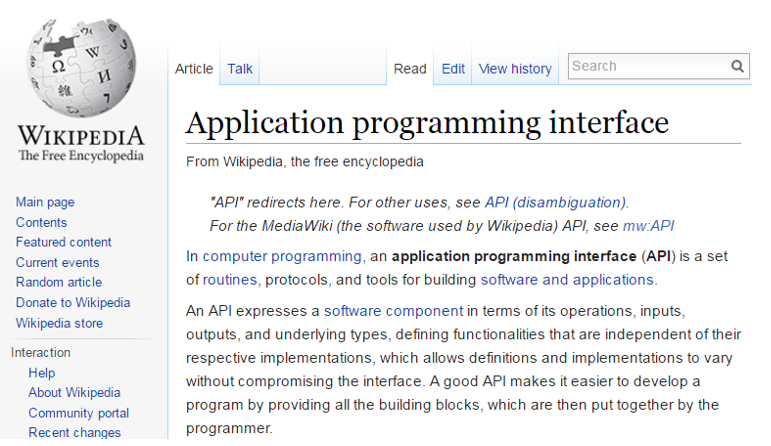
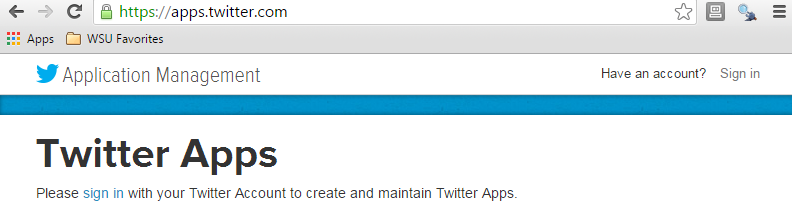
R: Twitter/Text Mining

Setting up an API, i.e. Twitter Side

Twitter uses an Application Programming Interface (API) that allows developers to pull Twitter data.



To setup an API, go to <http://apps.twitter.com>, sign in using an existing Twitter account.



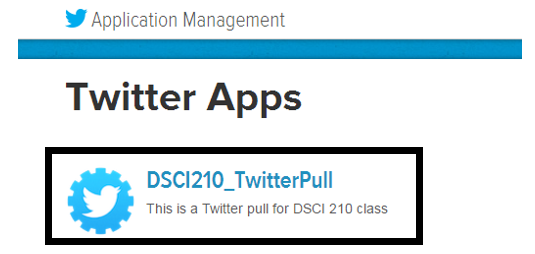
Once logged in, select Create New App



Next, specify the requested information to create a new application.

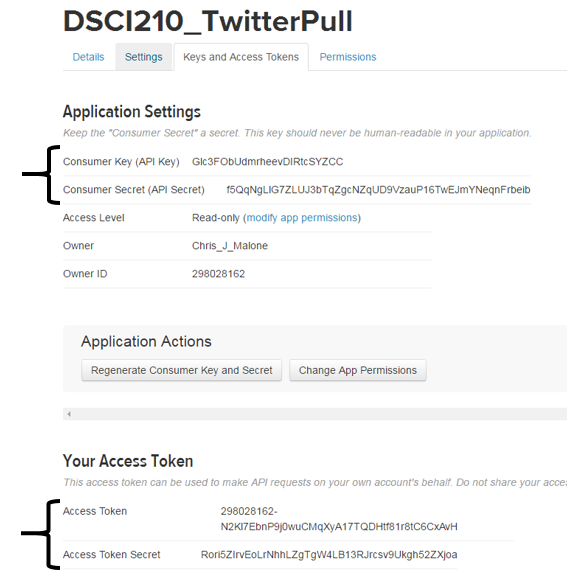


DSCI210\_TwitterPull is the newly created application.



After a successful application has been setup, the following information is needed to successfully access Twitter data from R.

* Consumer Key
* Consumer Secret
* Access Token
* Access Token Secret



Working with Twitter Pulls in R

The following packages/libraries will be used to analyze Twitter data in R.

library(twitteR)

library(RCurl)

library(tm)

library(wordcloud)

The following information is obtained from the DSCI210\_TwitterPull application on Twitter side.

#Specified from Twitter Account - DSCI\_210 is app name

consumer\_key <- "Glc3FObUdmrheevDIRtcSYZCC"

consumer\_secert <- "f5QqNgLlG7ZLUJ3bTqZgcNZqUD9VzauP16TwEJmYNeqnFrbeib"

token\_key <- "298028162-N2Kl7EbnP9j0wuCMqXyA17TQDHtf81r8tC6CxAvH"

token\_secert <- "Rori5ZIrvEoLrNhhLZgTgW4LB13RJrcsv9Ukgh52ZXjoa"

Setting up the OAUTH connection in R.

*Note*: OAUTH is an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications.

#Setting up OAUTH in R

setup\_twitter\_oauth(consumer\_key,consumer\_secert,token\_key,token\_secert)

Getting a Twitter pull

#Gettting a pull on recent Ecuador Earthquake

pull <- searchTwitter("#EcuadorEarthquake", n=1000, lang="en")

The object returned by the searchTwitter() function is a list.

#Checking to see if object is indeed a list

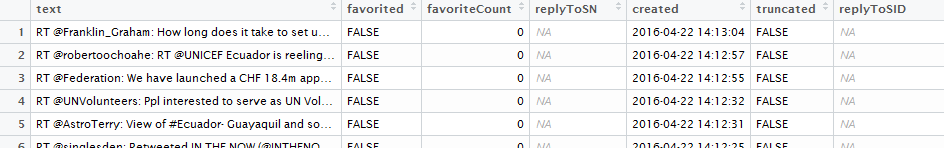
is.list(pull)

This (somewhat unstructured) list can be converted a standard data.frame using the following.

#Converting list to dataframe

df <- do.call("rbind", lapply(pull, as.data.frame))

View(df)



Writing data.frame to comma delimited file

#The write.csv() function to write this data.frame into a \*.csv file

write.csv(df,file="C:/Teaching/DSCI210/Datasets/TwitterPulls/EcuadorEarthquake.csv")

A summary of screenname

Get # posts by screenName

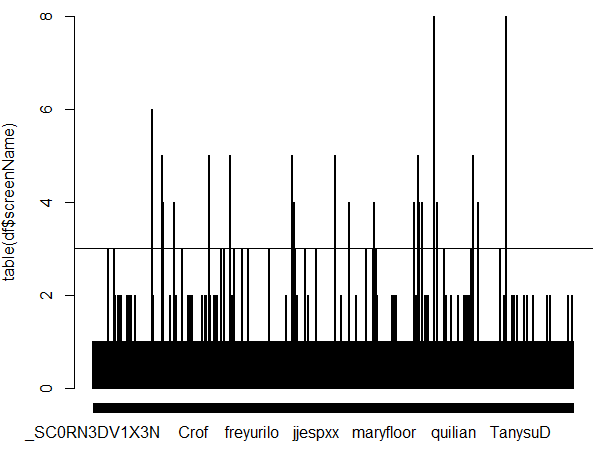
table(df$screenName)

#Plotting results

plot(table(df$screenName))

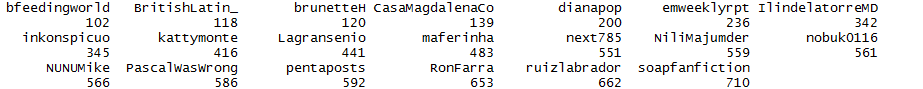
#adding a horizontal line at 3

abline(h=3)



#Idenitfy screenNames with more than 3 counts

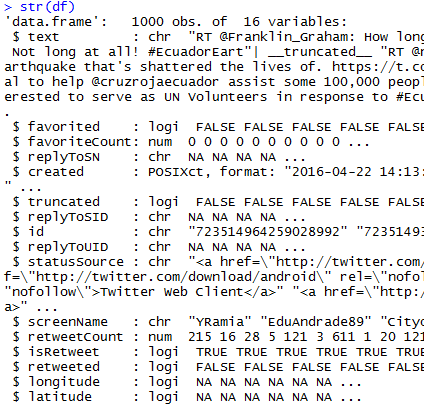
which(table(df$screenName)>3)



A summary of created

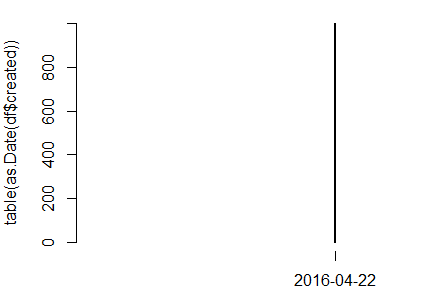
#Getting the structure of our data.frame

str(df)



#Plotting twitter pull across days

plot(table(as.Date(df$created)))

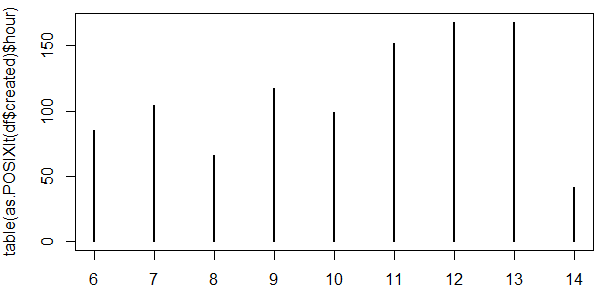


#The following can be used to pull hour off created variable

as.POSIXlt(df$created)$hour

#Next, table/plot outcome

plot(table(as.POSIXlt(df$created)$hour))



Text Mining Procedures

#usign the tm library for text mining

myCorpus <- Corpus(VectorSource(df$text))

#Clean up text using the tm\_map() function

myCorpus <- tm\_map(myCorpus, tolower)

myCorpus <- tm\_map(myCorpus, removePunctuation)

myCorpus <- tm\_map(myCorpus, removeNumbers)

Necessary to get rid of common English words. The stopwords() function can be used to accomplish this in R.

#Getting rid of common english words

myStopwords <- c(stopwords('english'))

myCorpus <- tm\_map(myCorpus, removeWords, myStopwords)

The following is used to convert the Corpus object into a TermDocumentMatrix which is then converted to a matrix, and eventually a data.frame.

myDtm <- TermDocumentMatrix(myCorpus, control = list(minWordLength = 1))

m <- as.matrix(myDtm)

v <- sort(rowSums(m), decreasing=TRUE)

myNames <- names(v)

#Creating data.frame for wordcloud

d <- data.frame(word=myNames, freq=v)

wordcloud(d$word, d$freq, min.freq=3)

A Wordcloud of text from Twitter pull



The wordcloud() function allows specification of a minimum frequency when plotting. This can be used to identify the most common words.

wordcloud(d$word, d$freq, min.freq=40)

