Sentimental_Analysis

Felipe Henrique da Silva

11/04/2021

Project 1 - Social Network Sentiment Analysis

The objective of this project is to capture data from the social network Twitter and perform sentiment analysis with the captured data. For this project to be carried out, several packages must be installed and loaded.

The entire project will be described according to its stages. We will use the sentiment score calculation

Step 1 - Authentication

Below you will find the authentication process. Remember that you need to have an account created on Twitter and create an application. The steps for creating the application are detailed in the project specification.

```r
#twitter handling package
#install.packages("twitteR")
#install.packages('plyr', repos = "http://cran.us.r-project.org")
#install.packages("httr")
#install.packages("knitr")
#install.packages("rmarkdown")
#install.packages("ROAuth")
#install.packages("base64inc")
library(RCurl)
library(twitteR)
library(httr)
library(knitr)
library(rmarkdown)

#Twitter authentication method. These keys are obtained through a twitter developer account
key <- "1ycMW2rSOayAOHwf54SCvtrWc"
secret <- "6lY5wmMiqaHWK9RrBDYrqbOGkLbBWxUiZkC2XepmWkgcHbCQOV"
token <- "3331282798-Krlmm6Hdl111ZpR7JfQuRjzKRLqHci8YpAzEfM8"
tokensecret <- "anmWPG9UJyxaXJ1PgZrCtvjWslpvWS5clREmoEOEak0sS"
```

Step 2 - Connection

Below you will find the authentication process. Remember that you need to have an account created on Twitter and create an application. The steps for creating the application are detailed in the project specification.

```r
# Authentication. Answer 1 (Yes) to use direct connection. THe objective is testing the connection with
setup_twitter_oauth(key, secret, token, tokensecret)
```

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

```r
userTimeline("felipehs93")
```

```
## [[1]]
## [1] "felipehs93: teste: segundo tweet"
##
## [[2]]
## [1] "felipehs93: Teste: Primeiro Twiter"
##
## [[3]]
## [1] "felipehs93: Baixei o livro O Caminho das Apps - Da ideia ao aplicativo em 30 pÃ¡g. #caminhodasap
```

```
#word that will basis our analysis
tweetdata = searchTwitter("#BigData", n = 100)
length(tweetdata)
```

```
## [1] 100
```

```
head(tweetdata)
```

```
## [[1]]
## [1] "aProgrammerBot: RT @IainLJBrown: electronic warfare (EW) cognitive artificial intelligence (AI)
##
## [[2]]
## [1] "BuanaFrank3: RT @SuzanneC0leman: Putting Disruptive Innovations Into Action Using Artificial In
##
## [[3]]
## [1] "trendsinAI: RT @SuzanneC0leman: Artificial intelligence in federal IT - Federal News Network\n\
##
## [[4]]
## [1] "MarcoPark21: RT @IainLJBrown: electronic warfare (EW) cognitive artificial intelligence (AI) - 
##
## [[5]]
## [1] "J3nTyrell: RT @MarcoPark21: Algorithmic Training Technique Aims to Democratize Deep Learning-En
##
## [[6]]
## [1] "BuanaFrank3: RT @SuzanneC0leman: Artificial intelligence in federal IT - Federal News Network\n
```

Step 3 -Treatment of data collected through text mining

Here we will install the tm package, for text mining. We will convert the collected tweets into an object of the Corpus type, which stores data and metadata and then we will do some cleaning process, such as removing punctuation, converting the data to lowercase letters and removing stopwords (common words in the English language, in this case) .

```
#the tm package, for text mining. I will convert the collected tweets into an object
#type Corpus, which stores data and metadata and then we will do some cleaning process, such as
#remove punctuation, convert data to lowercase letters and remove stopwords (common words in the
#English language in this case).
#stop words: In computing, stop words are words which are filtered out before or after processing of
#natural language data (text).[1] Though "stop words" usually refers to the most common words in a lang
#there is no single universal list of stop words used by all natural language processing tools, and ind
#not all tools even use such a list. Some tools specifically avoid removing these stop words to support

# installing the package Text Mining.
#install.packages("tm")
#install.packages("SnowballC")
library(SnowballC)
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:httr':
##
##     content
```

```
# Treatment (cleaning, organization and transformation) of the collected data
# x$ because  the type is a List.
tweetlist <- sapply(tweetdata, function(x) x$getText())
#?gettext
#tweetlist
#Creating a Corpus (large set of text used to do statistical analysis) and applying some transformation
tweetcorpus <- Corpus(VectorSource(tweetlist))
tweetcorpus <- tm_map(tweetcorpus, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(tweetcorpus, removePunctuation): transformation
## drops documents
```

```
tweetcorpus <- tm_map(tweetcorpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(tweetcorpus, content_transformer(tolower)):
## transformation drops documents
```

```
tweetcorpus <- tm_map(tweetcorpus, function(x)removeWords(x, stopwords()))
```

```
## Warning in tm_map.SimpleCorpus(tweetcorpus, function(x) removeWords(x,
## stopwords())): transformation drops documents
```

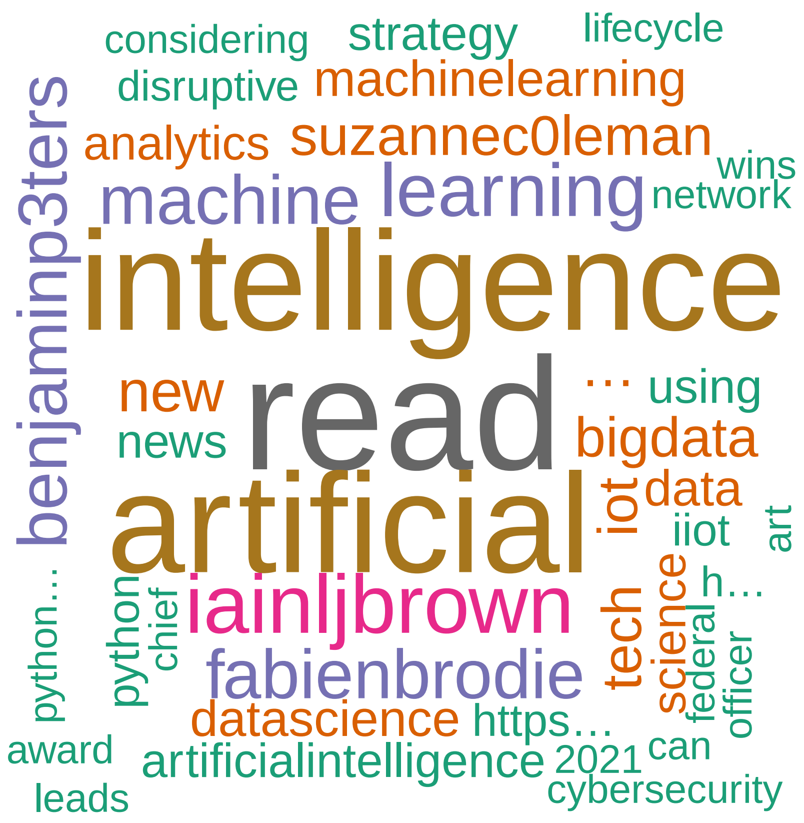
```
# Converting the Corpus object to plain text (unfortunately this code snippet make an error ahead )
#tweetcorpus <- tm_map(tweetcorpus, PlainTextDocument)
```

Step 4 -Treatment of data collected through text mining

Let's create a word cloud (wordcloud) to check the relationship between the words that occur most frequently. We create a table with the frequency of the words and then generate a dendrogram, which shows how the words relate and are associated with the main theme (in our case, the term BigData).

```
# Installing the wordcloud package
#install.packages("RColorBrewer")
#install.packages("wordcloud")
library(RColorBrewer)
library(wordcloud)
# Generating a word cloud
pal2 <- brewer.pal(8,"Dark2")
wordcloud(tweetcorpus,
          min.freq = 4,
          scale = c(5,1),
          random.color = F,
          max.word = 60,
          random.order = F,
          colors = pal2)
```

```
## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : 100daysofcode could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : gppulipaka could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : biospace could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : excellence could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : genuity could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : c4israet could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : emerging could not be fit on page. It will not be plotted.

## Warning in wordcloud(tweetcorpus, min.freq = 4, scale = c(5, 1), random.color =
## F, : natos could not be fit on page. It will not be plotted.
```



```
# Converting the text object to the matrix format
tweettdm <- TermDocumentMatrix(tweetcorpus)
tweettdm
```

```
## <<TermDocumentMatrix (terms: 550, documents: 100)>>
## Non-/sparse entries: 1199/53801
## Sparsity           : 98%
## Maximal term length: 23
## Weighting          : term frequency (tf)
```

```r
# Finding the words that appear most often
findFreqTerms(tweettdm, lowfreq = 11)
```

```
##  [1] "artificial"     "iainljbrown"    "intelligence"   "read"
##  [5] "suzannec0leman" "..."             "bigdata"        "new"
##  [9] "iot"            "tech"           "learning"       "benjaminp3ters"
## [13] "fabienbrodie"   "machine"
```

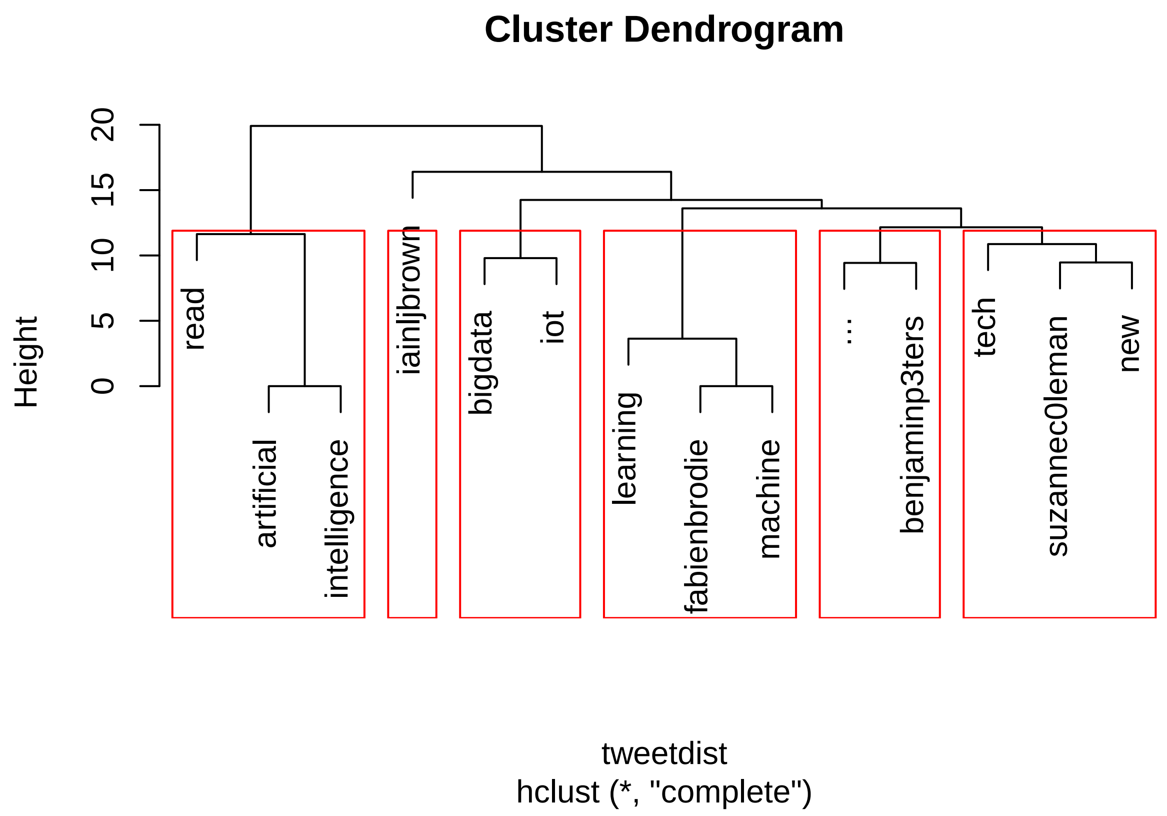
```r
# Seeking associations
findAssocs(tweettdm, 'datascience', 0.60)
```

```
## $datascience
##          iiot   considering cybersecurity    gppulipaka     lifecycle
##          0.66          0.65          0.65          0.65          0.65
##       python...
##          0.65
```

```r
# Removing sparse terms (not used often)
tweet2tdm <-removeSparseTerms(tweettdm, sparse = 0.9)

# Scaling the data
tweet2tdmscale <- scale(tweet2tdm)

# Distance Matrix
tweetdist <- dist(tweet2tdmscale, method = "euclidean")
# Preparing the dendrogram
tweetfit <- hclust(tweetdist)
# Creating the dendrogram (checking how words are grouped together)
plot(tweetfit)
# checking groups
cutree(tweetfit, k = 6)
```

```
##     artificial    iainljbrown   intelligence           read suzannec0leman
##              1              2              1              1              3
##            ...        bigdata            new            iot           tech
##              4              5              3              5              3
##       learning benjaminp3ters   fabienbrodie        machine
##              6              4              6              6
```

```r
# Viewing the word groups in the dendrogram
rect.hclust(tweetfit, k = 6, border = "red")
```



Step 5 -Sentiment Analysis

Now we can proceed with the sentiment analysis. We built a function (called sentimento.score) and a list of positive and negative words (these lists are part of this project). Our function checks each item in the data set and compares it with the provided word lists and from there calculates the feeling score, being positive, negative or neutral.

```
# Creating a function to evaluate the feeling
 #install.packages("stringr")
#Used to create small subsets of a large set and then combine the results
# install.packages("plyr")
library(stringr)
library(plyr)
```

```
##
## Attaching package: 'plyr'

## The following object is masked from 'package:twitteR':
##
##     id
```

```
felling.score = function(sentences, pos.words, neg.words, .progress = 'none')
{
  # Creating an score array with lapply
  scores = laply(sentences,
                 function(sentence, pos.words, neg.words)
                 {
                   #replace Punctuation character: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ .
```

```r
                    #| } ~ to ""
                    sentence = gsub("[[:punct:]]", "", sentence)
                    # replace Control characters to ""
                    sentence = gsub("[[:cntrl:]]", "", sentence)
                    #replace one or a sequence of digits to ''
                    sentence =gsub('\\d+', '', sentence)
                    #try to lower function
                    tryTolower = function(x)
                    {
                      y = NA
                      try_error = tryCatch(tolower(x), error=function(e) e)
                      if (!inherits(try_error, "error"))
                        y = tolower(x)
                      return(y)
                    }
                    #apllying the sentence to lower function
                    sentence = sapply(sentence, tryTolower)
                    #split string when encounter one or more spaces
                    word.list = str_split(sentence, "\\s+")
                    #changing the words.list to a vector
                    words = unlist(word.list)
                    #returns a vector of the positions of (first) matches of its first argument in its s
                    pos.matches = match(words, pos.words)
                    #print(pos.matches)
                    neg.matches = match(words, neg.words)
                    #print(neg.matches)
                    #is.na check the NA value and return True if exists and denying the result
                    pos.matches = !is.na(pos.matches)
                    #print(pos.matches)
                    neg.matches = !is.na(neg.matches)
                    #print(neg.matches)
                    #sum of true and false value
                    score = sum(pos.matches) - sum(neg.matches)
                    return(score)
                  }, pos.words, neg.words, .progress = .progress )
  scores.df = data.frame(text = sentences, score = scores)
  return(scores.df)
}
# Mapping positive and negative words
pos = readLines("palavras_positivas.txt")
neg = readLines("palavras_negativas.txt")
# Creating test data
test = c("Big Data is the future", "awesome experience",
          "analytics could not be bad", "learn to use big data")
# Testing the function on our dummy data set
testefelling = felling.score(test, pos, neg)
testefelling
```

```
##                         text score
## 1     Big Data is the future     0
## 2         awesome experience     1
## 3 analytics could not be bad    -1
## 4      learn to use big data     0
```

```
class(testefelling)
```

```
## [1] "data.frame"
```

```
# Checking the score
# 0 - expression has no word in our positive and negative word lists or
# found a negative and a positive word in the same sentence
# 1 - expression has a word with a positive connotation
# -1 - expression has a negative connotation word
testefelling$score
```

```
## [1]  0  1 -1  0
```

Step 6 - Generating Score of Sentiment Analysis

With the calculated score, we will separate by country, in this case Canada and the USA, as a way to compare sentiment in different regions. We then generate a boxplot and a histogram using the lattice package.

```
# Tweets per country
uktweets = searchTwitter("uk", n = 500, lang = "en")
head(uktweets)
```

```
## [[1]]
## [1] "liyaye__: \"5-year-old Merhawit Weldegebreal was shot in her leg. Her uncle, Abrha Zenebe, died
##
## [[2]]
## [1] "Trishool14: RT @PottskyRob: Everyone right now in the UK trying to see who that was in the pictu
##
## [[3]]
## [1] "RSVPMagazine: Harry is home\n\nhttps://t.co/rREXx156O0"
##
## [[4]]
## [1] "darshanleslie: RT @wehaveagronk: Ryan Pilkington breathes : \n\nThe UK : \n\n#LineofDuty https:/
##
## [[5]]
## [1] "theni9htOwl: RT @sjchapm: Everyone in the UK trying to see who was on the next page. #LineOfDut
##
## [[6]]
## [1] "Leslie_2348: RT @RagNBoneMan: Anywhere Away From Here is #1 on @bigtop40! @pink <U+0001F43A>\nC
```

```
usatweets = searchTwitter("usa", n = 500, lang = "en")
head(usatweets)
```

```
## [[1]]
## [1] "tulliveer: RT @havingagiraf: Megain was able to fly to the USA &amp; Back whilst 31wks pregnant
##
## [[2]]
## [1] "Ep500Mu: RT @MyBeauDes: A tunnel made out of trees\n<U+200B><U+0001F4CD> Cypress Tree Tunnel, C
##
## [[3]]
## [1] "MeronSo98061069: RT @BashaDesta: Watch out!\n\nIf the UN, EU, USA and other concerned bodies ar
##
## [[4]]
## [1] "uStarKing: @Shockuhfy You both born in USA ? or somebody is mexican like ur Mom?"
##
## [[5]]
## [1] "BARONMOND: RT @skeppy1586: Tiwa savage and Davido released park well, the song tank on all plati
##
## [[6]]
## [1] "adi7anand: RT @usacricket: <U+0001F5E8><U+FE0F>@CricFanUSA <U+0001F64C>\n\n\"What a week it's be
```

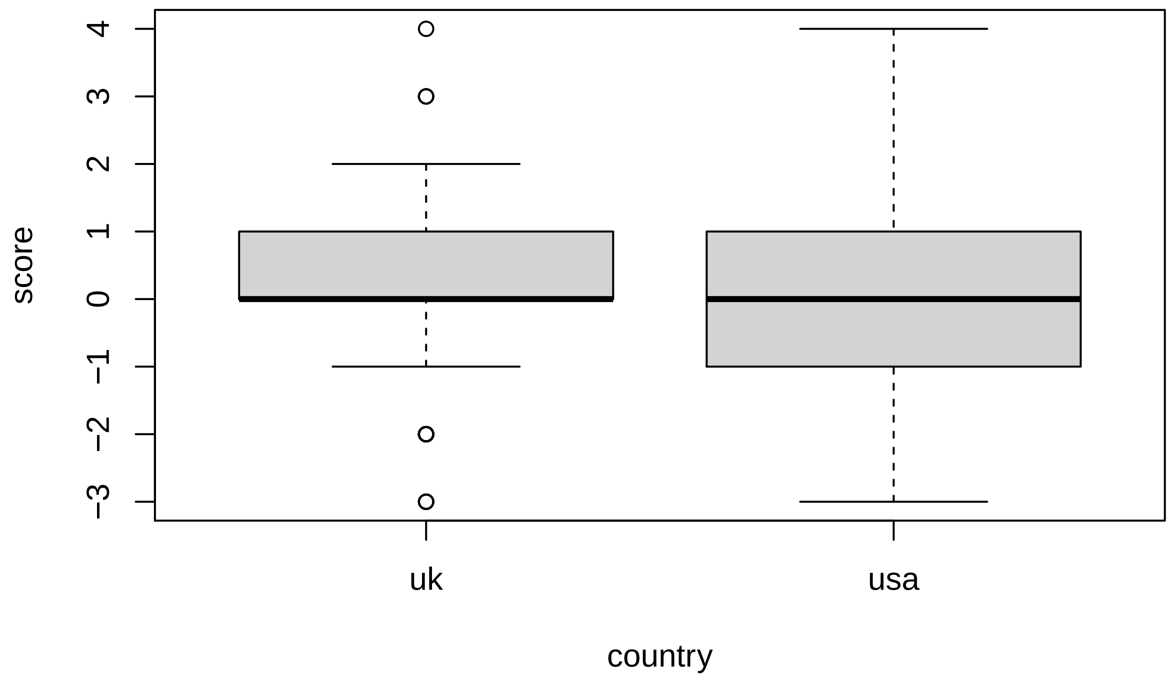
```
# getting text
uktxt = sapply(uktweets, function(x) x$getText())
usatxt = sapply(usatweets, function(x) x$getText())
# country tweets vector
cntTweet = c(length(uktxt), length(usatxt))
# gathering the texts
country = c(uktxt, usatxt)
# Applying function to calculate the felling score
scores = felling.score(country, pos, neg, .progress = 'text')
```

```
##   |                                                                            |
```

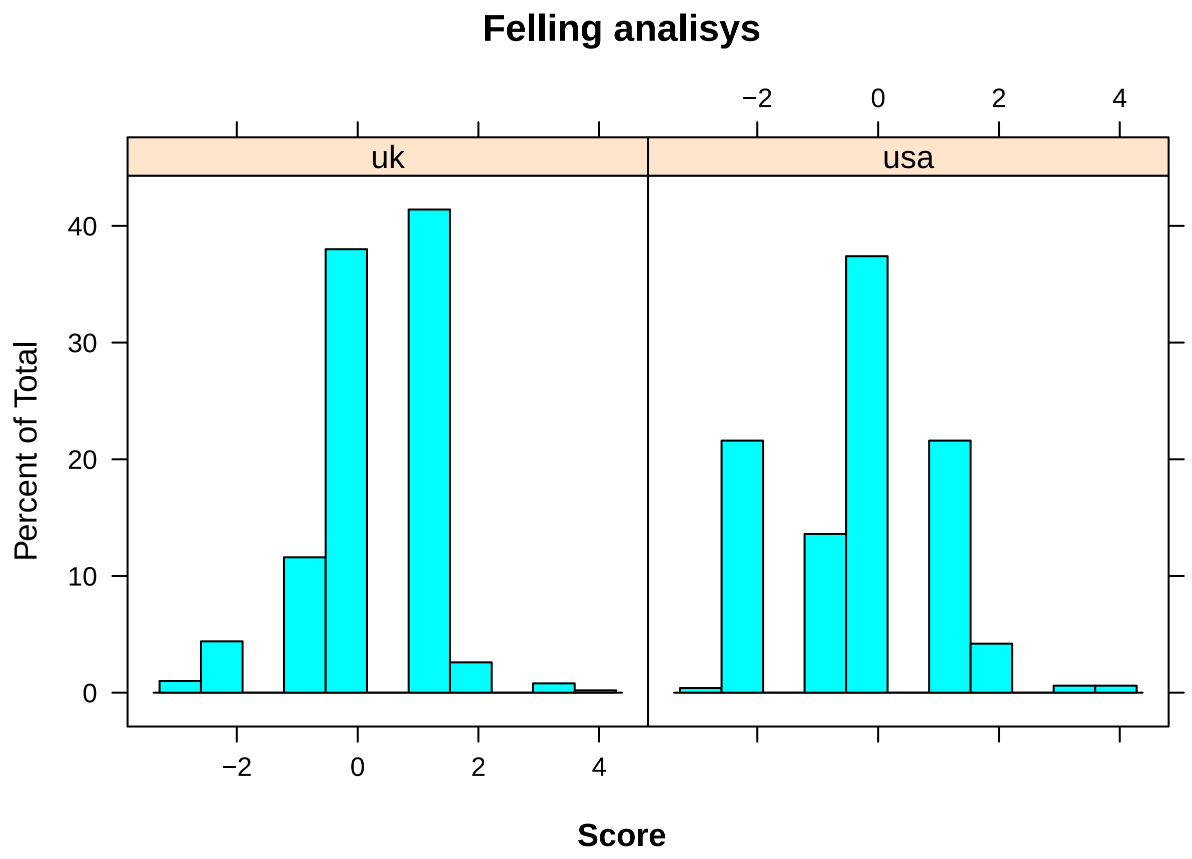
```
# Calculating the score by country
scores$country = factor(rep(c("uk", "usa"), cntTweet))
scores$very.pos = as.numeric(scores$score >= 1)
scores$very.neg = as.numeric(scores$score <= -1)
# Calculating the total
numpos = sum(scores$very.pos)
numneg = sum(scores$very.neg)
# global Score
global_score = round( 100 * numpos / (numpos + numneg) )
head(scores)
```

```
##
## 1               "5-year-old Merhawit Weldegebreal was shot in her leg. Her uncle, Abrha Zenebe, died 
## 2              RT @PottskyRob: Everyone right now in the UK trying to see who that was in the picture
## 3
## 4                                                         RT @wehaveagronk: Ryan Pilkington breathes 
## 5                                                    RT @sjchapm: Everyone in the UK trying to see who was
## 6 RT @RagNBoneMan: Anywhere Away From Here is #1 on @bigtop40! @pink <U+0001F43A>\nCheers @itswillman
##   score country very.pos very.neg
## 1    -1      uk        0        1
## 2     1      uk        1        0
## 3     0      uk        0        0
## 4     0      uk        0        0
## 5     0      uk        0        0
## 6     0      uk        0        0
```

```
boxplot(score ~ country, data = scores)
# Generating a histogram with the lattice
# install.packages("lattice")
library("lattice")
```



```
histogram(data = scores, ~score|country, main = "Felling analisys", xlab = "", sub = "Score")
```


Felling analisys
uk
usa
Percent of Total
0
10
20
30
40
−2
0
2
4
Score

End