DATA 607: Final Project

Election Contribution Analysis and Twitter Sentiment Analysis

Dilip Ganesan, 2017

Contents

Abstract

In this final project for DATA 607 we are going to use the Federal Election Commission data set to research the campaign contributions and expenditures for election 2016.

We are going to analyze the data of 2 major party candidates

Further would like to do some analysis on how the Political Action Committes have spend their amounts in 2016.

Would like to have a sentiment analysis of Hillary Clintons tweets especially during the Comey Letter episode.

Environment Preparation

Data Acquisition :

We got the data for our first analysis from FEC website. These files are pretty huge files containing lots of individual contributor information. The two separate files alond with PAC expenditure files are loaded.

```
# The HRC DataSet to data.frame

hrccontributions=fread('HRC_Cont.csv')
```

```
##
Read 5.4% of 3506081 rows
Read 15.1% of 3506081 rows
Read 23.1% of 3506081 rows
Read 31.1% of 3506081 rows
Read 41.6% of 3506081 rows
Read 53.6% of 3506081 rows
Read 67.6% of 3506081 rows
Read 83.9% of 3506081 rows
Read 85.9% of 3506081 rows
Read 3506081 rows and 19 (of 19) columns from 0.621 GB file in 00:00:14
```

```
hrccontributions=data.frame(hrccontributions)

# Calculating the Mean Contributions for HRC.
mean(hrccontributions$V10)
```

```
## [1] 147.0197
```

```
summary(hrccontributions$V10)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   -20000       15       25      147      100 12780000
```

```
# The DJT DataSet to data.frame

djtcontributions=fread('DJT_Cont.csv')
djtcontributions=data.frame(djtcontributions)

# Calculating the Mean Contributions for DJT
mean(djtcontributions$V10)
```

```
## [1] 158.8418
```

```
summary(djtcontributions$V10)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -84240.0     28.0     64.0    158.8    160.0  86940.0
```

```
# Loading Expenditures of PAC.
pacexpn=fread('FEC_independent-expenditure.csv')
pacexpn=data.frame(pacexpn)
```

Data Cleansing :

In this data munging activity, we are using regex and dplyr functions, to select only the fields which are of interest for us and removing trivial fields.

```
# Selecting only important fields. Excluding non trivial data.
hrccontributions=subset(hrccontributions,select = c(V3:V13))

# Regex to change the date formats
```

```
hrccontributions$V11=str_replace_all(hrccontributions$V11, "[:digit:]","")
hrccontributions$V11=str_replace_all(hrccontributions$V11, "-","")


# Selecting only important fields. Excluding non trivial data.
djtcontributions=subset(djtcontributions,select = c(V3:V13))

# Regex to change the date formats
djtcontributions$V11=str_replace_all(djtcontributions$V11, "[:digit:]","")
djtcontributions$V11=str_replace_all(djtcontributions$V11, "-","")


# Selecting only important fields. Excluding non trivial data.
pacexpn=subset(pacexpn,select = c(2,10,11,12,13,14))
```

Data Management :

We are using Mongo DB to load Data into NoSQL database. Some of challenges of NoSQL DB. After trying on to many packages in R, finally settled on `mongolite` Though this package seems to be very good. There is not much documentation for this package. Hence it requires lots of research to figure out the correct command for small search operation.

```
# Loading HRC contributions into a collection.
con=mongo(collection = "test", db = "test", url = "mongodb://localhost",
verbose = FALSE, options = ssl_options())

con$insert(hrccontributions)
```

```
## List of 5
##  $ nInserted  : num 3506081
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```
alldata=con$find(query = '{"V10" : { "$gt" : 2000 } }')

# Sorting of Max Contribution. With Out Indexing
top10 = con$find('{"V10" : { "$gt" : 2000 }, "V4" : { "$ne" : "HILLARY VICTORY FUND - UNITEMIZED" }}', s
knitr::kable(top10)
```

V3	V4	V5	V6	V7	V8
Clinton, Hillary Rodham	GOCKE, THOMAS	JUPITER	FL	334691584	SELF-E
Clinton, Hillary Rodham	YOUNG, SAMUEL J.	TEHACHAPI	CA	935618652	SELF-E
Clinton, Hillary Rodham	YOUNG, SAMUEL J.	TEHACHAPI	CA	935618652	INFOR
Clinton, Hillary Rodham	HILLARY ACTION FUND - UNITEMIZED	NEW YORK	NY	101855256	
Clinton, Hillary Rodham	PROPPER, GREG	LOS ANGELES	CA	900691429	PROPI
Clinton, Hillary Rodham	PICKER, MICHAEL	SACRAMENTO	CA	958413111	STATE
Clinton, Hillary Rodham	AUSTIN, ALAN	ATHERTON	CA	940275458	N/A
Clinton, Hillary Rodham	FOX, ALAN	STUDIO CITY	CA	916042407	ACF P
Clinton, Hillary Rodham	BEAUBIEN, JAMES	SANTA MONICA	CA	904023024	LATHA
Clinton, Hillary Rodham	CARROLL, DANIEL ASHTON	SAN FRANCISCO	CA	941151125	TPG C

```
# One of the interesting things figured during usage of MongoDB is how Indexing helps in reducing the t
system.time(con$find('{"V10" : { "$gt" : 2000 }, "V4" : { "$ne" : "HILLARY VICTORY FUND - UNITEMIZED" }}
```

```
##    user  system elapsed
##   0.001   0.000   1.823
```

```
#Trying to add an index of the field contribution, To measure how quick the sort operations in happenin
con$index(add = '{"V10" : 1}')
```

```
##   v key._id key.V10  name        ns
## 1 2       1      NA  _id_ test.test
## 2 2      NA       1 V10_1 test.test
```

```
system.time(con$find('{"V10" : { "$gt" : 2000 }, "V4" : { "$ne" : "HILLARY VICTORY FUND - UNITEMIZED" }}
```

```
##    user  system elapsed
##   0.001   0.000   0.002
```

```
#Loading DJT Contribution into a new collection.
con_2=mongo(collection = "test_djt", db = "test", url = "mongodb://localhost",
verbose = FALSE, options = ssl_options())

#Top 10 Contributors.
con_2$insert(djtcontributions)
```

```
## List of 5
##  $ nInserted  : num 782711
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```
alldata_trmp = con_2$find(query = '{"V10" : { "$gt" : 2000 } } ', sort = '{"V10": -1}', limit = 10)

#Remove all data from MongoDB
con$remove('{}')
con_2$remove('{}')

#Drop the collection
con$drop()
con_2$drop()
```

Data Analysis :

The analysis that we did using MongoDB functions, we are trying to achieve the same using dplyr functions in R.

```
# Performing the operations that was done using MongoDB in R.
#HRC Campaign.
top_fifty = hrccontributions %>%
         filter(rank(desc(hrccontributions$V10))<=100)

# Removing HRC Victory Fund to find Individual contributors.
top_fifty =top_fifty %>%
      filter(top_fifty$V4 != "HILLARY VICTORY FUND - UNITEMIZED")
```

```
# Top Fifty Individual Contributor's to HRC Campaign.
knitr::kable(head(plyr::arrange(top_fifty,desc(top_fifty$V10)), n = 10))
```

V3	V4	V5	V6	V7	V8
Clinton, Hillary Rodham	GOCKE, THOMAS	JUPITER	FL	334691584	SELF-E
Clinton, Hillary Rodham	YOUNG, SAMUEL J.	TEHACHAPI	CA	935618652	SELF-E
Clinton, Hillary Rodham	YOUNG, SAMUEL J.	TEHACHAPI	CA	935618652	INFOR
Clinton, Hillary Rodham	HILLARY ACTION FUND - UNITEMIZED	NEW YORK	NY	101855256	
Clinton, Hillary Rodham	PROPPER, GREG	LOS ANGELES	CA	900691429	PROPI
Clinton, Hillary Rodham	PICKER, MICHAEL	SACRAMENTO	CA	958413111	STATE
Clinton, Hillary Rodham	AUSTIN, ALAN	ATHERTON	CA	940275458	N/A
Clinton, Hillary Rodham	FOX, ALAN	STUDIO CITY	CA	916042407	ACF P
Clinton, Hillary Rodham	BEAUBIEN, JAMES	SANTA MONICA	CA	904023024	LATHA
Clinton, Hillary Rodham	CARROLL, DANIEL ASHTON	SAN FRANCISCO	CA	941151125	TPG C

```
# DJT Campaign.
top_fifty_djt = djtcontributions %>%
         filter(rank(desc(djtcontributions$V10))<=100)


# Top Fifty Individual Contributor's to DJT Campaign.
knitr::kable(head(plyr::arrange(top_fifty_djt,desc(top_fifty_djt$V10)), n = 10))
```

V3	V4	V5	V6	V7	V8
Trump, Donald J.	BOCH, ERNIE	NORWOOD	MA	02062	BOCH AUTOMOTIVE GRO
Trump, Donald J.	FERRERO, LOUIS P MR.	CANTON	GA	30115	INFORMATION REQUEST
Trump, Donald J.	CONSERVATIVE ACTION FUND	ALEXANDRIA	VA	22314	
Trump, Donald J.	COBB, ROBERT	BIRMINGHAM	AL	35209	COBB THEATERS
Trump, Donald J.	DOBSKI, ROBERT	BLOOMINGTON	IL	61704	INFORMATION REQUEST
Trump, Donald J.	GORMAN, L.D. MR.	HAZARD	KY	41702	INFORMATION REQUEST
Trump, Donald J.	ROVT, ALEXANDER MR.	BROOKLYN	NY	11234	INFORMATION REQUEST
Trump, Donald J.	GIGANTE, PETER	BELLINGHAM	WA	98225	SELF-EMPLOYED
Trump, Donald J.	TANZER, LEONARD J MR.	SCARSDALE	NY	10583	PATIENT CARE ASSOCIAT
Trump, Donald J.	NORTHCUTT, JOHN D MR. III	FAIRHOPE	AL	36532	INFORMATION REQUEST

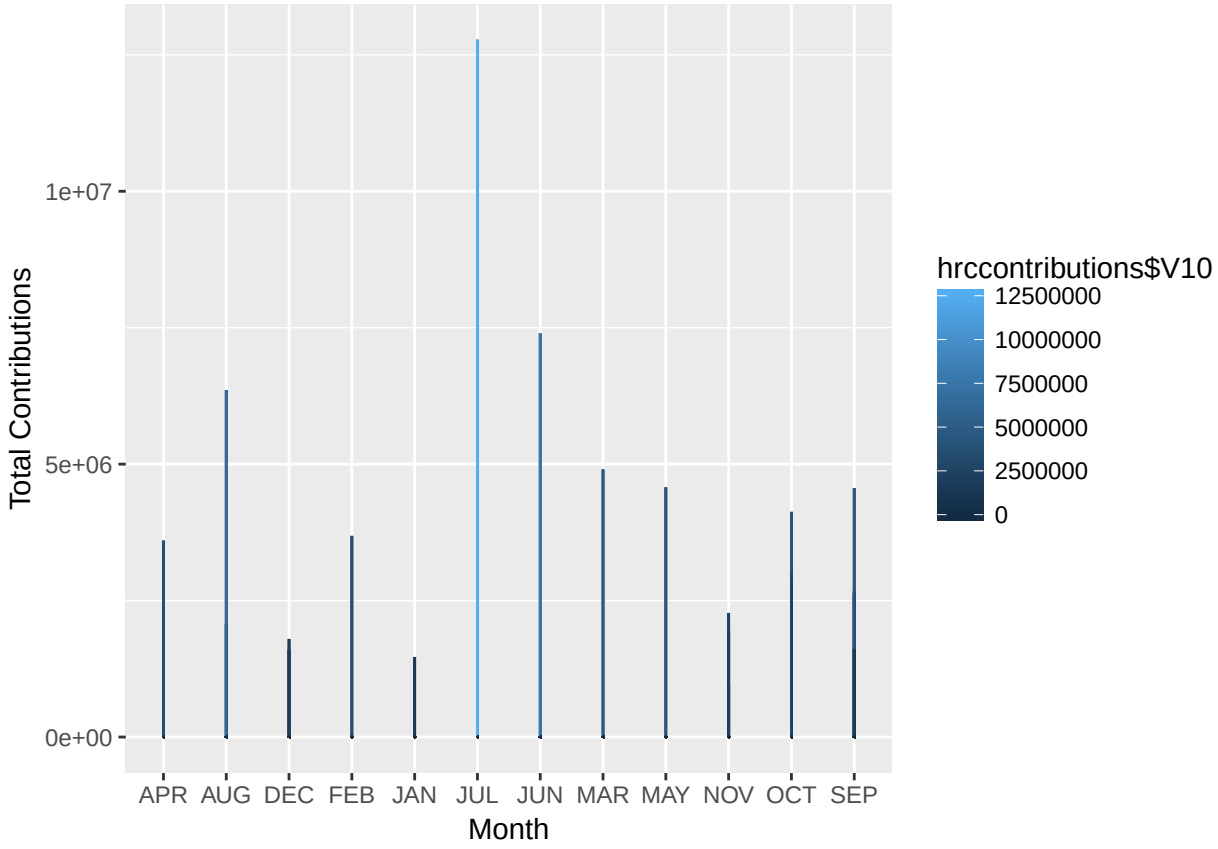
```
# PAC Expenditure Analysis
hrcpac = pacexpn[grepl("clin", pacexpn$can_nam, ignore.case = TRUE) , ]
hrcunexp  = hrcpac %>%
         filter(rank(desc(hrcpac$exp_amo))<=100)

djtpac = pacexpn[grepl("donald", pacexpn$can_nam, ignore.case = TRUE) , ]
djtunexp  = djtpac %>%
         filter(rank(desc(djtpac$exp_amo))<=100)
```

On Comparing the top expenditure of both the PACs, DJT PAC has spent lots of amount on Campaign OutReach. So they were smart enough to invest heavily on On-line campaigning.

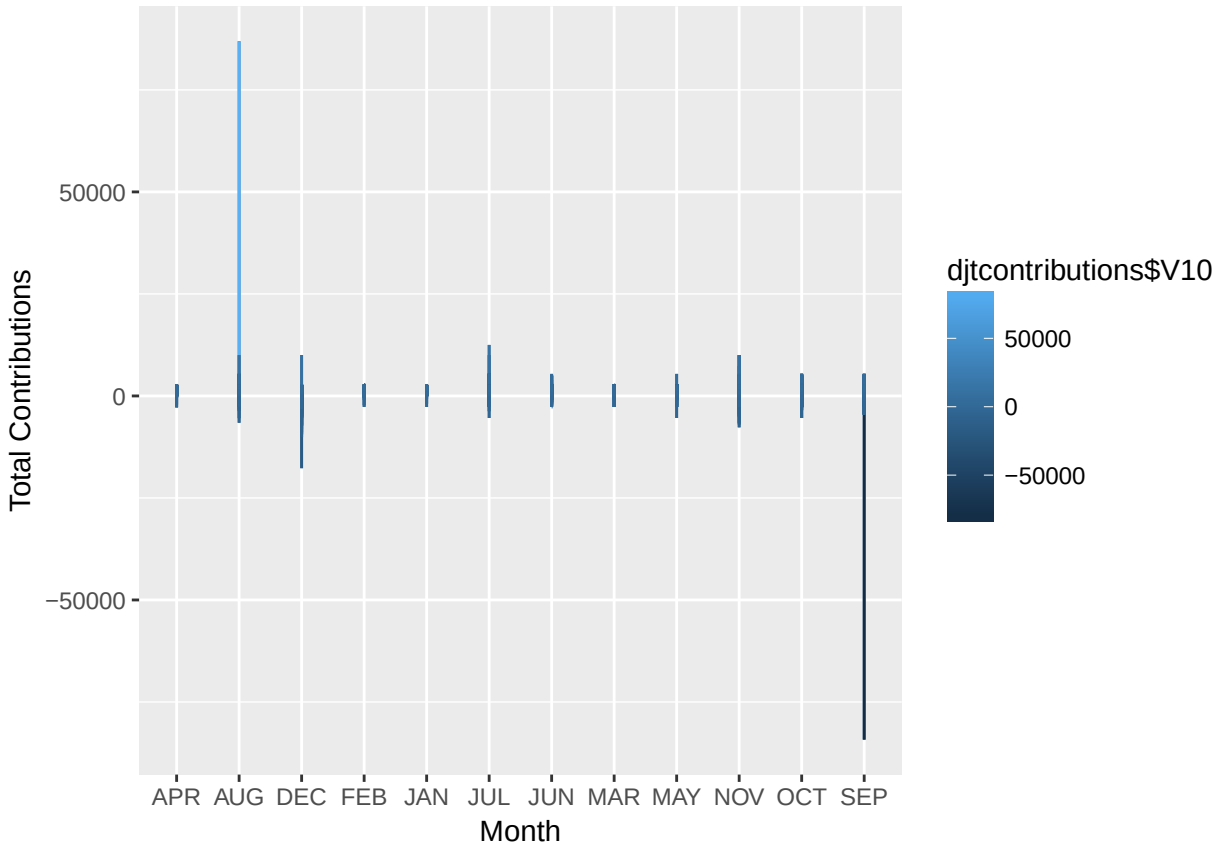
Plots:

Plots of Month wise contributions for both the candidates. From the plots it is clear for DJT more the contributions, refunds were higher.

```
ggplot(hrccontributions, aes(x=hrccontributions$V11, y=hrccontributions$V10 ,group=hrccontributions$V11
    geom_line() + ylab("Total Contributions") +
    xlab("Month")
```




```
ggplot(djtcontributions, aes(x=djtcontributions$V11, y=djtcontributions$V10 ,group=djtcontributions$V11
    geom_line() + ylab("Total Contributions") +
    xlab("Month")
```



Sentiment Analysis:

Sentiment Analysis: The aim of my twitter sentiment analysis was to see how the tweets were exchanged during Comey Letter reveal during 2016 Election. But i figured out that twitter will not be able to get the tweets that past. So to proceed with sentiment analysis. I looked for the current hashtag. Having from Commonwealth country, cricket is the only sport that we grew up with. Two of the most famous cricketers from our arch rivals team retired and twitter was abuzz about their achievements and criticsm. So i searched for tweets with tag `#MisYou` to get the tweets and do sentiment analysis.

Twitter API:

To access the twitter API I learnt that we need to create a application. Once application is created then it needs following information to access.

api_key

api_secret

access_token

access_token_secret

```
source("twitterapi.R")

options(httr_oauth_cache=TRUE)
setup_twitter_oauth(key, secret, token, tokensecret)
```

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

```
#version to just look for date range
#data2 <- searchTwitter("#MisYou", n=10000, lang="en")

#df2 <- twListToDF(data2)
# For our intial analysis downloaded and stored the files.
#write.csv(df2,file = "MisYou_2.csv")
#write.csv(df1, file="MisYou.csv")

rm(key, secret, token, tokensecret)
```

Opinion Lexicon

Next for sentiment analysis, we are loading the lexicon of both positive and negative words. Reference for this lexicon is listed below.

```
positive = scan('opinion-lexicon-English/positive-words.txt',
                  what='character', comment.char=';')
negative = scan('opinion-lexicon-English/negative-words.txt',
                  what='character', comment.char=';')


texttweet=fread('MisYou.csv')
texttweet=data.frame(texttweet)

texttweet_2=fread('MisYou_2.csv')
texttweet_2=data.frame(texttweet_2)

fulltweettext = rbind(texttweet,texttweet_2)

# Reference for this function given below.
score.sentiment <- function(sentences, good_text, bad_text, .progress='none'){
    scores = plyr::laply(sentences, function(sentence, good_text, bad_text) {

        # clean text
        sentence <- gsub('[[:punct:]]', '', sentence)
        sentence <- gsub('[[:cntrl:]]', '', sentence)
        sentence <- gsub('\\d+', '', sentence)
        sentence <- iconv(sentence, 'latin1', 'ASCII', 'byte')
        sentence <- tolower(sentence)
        words <- unlist(str_split(sentence, '\\s+'))

        # compare to lexicon
        pos.matches <- !is.na(match(words, good_text))
        neg.matches <- !is.na(match(words, bad_text))

        # calc score as difference between two
        score <- sum(pos.matches) - sum(neg.matches)

        return(score)
    }, good_text, bad_text, .progress=.progress )
    return(scores)
}
```

```
fulltweettext$score=score.sentiment(fulltweettext$text,positive,negative)

table(fulltweettext$score)
```

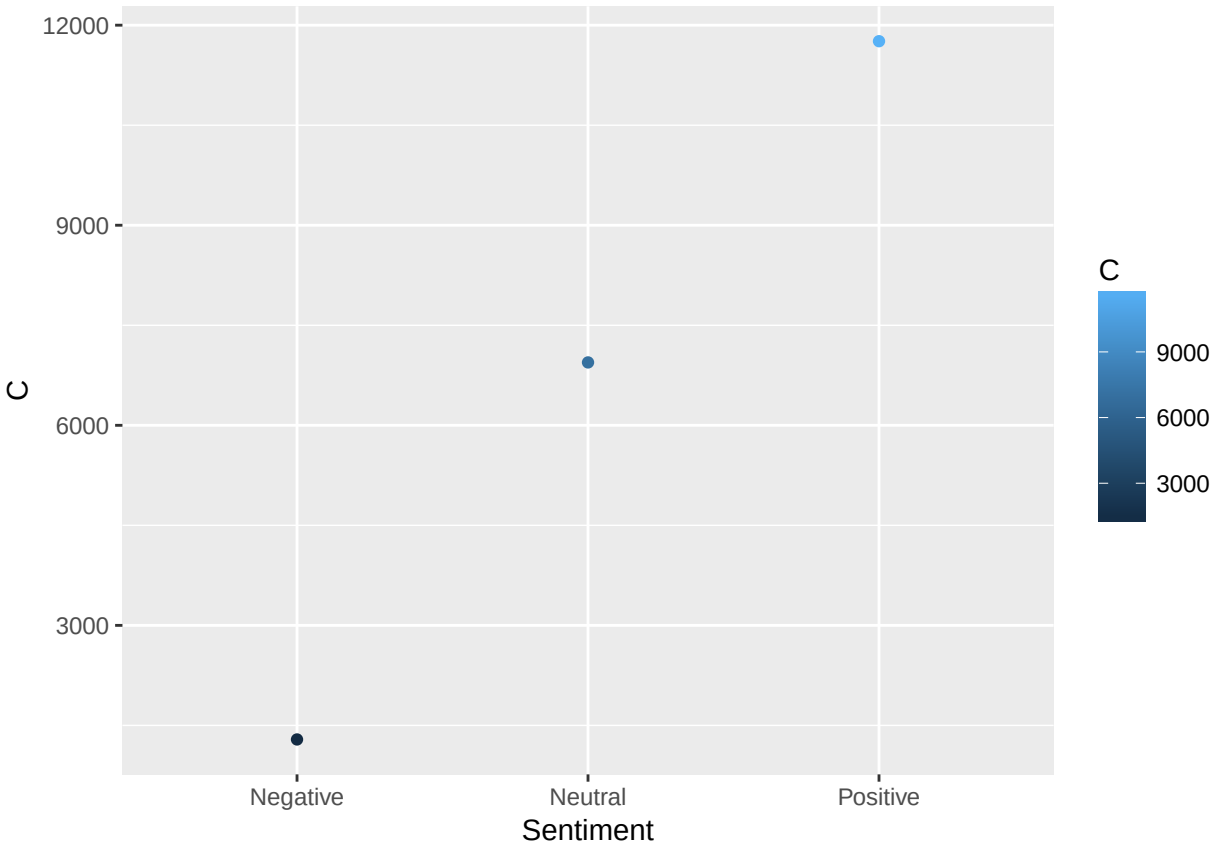
```
##
##    -5    -4    -3    -2    -1     0     1     2     3     4     5     6
##     1     1    24   139  1132  6943 10146  1212   335    52    12     3
```

Score Analysis

```
neutral = length(which(fulltweettext$score == 0))
positive = length(which(fulltweettext$score > 0))
negative = length(which(fulltweettext$score < 0))
Sentiment = c("Negative","Neutral","Positive")
C = c(1289,6943,11760)

output <- data.frame(Sentiment,C)

knitr::kable(output)
```

Sentiment	C
Negative	1289
Neutral	6943
Positive	11760

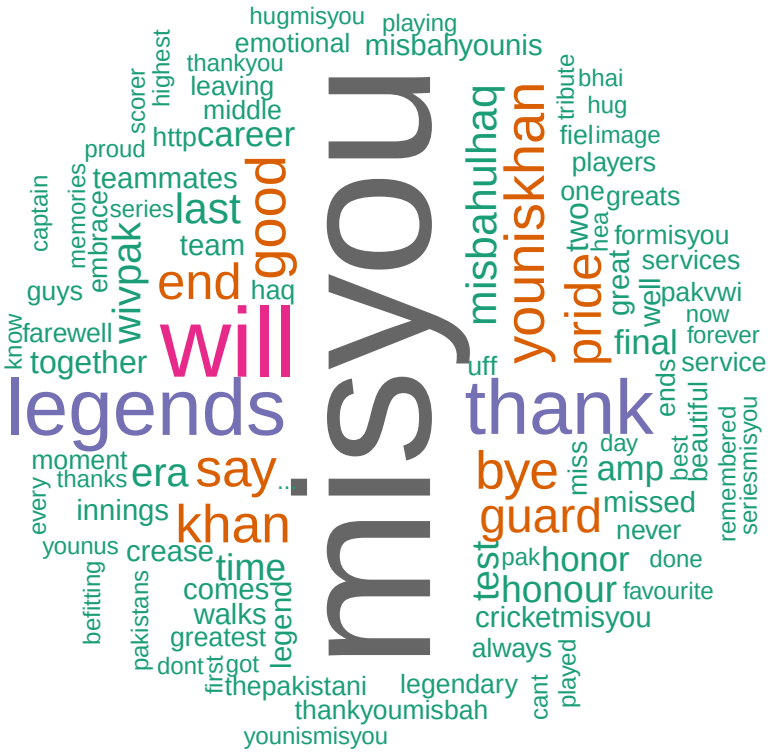
```
qplot(Sentiment,C,data=output,colour = C)
```



Word Cloud.

```
# Ignore graphical Parameters to avoid input errors
tweets.txt <- str_replace_all(fulltweettext$text,"[^[:graph:]]", " ")

# Reference for this function provided below.
clean.text = function(x)
{

   # tolower
   x = tolower(x)
   # remove rt
   x = gsub("rt", "", x)
   # remove at
   x = gsub("@\\w+", "", x)
   # remove punctuation
   x = gsub("[[:punct:]]", "", x)
   # remove numbers
   x = gsub("[[:digit:]]", "", x)
   # remove links http
   x = gsub("http\\w+", "", x)
   # remove tabs
   x = gsub("[ |\t]{2,}", "", x)
   # remove blank spaces at the beginning
   x = gsub("^ ", "", x)
   # remove blank spaces at the end
   x = gsub(" $", "", x)
   return(x)
```

```
}

cleanText <- clean.text(tweets.txt)
vector <- paste(cleanText,collapse=" ")
remwords <- c("pakistan","cricket","Misbah", "Younis", "Khan", "misbah", "younis","misbahulhaqmisyou","
vector <- removeWords(vector,c(stopwords("english"),remwords))

wordcloud(vector, scale=c(6,0.7), max.words=100,
          random.order=FALSE, rot.per=0.35,colors=brewer.pal(8,"Dark2"))
```



Conclusion:

This project was a great learning experience for me in all the technologies and languages I have used to complete it. Whether it is R or MongoDB or Twitter feed analysis, it was full of challenges and lots of research.

Reading of twitter feed is not as easy as it sounds, twitter puts heavy restrictions if we try to read the tweets from single ip. It would be better if we have a cluster server and a load balancer to retreive the tweets so the ips keep switching. I was not able to better search tweets which are like 8 months old, there should be some better way to get that tweets.

I had my own doubts on MongoDB, but it is a breeze to work on, though as far as R-Mongo integration better documentations are neccessary but working on Mongo as a whole was relatively easy.

All in all it was wonderful learning and it was a great experience.

Reference

Opinion Lexicon

Sentiment Analyzer Function

The below function is from rbloggers, which will give us the sentiment scores when we provided the text and positive and negative lexicon.

```
score.sentiment <- function(sentences, good_text, bad_text, .progress='none'){
    scores = plyr::laply(sentences, function(sentence, good_text, bad_text) {

        # clean text
        sentence <- gsub('[[:punct:]]', '', sentence)
        sentence <- gsub('[[:cntrl:]]', '', sentence)
        sentence <- gsub('\\d+', '', sentence)
        sentence <- iconv(sentence, 'latin1', 'ASCII', 'byte')
        sentence <- tolower(sentence)
        words <- unlist(str_split(sentence, '\\s+'))

        # compare to lexicon
        pos.matches <- !is.na(match(words, good_text))
        neg.matches <- !is.na(match(words, bad_text))

        # calc score as difference between two
        score <- sum(pos.matches) - sum(neg.matches)

        return(score)
    }, good_text, bad_text, .progress=.progress )
    return(scores)
}
```

Tweets Cleaner Function

```
clean.text = function(x)
{

   # tolower
   x = tolower(x)
   # remove rt
   x = gsub("rt", "", x)
   # remove at
   x = gsub("@\\w+", "", x)
   # remove punctuation
   x = gsub("[[:punct:]]", "", x)
   # remove numbers
   x = gsub("[[:digit:]]", "", x)
   # remove links http
   x = gsub("http\\w+", "", x)
   # remove tabs
   x = gsub("[ |\t]{2,}", "", x)
   # remove blank spaces at the beginning
```

```
  x = gsub("^ ", "", x)
  # remove blank spaces at the end
  x = gsub(" $", "", x)
  return(x)
}
```