MA415 Final Project- Analysis of Tweets' Hashtag About President Trump

Sibo Zhu

2017/12/15

Introduction

I believe celebrities' work can be reflected by how people talk about them. As President Trump has inaugurated for already an whole year, how people discuss about him within their social media could express their current attitude towards President Trump, and the work he has done. By utilizing the Twitter api, with all the tweets contains hashtag about Trump and the geo location info, researches about regions that interested in President Trump and related analysis would also be useful.

```
### prepare packages
library(ROAuth)
library(streamR)
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

```
## Loading required package: rjson
```

```
library(twitteR)
library(tm)
```

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

```
## Loading required package: NLP
```

```
library(SnowballC)
library(ggplot2)
```

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

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

```
library(ggvis)
```

```
##
## Attaching package: 'ggvis'
```

```
## The following object is masked from 'package:ggplot2':
##
##     resolution
```

```
library(ggmap)
library(RgoogleMaps)
library(devtools)
library(maps)
library(mapdata)
library(grid)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:twitteR':
##
##     id, location

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:maps':
##
##     ozone

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

```
library(shiny)
library(stringr)
library(dplyr)
library(wordcloud)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(topicmodels)
```

```
### setting up connections for Twitter
#requestURL <- "https://api.twitter.com/oauth/request_token"
#accessURL <- "https://api.twitter.com/oauth/access_token"
#authURL <- "https://api.twitter.com/oauth/authorize"
#api_key             <- "dSzLRlqni1W2f9CJagS19RP6b"
#api_secret          <- "pLX7gIERogxsgn7AT2eEAQlhmxGKwEx9de5pjeOtjkQDDbVVCD"
#access_token        <- "3134023545-OZ9TIdXFbEFW66HmsCUo0EWGIANJ4SgS1OgdDO7"
#access_token_secret <- "kAzcVK0EMOhCFAs13Mau2Y8q2fCYHjvA5XXl7Rd0SCO1o"


###Above codes are my personal Twitter App Keys, I shouldn't have included them in this project,
###but I just did so for your convenience, so please don't use them to do bad things

### Setting up streamR

#my_oauth <- OAuthFactory$new(consumerKey = api_key, consumerSecret = api_secret,
#requestURL = requestURL, accessURL = accessURL, authURL = authURL)
#my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))
#save(my_oauth, file = "my_oauth.Rdata")      #Save my Oauth info for future convenience
```

```
#load("my_oauth.Rdata")     #Load my Oauth file save above to avoid authentication everytime running th

#  filterStream( file="D_Tweet.json", track="realDonaldTrump",  #naming the downloaded file as "D_Tweet
#      locations=c(-74,40,-73,41), timeout=7200, oauth=my_oauth )     #restrict tweets within United St

###Since it would take 2 hours to complete the capturing, it's certainly meaningless to let this chunk r
###If you'd like to run the code yourself, you are welcome to uncomment the above lines and go for it;
###Also, due to Github's regulation of file size restriction of 100MB maximum, I cannot upload my "D_Twe
### data in that json file will be cleaned and write into "cleaned_tweets_df.csv" later
```

```
#tweets_raw.df <- parseTweets("DT_Tweet.json",verbose = FALSE)


#keep <- c("text","lang","listed_count","geo_enabled","statuses_count","followers_count",
#          "favourites_count","friends_count","time_zone","country_code","full_name",
#          "place_lat","place_lon")

#tweets.df <- tweets_raw.df[,keep];


#write.csv(tweets.df,"tweets_df.csv")

#save(tweets.df, file = "tweets.df.Rdata")      #save data for future convenience
```

```
load("tweets.df.Rdata")      #load data

tweets.df <- tweets.df[tweets.df$lang=="en",]     #filtering for only English tweets
tweets.df <- tweets.df[tweets.df$country_code=="US",]     #filtering for US region tweets
tweets.df <- tweets.df[tweets.df$geo_enabled==TRUE,]     #filtering for goelocations enabled tweets

write.csv(tweets.df,"cleaned_tweets_df.csv")     #writing cleaned data into CSV
```

```
#filtering out three typical states that worth analyzing
ca <- data.frame(filter(tweets.df, grepl(', CA', full_name)))
```

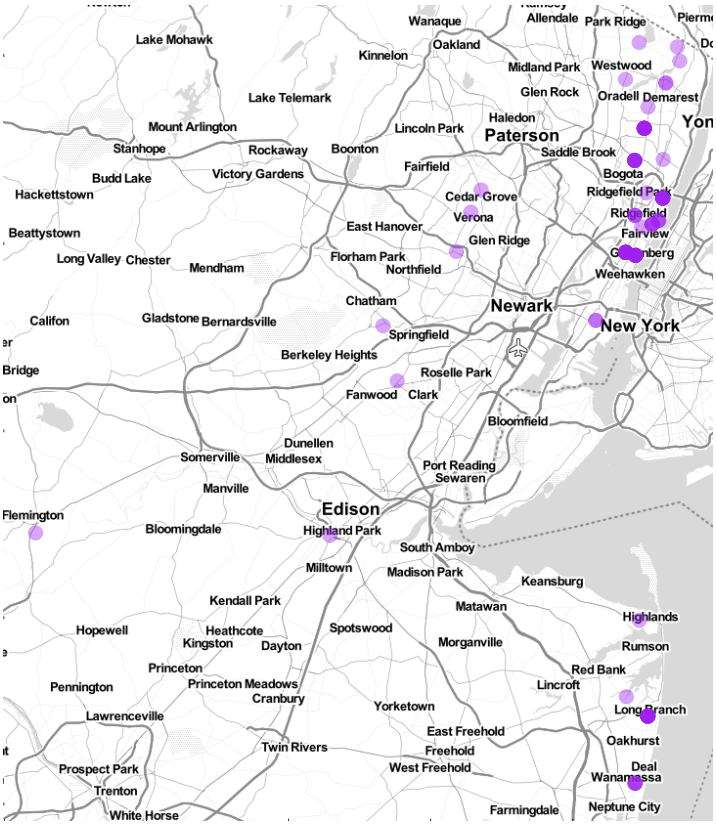
```
nj <- data.frame(filter(tweets.df, grepl(', NJ', full_name)))
ny <- data.frame(filter(tweets.df, grepl(', NY', full_name)))

###Plotting
NJplot <- qmplot(place_lon, place_lat, data = nj, colour = I('purple'), size = I(2), alpha=I(0.4),
       mapcolor = "bw", main="New Jersey")
NYplot <- qmplot(place_lon, place_lat, data = ny, colour = I('darkred'), size = I(2), alpha=I(0.4),
       mapcolor = "bw", main="New York")
CAplot <- qmplot(place_lon, place_lat, data = ca, colour = I('blue'), size = I(2), alpha=I(0.4),
       mapcolor = "bw",main="California")

###Plotting data within whole USA region
USMap <- get_map (location = c(lon = -95.71289, lat = 37.09024), zoom=4, scale=2, maptype="roadmap", so
USAplot <- ggmap(USMap) +
  geom_point(data = tweets.df, aes(x=tweets.df$place_lon,y=tweets.df$place_lat),alpha=0.4,color="red") +
  ggtitle("US Map for the Total Dataset")
```

```
NYplot
```

New York


```
NJplot
```

New Jersey



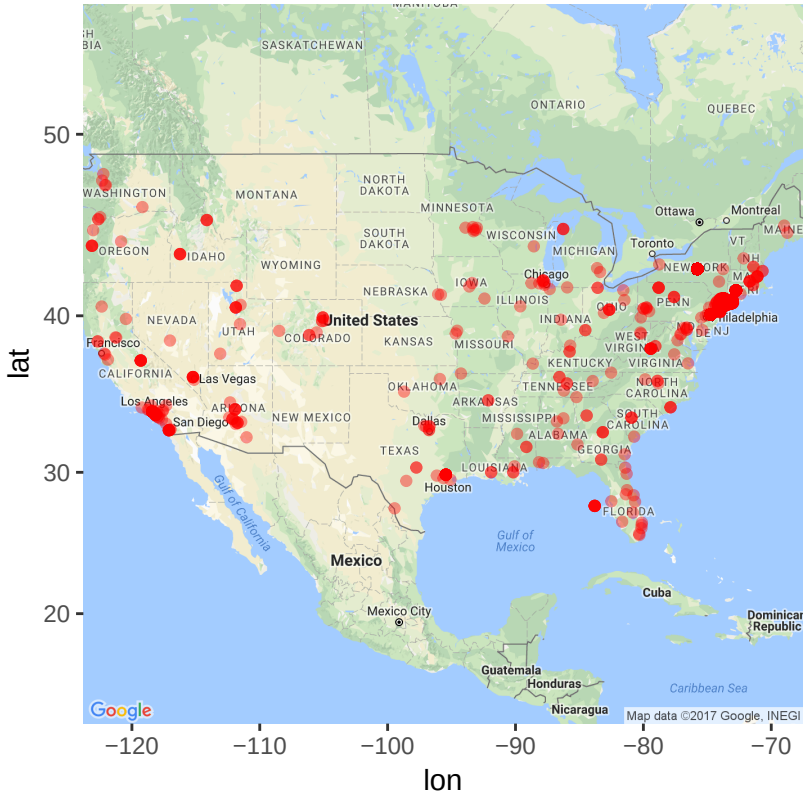

```
CAplot
```

California


```
USAplot
```

```
## Warning: Removed 30016 rows containing missing values (geom_point).
```

US Map for the Total Dataset




```
#Analysis about states and USA plots goes here
```

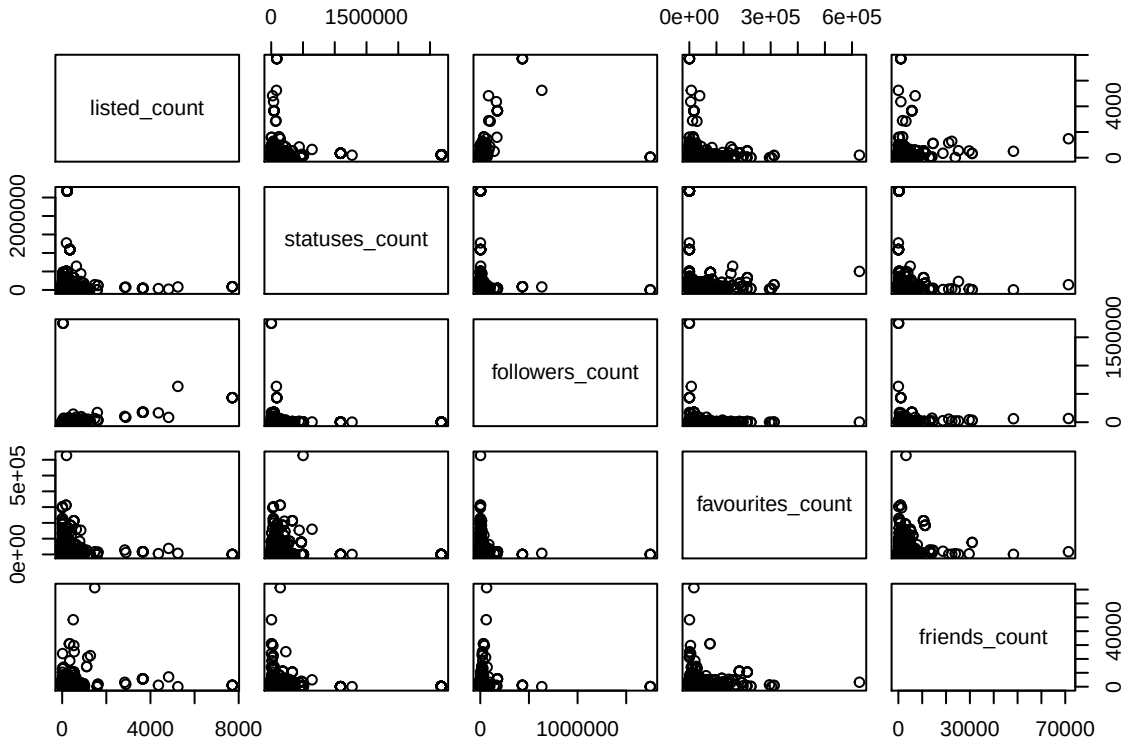
```
###Adding a new variable named "state" for future usage
ny[,"state"] <- rep("NY",nrow(ny));
nj[,"state"] <- rep("NJ",nrow(nj));
ca[,"state"] <- rep("CA",nrow(ca));

###getting city's names
ny$full_name <- as.character(ny$full_name);
nj$full_name <- as.character(nj$full_name);
ca$full_name <- as.character(ca$full_name);
tweets.df$full_name <- as.character(tweets.df$full_name);

### helping function that used for capturing cities
city_get <- function(x){
  city_name <- c()
  name <- strsplit(x$full_name,",")
  for(i in 1:nrow(x)){
    city_name <- c(city_name,name[[i]][1])
  }
  return(city_name)
}

ny[,"city"] <- factor(city_get(ny));
nj[,"city"] <- factor(city_get(nj));
ca[,"city"] <- factor(city_get(ca));
tweets.df[,"city"] <- factor(city_get(tweets.df))
```

```
total_us <- filter(tweets.df,lang=="en")
```

```
###Counting tweets for every city in each states
count <- summary(ny$city)[1:15];ny_city_count <- as.data.frame(count)
count <- summary(nj$city)[1:15];nj_city_count <- as.data.frame(count)
count <- summary(ca$city)[1:15];ca_city_count <- as.data.frame(count)
###Do the same thing towards whole United States region
count <- summary(tweets.df$city)[1:15];us_city_count <- as.data.frame(count)


NYCplot <- ggplot(ny_city_count,aes(reorder(rownames(ny_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("NY")
#nycityplot
NJCyplot <- ggplot(nj_city_count,aes(reorder(rownames(nj_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("NJ")
#njcityplot
CACplot <- ggplot(ca_city_count,aes(reorder(rownames(ca_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("CA")
#cacityplot
#grid.arrange(nycityplot, njcityplot, cacityplot, ncol=3)

USCplot <- ggplot(us_city_count,aes(reorder(rownames(us_city_count),count),count))+
  geom_bar(stat = "identity")+coord_flip()+xlab("City")+ggtitle("United States")

#uscityplot
```

```
#Analysis about cities and USA tweets count goes here
```

```
pairs(~listed_count+statuses_count+followers_count+favourites_count+friends_count, data=total_us)
```



```r
cor(total_us[,c(3,5,6,7,8)])
```

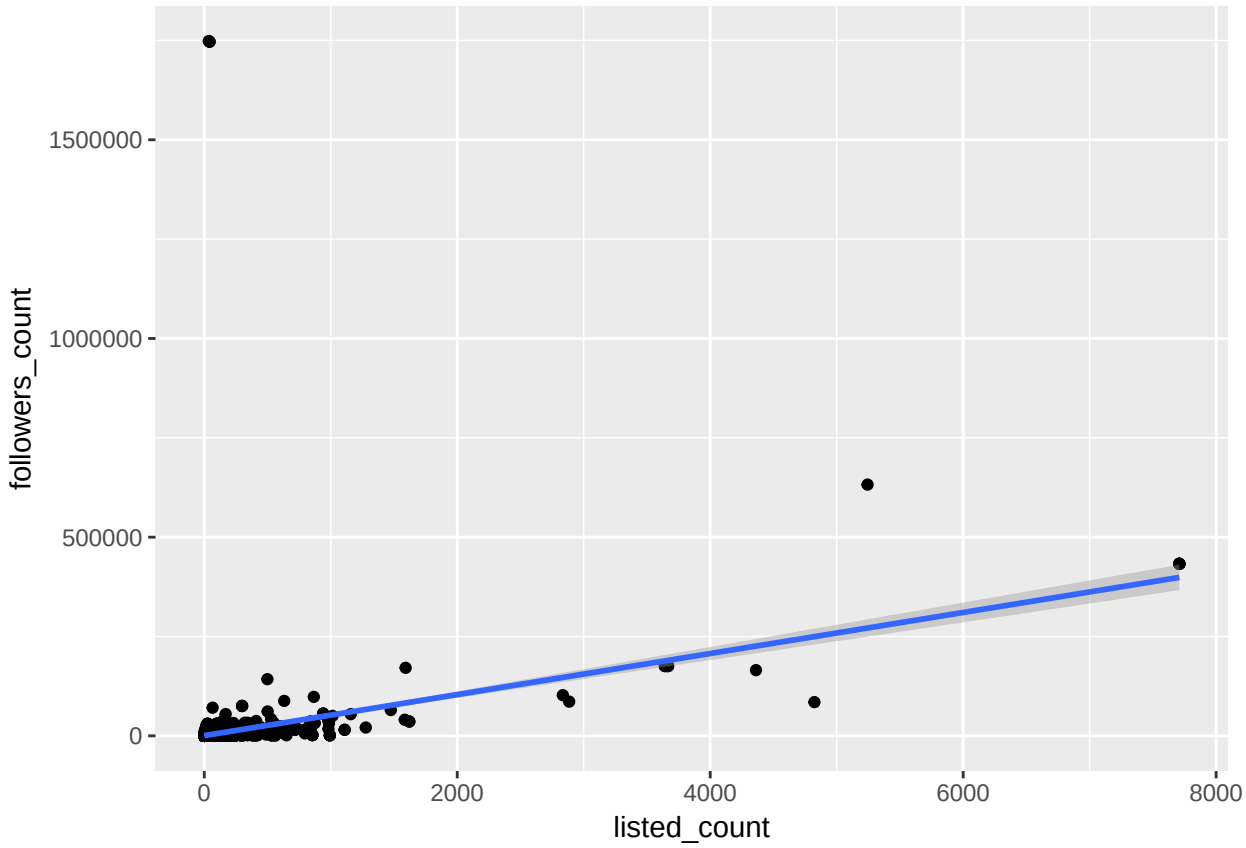
```
##                  listed_count statuses_count followers_count
## listed_count       1.00000000    0.076124932     0.357339575
## statuses_count     0.07612493    1.000000000    -0.001358509
## followers_count    0.35733957   -0.001358509     1.000000000
## favourites_count   0.03683789    0.028239280    -0.002667248
## friends_count      0.15308667   -0.013025551     0.043717263
##                  favourites_count friends_count
## listed_count          0.036837887    0.15308667
## statuses_count        0.028239280   -0.01302555
## followers_count      -0.002667248    0.04371726
## favourites_count      1.000000000    0.17700534
## friends_count         0.177005339    1.00000000
```

```r
#pairs and correlations goes here
```

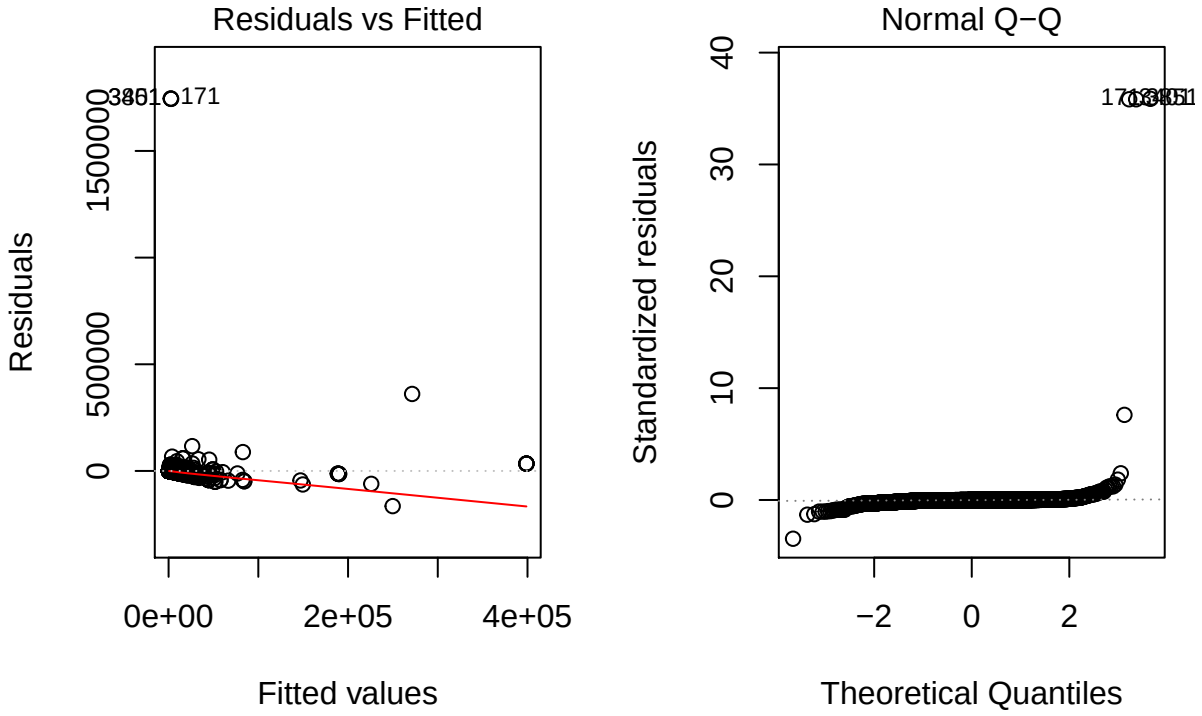
```r
model_fl <- lm(followers_count~listed_count,data=total_us)
summary(model_fl)
```

```
##
## Call:
## lm(formula = followers_count ~ listed_count, data = total_us)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -164852   -1362    -551    -254 1746106
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   532.863    787.149   0.677    0.498
## listed_count   51.660      2.137  24.171   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 48690 on 3991 degrees of freedom
## Multiple R-squared:  0.1277, Adjusted R-squared:  0.1275
## F-statistic: 584.2 on 1 and 3991 DF,  p-value: < 2.2e-16
```

```r
regplot <- ggplot(total_us,aes(x=listed_count,y=followers_count))+geom_point()+geom_smooth(method = "lm
regplot
```




```
#analysis of regression goes here
par(mfrow=c(1,2))
plot(model_fl,1)
plot(model_fl,2)
```



```r
#wordcloud

total_us_word <- total_us
total_us_word <- total_us_word[total_us_word$lang=="en",]
total_us_word <- total_us_word[total_us_word$geo_enabled==TRUE,]
total_us_word <- total_us_word[total_us_word$place_lat >25 &
                    total_us_word$place_lat <50 & total_us_word$place_lon > -125 &
                    total_us_word$place_lon< -66,]
total_us_word$geo_enabled <- NULL

# build a corpus, and specify the source to be character vectors
myCorpus <- Corpus(VectorSource(total_us_word$text))
# remove anything other than English letters or space(!!!)
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))
# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
# remove stopwords
myStopwords <- c(stopwords('english'),"use", "see", "used", "via", "amp","im")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
# remove extra whitespace
myCorpus <- tm_map(myCorpus, stripWhitespace)
# remove punctuation
myCorpus <- tm_map(myCorpus, removePunctuation)

# Build Term Document Matrix
tdm <- TermDocumentMatrix(myCorpus, control = list(wordLengths = c(1, Inf)))
term.freq <- rowSums(as.matrix(tdm))
term.freq2 <- subset(term.freq, term.freq >= 5)
df <- data.frame(term = names(term.freq2), freq = term.freq2)
m <- as.matrix(tdm)
```

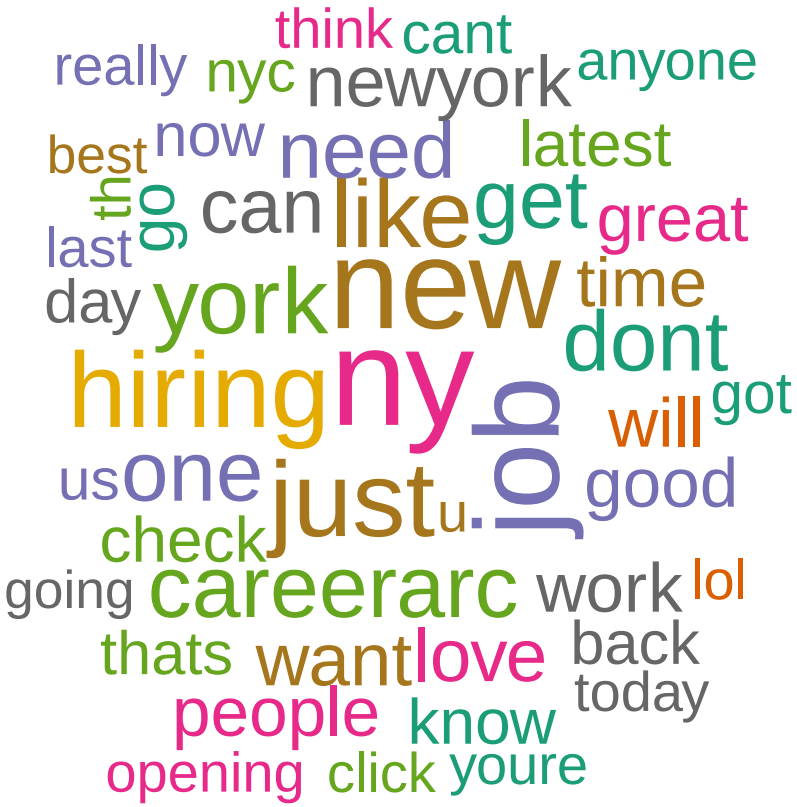
```r
# calculate the frequency of words and sort it by frequency
word.freq <- sort(rowSums(m), decreasing = T)

# plot word cloud
wordcloud(words = names(word.freq),
          freq = word.freq,
          max.words = 50,
          min.freq = 3,
          scale = c(4.5, 1),
          colors = brewer.pal(8, "Dark2"),
          random.color = T,
          random.order = F)
```

```
## Warning in wordcloud(words = names(word.freq), freq = word.freq, max.words
## = 50, : realdonaldtrump could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = names(word.freq), freq = word.freq, max.words
## = 50, : incident could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = names(word.freq), freq = word.freq, max.words
```

```
## = 50, : business could not be fit on page. It will not be plotted.
```

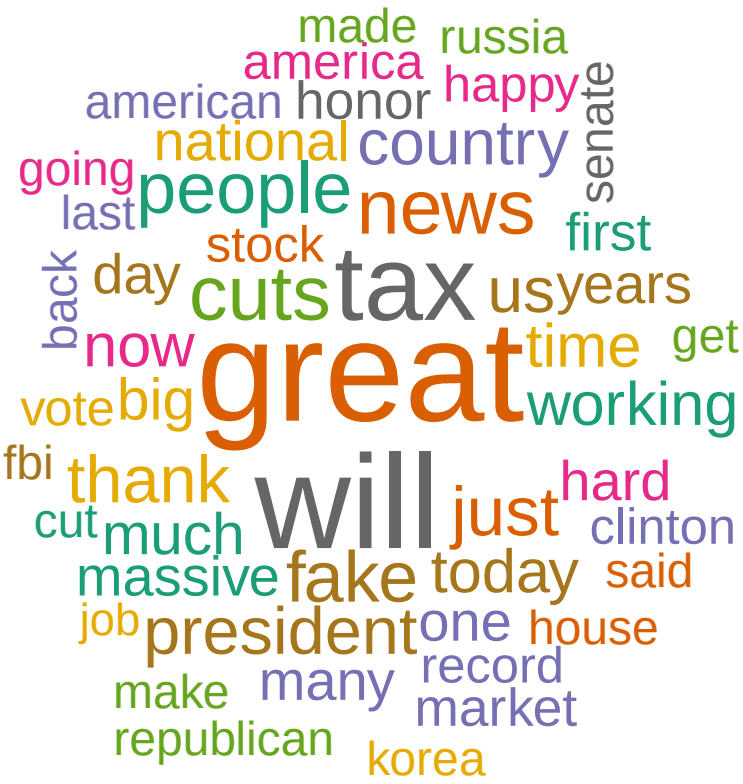



```
#analysis of wordcloud goes here
```

```
###Also, on the other hand, let's see President Trump his wordcloud analysis

###Here I used a new Twitter Application Api keys for different purpose, I also shouldn't have included

#api_key             <- "RE3jcprdcB3YwIXwcCG2rHPmj"
#api_secret          <- "rPeMG3nqOwDVKtXOuIsd4czCZjJd1eBh9BHqPWuUf8xBDaIY4j"
#access_token        <- "3134023545-cJbCbhXSeklrnVtEKU2yvv3bTI7APn93As93WiS"
#access_token_secret <- "glaUQunnz9e2QQT2gpKNgJHWtqKGRBK12JP9HtN44rCdT"
#setup_twitter_oauth(api_key, api_secret, access_token, access_token_secret)

###I comment this chunk out for avoid unneccessary time when kniting the whole project, but you are wel
###uncomment them for running the code throughly. However, since I will store Trump's tweets later and
###There won't be any problem you just leave this chunk commented out.

#user_id <- "@realDonaldTrump"
#trump_tweets_raw <- userTimeline(user_id, n = 3200)

#save(trump_tweets_raw, file = "trump_tweets_raw.Rdata")
```

```
load("trump_tweets_raw.Rdata")

# remove retweets
trump_tweets_df <- twListToDF(strip_retweets(trump_tweets_raw, strip_manual = TRUE, strip_mt = TRUE))

# normalize data
```

```r
trump_tweets_df$text <- iconv(trump_tweets_df$text, 'latin1', 'ASCII', 'byte')

Trump_Corpus <- Corpus(VectorSource(trump_tweets_df$text))      # build a corpus, and specify the sourc

removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)     # remove anything other than En
Trump_Corpus <- tm_map(Trump_Corpus, content_transformer(removeNumPunct))

Trump_Corpus <- tm_map(Trump_Corpus, stripWhitespace)     # remove extra whitespace

Trump_Corpus <- tm_map(Trump_Corpus, content_transformer(tolower))      # convert to lower case

myStopwords <- c(stopwords('english'),"use", "see", "used", "via", "amp","im")      # remove stopwords
Trump_Corpus <- tm_map(Trump_Corpus, removeWords, myStopwords)

rm_url   <- function(x) gsub("http[^[:space:]]*", "", x)      # remove urls
Trump_Corpus <- tm_map(Trump_Corpus, content_transformer(rm_url))

Trump_Corpus <- tm_map(Trump_Corpus, removePunctuation)     # remove punctuation



# build term document matrix
Trump_tdm <- TermDocumentMatrix(Trump_Corpus, control = list(wordLengths = c(1, Inf)))
Trump_term.freq <- rowSums(as.matrix(Trump_tdm))
Trump_term.freq <- subset(Trump_term.freq, Trump_term.freq >= 5)
df <- data.frame(term = names(Trump_term.freq), freq = Trump_term.freq)
Trump_m  <- as.matrix(Trump_tdm)

# calculate the frequency of words and sort it by frequency
Trump_word_freq <- sort(rowSums(Trump_m), decreasing = T)

wordcloud(words = names(Trump_word_freq),
          freq  = Trump_word_freq,
          scale = c(4.5, 1),
          max.words = 50,
          min.freq = 3,
          colors = brewer.pal(8, "Dark2"),
          random.color = T,
          random.order = F)
```




```
# remove sparse terms
tdm2 <- removeSparseTerms(tdm,sparse = 0.965)
# showing the words that are left gor the analysis
print(dimnames(tdm2)$Terms)
```

```
## [1] "like"            "hiring"          "realdonaldtrump" "just"
## [5] "job"             "ny"              "new"
```

```
m2 <- as.matrix(tdm2)
m3 <- t(m2)   # transpose the matrix to cluster documents
set.seed(122)   # set a fixed random seed
k <- 6   # number of clusters
kmeansResult <- kmeans(m3,k)
round(kmeansResult$centers, digits = 3)   # cluster centers
```

```
##    like hiring realdonaldtrump  just   job    ny   new
## 1 0.005  0.000           0.015 0.036 0.005 0.260 1.184
## 2 0.000  0.003           0.078 0.048 0.000 0.000 0.000
## 3 1.059  0.000           0.086 0.112 0.007 0.000 0.020
## 4 0.016  0.000           0.000 0.016 0.143 1.063 0.016
## 5 0.000  0.438           0.031 0.031 1.016 0.000 0.016
## 6 0.000  1.000           0.000 0.000 0.966 1.021 0.000
```

```
for(i in 1:k){
  cat(paste("cluster ", i, ": ", sep = ""))
  s <- sort(kmeansResult$centers[i,],decreasing = T)
  cat(names(s)[1:5],"\n")
  #print the tweet of every cluster
}
```

```
## cluster 1: new ny just realdonaldtrump like
```

```
## cluster 2: realdonaldtrump just hiring like job
## cluster 3: like just realdonaldtrump new job
## cluster 4: ny job like just new
## cluster 5: job hiring realdonaldtrump just new
## cluster 6: ny hiring job like realdonaldtrump
```

```r
# cluster terms
distMatrix <- dist(scale(m2))
fit <- hclust(distMatrix, method = "complete")
# show cluster dendrogram
p <- plot(fit, xlab="")
p <- rect.hclust(fit, k=6)
```