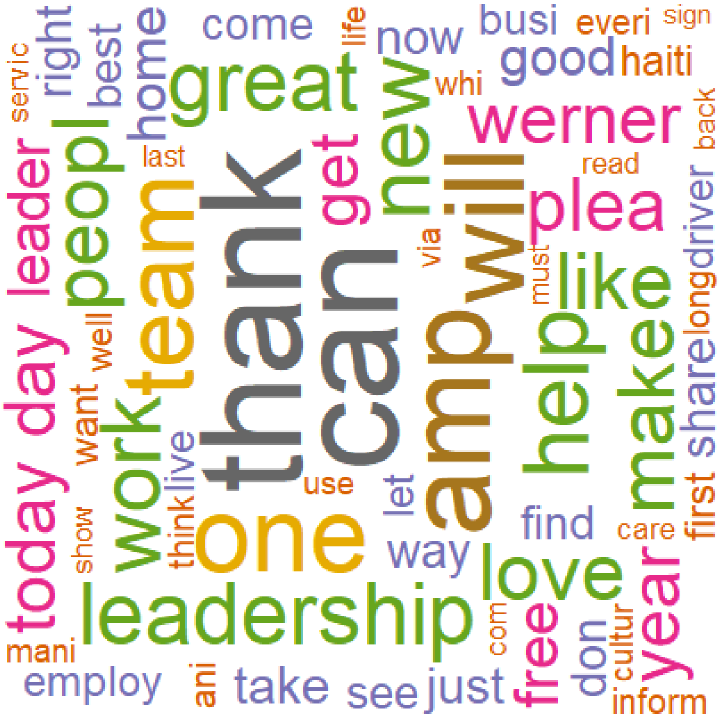
Bar plot

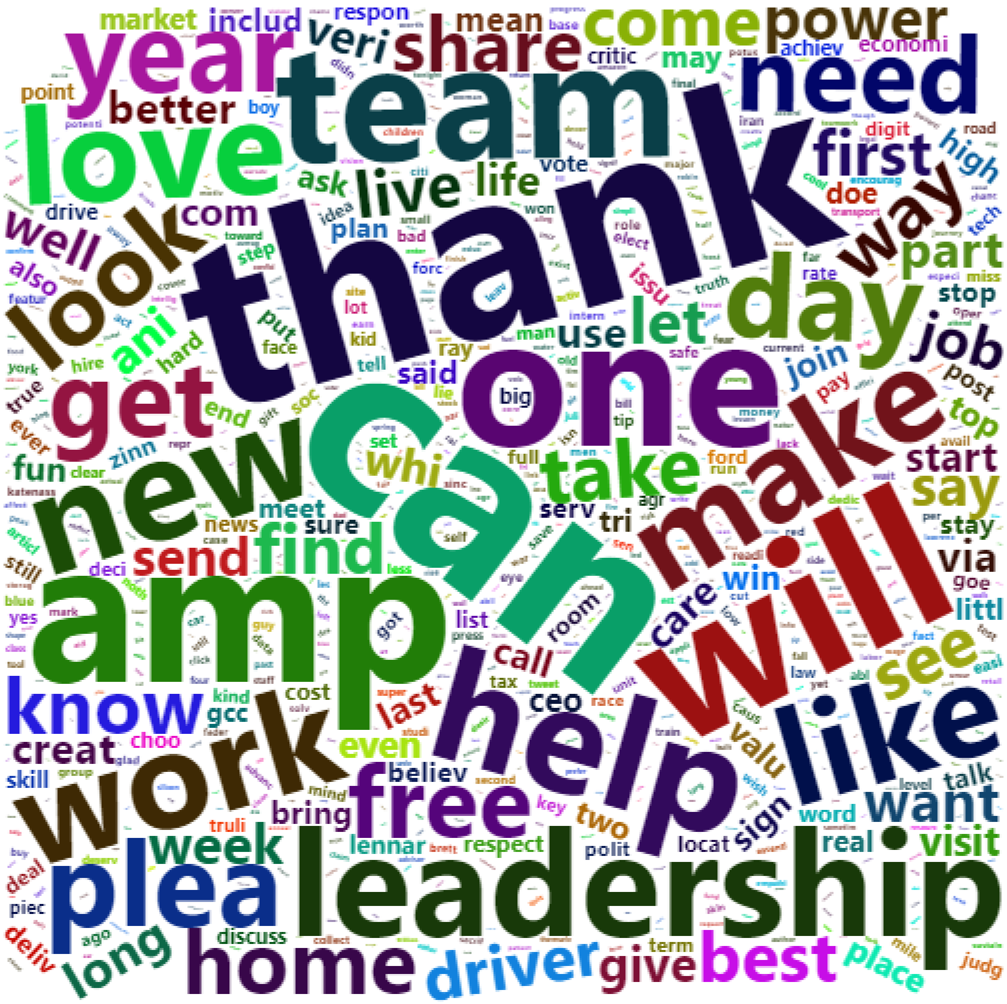
Plotting the words on a bar chart is a simple approach to view the frequent statistics for this term. You may easily make a bar chart for visualisation.



Word cloud

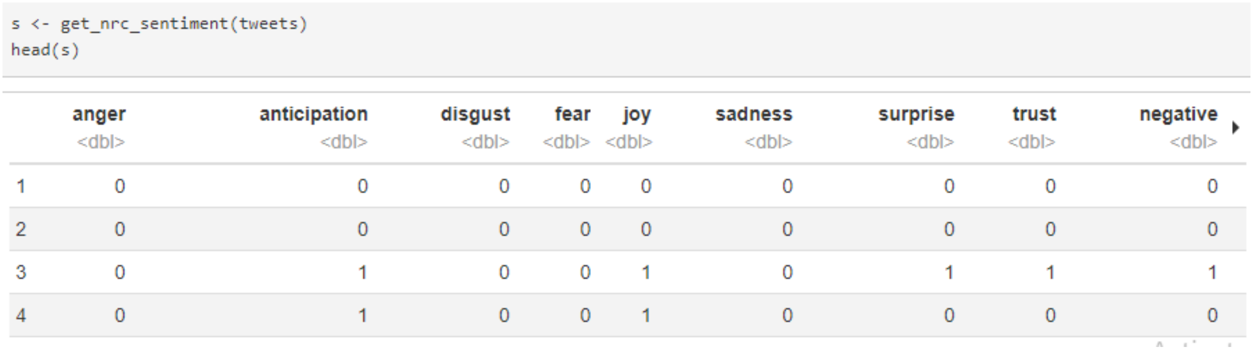
A word cloud is one of the most used methods for visualising and analysing text data. It's a graphic made out of keywords discovered inside a body of text, with the size of each word indicating its count within that text. To produce the word cloud, use the previously created word frequency data frame (table).


thank
can
amp
will
one
team
leadership
great
new
werner
plea
help
like
make
love
year
peopl
work
today
day
leader
get
right
best
home
come
life
now
busi
everi
sign
good
haiti
whi
servic
last
read
back
driver
long
share
first
care
find
way
let
use
live
think
show
want
well
via
must
free
don
cultur
inform
just
see
take
ani
employ
mani
com



Sentiment analysis

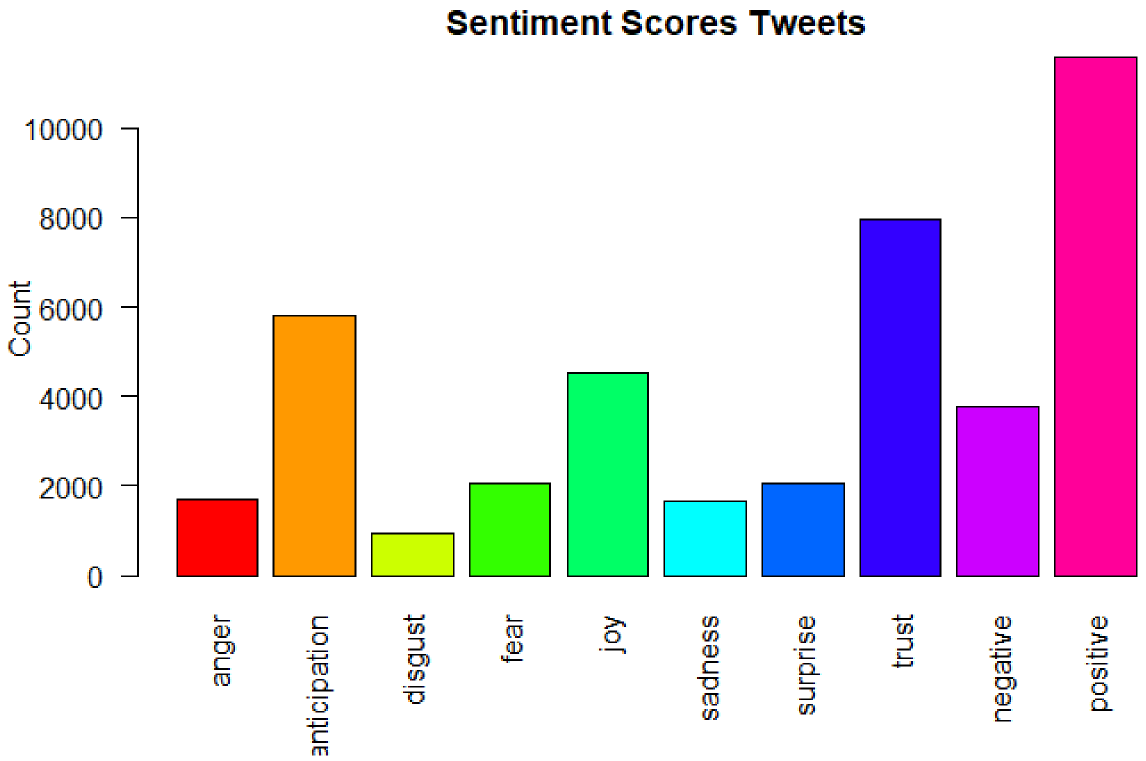
As previously said, emotions can be categorised as positive, neutral, or negative. They can also be expressed numerically to better illustrate the degree of positivity or negativity included in a body of text. This example generates sentiment scores using the Syuzhet package, which includes four sentiment dictionaries and a mechanism for accessing the sentiment extraction tool built by Stanford's NLP lab. The function get sentiment takes two arguments: a character vector (containing sentences or words) and a method. Which of the four available sentiment extraction methods will be utilised is determined by the approach chosen. Each approach employs a different scale and hence yields slightly different results.



```
s <- get_nrc_sentiment(tweets)
head(s)
```

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	1	1	1
4	0	1	0	0	1	0	0	0	0

Sentiment Score Tweets:


Sentiment Scores Tweets
Count
anger
anticipation
disgust
fear
joy
sadness
surprise
trust
negative
positive